Deep Neural Network Attribution Methods for Leakage Analysis and Symmetric Key Recovery

Benjamin Hettwer[1,2(✉)], Stefan Gehrer[1], and Tim Güneysu[2]

[1] Robert Bosch GmbH, Corporate Sector Research, Stuttgart, Germany
{benjamin.hettwer,stefan.gehrer}@de.bosch.com
[2] Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum, Germany
tim.gueneysu@rub.de

Abstract. Deep Neural Networks (DNNs) have recently received significant attention in the side-channel community due to their state-of-the-art performance in security testing of embedded systems. However, research on the subject mostly focused on techniques to improve the attack efficiency in terms of the number of traces required to extract secret parameters. What has not been investigated in detail is a constructive approach of DNNs as a tool to evaluate and improve the effectiveness of countermeasures against side-channel attacks. In this work, we close this gap by applying attribution methods that aim for interpreting Deep Neural Network (DNN) decisions in order to identify leaking operations in cryptographic implementations. In particular, we investigate three different approaches that have been proposed for feature visualization in image classification tasks and compare them regarding their suitability to reveal Points of Interest (POIs) in side-channel traces. We show by experiments with four separate data sets that the three methods are especially interesting in the context of side-channel protected implementations and misaligned measurements. Finally, we demonstrate that attribution can also serve as a powerful side-channel distinguisher leading to a successful retrieval of the secret key with at least five times fewer traces compared to standard key recovery in DNN-based attack setups.

Keywords: Side-Channel Attacks · Deep Learning · Machine Learning · Leakage analysis

1 Introduction

Side-Channel Analyis (SCA) is a technique by which an adversary circumvents the security assumptions of a cryptographic system by analyzing its physical properties. In this regard, timing [19], power consumption [18], and Electromagnetic (EM) emanation [3] have been investigated to reveal secret parameters. In order to decrease the information leakage of cryptographic implementations,

K. G. Paterson and D. Stebila (Eds.): SAC 2019, LNCS 11959, pp. 645–666, 2020.
https://doi.org/10.1007/978-3-030-38471-5_26

researchers and industry came up with dedicated countermeasures which can be roughly classified into *Masking* and *Hiding* [21]. However, more powerful attacks demonstrated that even side-channel protected implementations may still be vulnerable [23].

A new line of work that deals with the application of DNNs for side-channel evaluation of protected and unprotected cryptographic implementations has been presented recently. In general, DNNs provide a powerful method for a variety of different real-world problems such as image classification [16], natural language processing [31], and medical applications [10]. In the context of SCA, especially Convolutional Neural Networks (CNNs) have shown to be advantageous over standard analyzing tools like TAs in different settings (for example in case of desynchronized traces or an unknown leakage model) [8,17,20,30].

Due to the black-box nature of DNNs, understanding the operation of Deep Learning (DL) models is an active area of research. It is evident that safety critical applications such as medicine or autonomously driving cars need to be validated exhaustively prior to their actual release. Regarding image classification, several so-called *attribution* or *heatmapping* methods have been proposed to explain the predictions of DNNs. The idea is to visualize the pixels of an input image which had the greatest influence of classifying it into a certain category. By doing so, it is possible to make the decisions of a DNN more transparent and explainable as it helps to identify if a DNN was able to learn the "correct" features during training.

In this work, we analyze different attribution methods of DNNs for their suitability in SCA. More specifically, we investigate *saliency maps* [28], *occlusion* [32], and *Layer-wise Relevance Propagation (LRP)* [6] to extract the features or POIs from a trained DNN which are most informative for symmetric key recovery. Proper POI detection is commonly considered as crucial for the success of profiled SCA (i.e. attacks which assume an adversary with access to a profiling device which is similar to the target) and usually performed as a preprocessing step ahead of the actual attack [24]. Here, we take another perspective and show a technique to compute the relevance of sample points in side-channel traces after the profiling step, which is applicable even in case of employed countermeasures. This can be seen as a constructive method for evaluators to identify the operations of the implementation under test causing the highest leakage. Furthermore, we demonstrate that attribution methods can also be used as a distinguisher in DNN-based SCA.

1.1 Contribution

The contributions of this paper are manifold:

1. We show a generic technique that can be used to calculate the POIs from a trained DNN. It is generic in a sense that it is independent of the actual used attribution method.
2. Based on the commonly known Key Guessing Entropy (KGE), we define two novel metrics tailored to the specifics of DNNs in order to quantitatively asses how well the selection of POIs is done.

3. We compare three attribution methods on four different data sets: a hardware implementation (with and without jitter in the traces) and two protected software implementations of the Advanced Encryption Standard (AES). Our results confirm that attribution methods are more suitable to extract POIs from protected implementations than a standard technique from the side-channel domain.
4. We show how LRP can be embedded in profiled attack setups to distinguish between correct and incorrect key hypotheses. We demonstrate by practical experiments that our proposed method is more efficient than using the network predictions directly for key recovery.

1.2 Related Work

Identifying POIs in side-channel traces has been traditionally studied in the context of TAs introduced by Chari et al. [9], in order to reduce the computational overhead during calculation of the covariance matrices. In particular, *Difference of Means (DOM)* [9], *Sum Of Squared pairwise T-differences (SOST)*, *Sum Of Squared Differences (SOSD)* [14], and *Principal Component Analysis (PCA)* [5] have been proposed for that purpose. Another common strategy for POIs selection is based on Pearson correlation, whereby the importance of sample points is measured by the correlation coefficient of the actual power consumption and some key-dependent target intermediate value [21]. Picek et al. investigated the so-called Wrapper and Hybrid methods stemming from the machine learning domain to determine a suitable subset of features in order to boost the efficiency of side-channel attacks [24].

In contrast to most of the aforementioned approaches, leakage detection techniques such as *Test Vector Leakage Assessment (TVLA)* aim for revealing data-dependent information leakage independent of any power model or intermediate value [11]. It can thus be considered as a complementary tool to identifying leaking operations in a first step, then performing an actual attack to check whether the found leakage can be exploited for a successful key extraction.

Very recently, Masure et al. came up with an idea similar to our work: POIs visualization after successful training of a neural network [22]. Their method based on sensitivity analysis is related to the saliency technique. In this work, we conduct a more comprehensive study of DNN attribution methods for side-channel analysis by comparing different techniques using a novel framework for POIs selection and evaluation. We additionally present, to the best of our knowledge, the first SCA distinguisher based on DNN attribution.

1.3 Structure of the Paper

The structure of the paper is as follows: In Sect. 2, we shortly recap DL-based SCA and give an introduction to DNN attribution methods. In Sect. 3, we present our approach for POIs visualization and apply them to four data sets for leakage analysis. In Sect. 4, we evaluate the quality of side-channel heatmaps. In Sect. 5, we describe our attribution-based technique for key recovery and use them to

attack an unprotected and a protected implementation of the AES. The last section summarizes the paper and gives insights on possible future work.

2 Preliminaries

This section outlines the foundations of DL-based SCA. Furthermore, background and motivation of DNN attribution methods is provided.

2.1 Deep Learning-Based Profiled Side-Channel Analysis

Profiled SCA is divided in two stages: profiling phase and key recovery phase. In the former, the adversary takes advantage of a profiling device on which he can fully control input and secret key parameters of the cryptographic algorithm. He uses this to acquire a set of N_P profiling side-channel traces $\mathbf{x} \in \mathbb{R}^D$, where D denotes the number of sample points in the measurements. Let $V = g(p, k)$ be a random variable representing the result of an intermediate operation of the target cipher which depends partly on public information p (plaintext or ciphertext chunk) and secret key $k \in \mathcal{K}$, where $\mathcal{K}$ is the set of possible key values. V is assumed to have an influence on the deterministic part of the side-channel measurements. In the context of DL or Machine Learning (ML) in general, the goal of the attacker during the profiling phase is to construct a classifier that estimates the probability distribution $f(\mathbf{x}) \approx \mathrm{P}[V|\mathbf{x}]$ using the training set $\mathcal{D}_{Train} = \{\mathbf{x}_i, v_i\}_{i=1,\ldots,N_P}$.

During the key recovery phase, the adversary generates a new set $\mathcal{D}_{Attack}$ with N_A attack traces from the actual target device (which is structurally identical to the profiling device) whereby the secret key k is fixed and unknown. In order to retrieve it, Log-likelihood (LL) scores over all possible key candidates $k^* \in \mathcal{K}$ are computed and combined to:

$$\mathbf{d}^{LL}(\mathcal{D}_{Attack}, f) = \sum_{i=1}^{N_A} \log f(\mathbf{x}_i)[g(p_i, k^*)] \tag{1}$$

The k-th entry in score vector $\mathbf{d}^{LL}$ corresponds to the correct key candidate [25]. A commonly known metric in profiled SCA is the so-called KGE or key rank function which quantifies the difficulty to retrieve the correct value of the key regarding the required number of traces from $\mathcal{D}_{Attack}$ [29]. It is computed by performing a ranking of $\mathbf{d}$ after the evaluation of each attack trace. Intuitively, the faster the key rank converges to one, the more powerful is the attack.

2.2 Deep Neural Network Attribution Methods

In recent years there has been a growing interest in neural networks having several layers of neurons stacked upon each other, which are commonly referred to as DNNs. They represent a particular powerful type of ML techniques that are able to represent the learning task as a nested hierarchy of concepts, where more

abstract concept representations are built from simpler ones. Throughout the paper we assume a DNN as a classification function that takes an input vector $\mathbf{x} = [x_1, \ldots, x_D] \in \mathbb{R}^D$ and produces an output $f(\mathbf{x}, \mathbf{W}) = [f_1(\mathbf{x}), \ldots, f_C(\mathbf{x})]$, where C denotes the number of output neurons (=number of categories). The parameters $\mathbf{W}$ are learned during training to approximate f from a broad class of functions to map $\mathbf{x}$ to the desired output. Training a DNN is usually done in an iterative, multi-step process by which the parameters of the network are optimized to minimize a loss function, which depicts the difference between the expected output (i.e. labels) and the prediction result. In practice, optimizer algorithms such as Stochastic Gradient Descent (SGD) or ADAM are employed for that purpose [15].

Given a specific class c, attribution methods for DNNs aim to determine the influence $\mathbf{r}^c = [r_1^c, \ldots, r_D^c] \in \mathbb{R}^{\mathbb{D}}$ of each data point x_i of an input vector (sometimes also called features) with respect to the output neuron f_c [4]. The result can be visualized, e.g., as a heatmap that indicates the features that contributed positively and/or negatively to the activation of the target output. In the following, we briefly summarize three recent attribution methods that have been proposed for calculating heatmaps for 2D images, which we later apply to 1D side-channel traces. We have chosen these three methods for two reasons: First, they are not designed for a specific type of DNN architecture (as for example Grad-CAM [27] and deconvolution [32] for CNNs) but generally applicable to several types of DNN and activation units. Second, we intend to compare techniques coming from different classes of attribution methods, i.e., a gradient-based method (saliency maps), a LRP-based method, and one that is based on the perturbation of the input (occlusion).

Saliency Maps. Simonyan et al. established saliency maps in 2013 in order to highlight class discriminative areas of images captured by CNNs [28]. To this end, the norm value $\|\cdot\|_\infty$ over partial derivatives of the output category is computed with respect to the input features:

$$r_i^c = \left\| \frac{\partial f_c(\mathbf{x})}{\partial x_i} \right\|_\infty \tag{2}$$

Partial derivatives are found by running the back-propagation algorithm throughout the layers of the network. Intuitively, the magnitude of the derivative indicates which features need to be modified the least to affect the class score the most. However, since the sign of the derivative is lost when using the norm, only positive attributions of input features can be detected with the saliency method. It consequently provides only local explanations, e.g., by indicating the features that make a car more/less a car, but no global explanations which features compose a car [26].

Layer-wise Relevance Propagation (LRP) was introduced by Bach et al. as a general concept to achieve a pixel-wise decomposition of the prediction $f(\mathbf{x})$ as a term of the separate input dimensions [6]:

$$f(\mathbf{x}) \approx \sum_{i=1}^{N} r_i \tag{3}$$

where $r_i > 0$ can be interpreted as positive evidence for the presence of a structure in the picture, and $r_i < 0$ as evidence for its absence. The algorithm follows a conservation principle that proceeds layer by layer, by which the prediction score f_c is propagated recursively through the network until the input layer is reached. For redistributing a layers relevance onto the preceding layer, Bach et al. proposed the following propagation rule:

$$r_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j} + \epsilon \cdot sign(\sum_{i'} z_{i'j})} r_j^{(l+1)} \tag{4}$$

Here, $r_i^{(l)}$ denotes the relevance associated with the ith neuron in layer l received from the jth neuron in the layer $l+1$, and $z_{ij} = a_i^{(l)} w_{ij}^{(l,l+1)}$ the weighted activation of neuron i onto neuron j in the next layer. The ϵ term is added in order to cope with numerical instabilities in case the denominator tends to zero. The idea of the propagation rule is that neurons of the preceding layer that gave an larger activation to neurons of the higher layer in the forward pass get more relevance on the backward pass.

Compared to gradient-based attribution methods such as saliency, LRP is applicable to any network with monotonous activation units (even non-continuous). LRP furthermore provides a clear interpretation by indicating the features for and against a category [26]. We will see later in the paper that this property can be exploited to construct a side-channel distinguisher.

Occlusion sensitivity analysis as proposed by Zeiler and Vergus attempts to identify the location of objects in images by systematically occluding different regions of the input with a grey square, and monitoring the classification result [32]. Therefore, the relevance of input features can be described as probability drop of the correct class with respect to the position of the grey patch. It is evident that the runtime and result of the algorithm heavily depends on the number of features that are removed together per iteration.

In the remainder of the paper, we refer to the *1-occlusion* approach given in [4]. In 1-occlusion, exactly one feature of the input data is set to zero per time, while the effect on the output is measured. More formally, the attribution of a single feature can be calculated as:

$$r_i^c = f_c(\mathbf{x}) - f_c(\mathbf{x}[i] = 0) \tag{5}$$

where $\mathbf{x}[i] = v$ indicates an input vector whose ith data point has been replaced with the value v. We have chosen 1-occlusion since the leakage information present in side-channel traces is often concentrated in a small number of sample points [24].

3 Attribution for POI Analysis

In this section, we describe a method to generate heatmaps for side-channel traces using DNN attribution and apply it to four data sets.

3.1 Side-Channel Heatmaps

DNN-based SCA aimed mainly for symmetric key recovery in the past. In this context, especially CNNs have shown to be a suitable tool due to the fact that they are able to automatically extract the areas in the side-channel traces which contain the most information [8,20]. Furthermore, CNNs are able to detect POIs that would normally not be considered by an attacker. These can be used by the network in conjunction with the areas that contain a lot of leakage to make the attack even more efficient, i.e., requiring less/smaller traces for a successful attack. When using established SCA techniques such as TAs, the selection of POIs has to be done manually as a preprocessing step ahead of the actual attack. This is not only tedious, but also error prone as proper POI selection has shown to have a significant impact on the attack efficiency [33]. Furthermore, in case of first-order secure implementations without access to the mask values during the profiling step, the adversary has to combine the leakage location of the mask and the masked target intermediate value when considering a second-order Correlation Power Analysis (CPA) for POIs detection. This requires to combine all possible combination of sample points and the overhead grows roughly quadratically with the size of the traces [21].

In this section, we go one step further and describe a way to extract the POIs from a trained Convolutional Neural Network (CNN) (or any type of DNN) that have been considered as most discriminative to reveal the correct key, based on the attribution methods presented in the previous section. The approach works as follows and is summarized in (6): Given a trained DNN f, the relevance $\mathbf{r}^{C_k}$ for an input trace $\mathbf{x}$ is found by using one of the attribution methods mentioned in Sect. 2. C_k represents the output class under the correct key, i.e., the labels that have been used for training of f. This procedure is conducted for a set of N_{Attr} traces and the average relevance $\bar{\mathbf{r}} \in \mathbb{R}^D$ is calculated.

$$\bar{\mathbf{r}} = \frac{1}{N_{Attr}} \sum_{i=1}^{N_{Attr}} \mathbf{r}^{C_k}(\mathbf{x}_i, f) \tag{6}$$

Because $\bar{\mathbf{r}}$ has the same dimensionality as $\mathbf{x}$, it can be visualized as 1D side-channel heatmap plot. The information can be used, for example, to determine leaking operations in cryptographic implementations since it is easily possible to trace which operations are performed at which time intervals (at least in white-box evaluation settings). Another use case would be to identify relevant regions in the side-channel traces using only a subset of the available traces in a first step, in order to decrease the number of data points for the actual attack with the complete data set (and thus speed up calculations).

3.2 Experimental Results

We consider four data sets for the experiments of the paper: An unprotected hardware AES with and without jitter in the traces (denoted as AES-Serial and AES-Serial-Desync), and two protected software implementations of the AES (denoted as ASCAD and AES-RSM). An overview about the data sets is given in Table 1. We have created attribution heatmaps for all data sets according to (6) using the Python frameworks Keras [2] and DeepExplain [1]. Additionally, we have computed Pearson correlations as a baseline. The same DNN architecture has been used in all experiments in order to allow an unbiased evaluation. The employed network is a CNN which consists of four blocks of convolution and max-pooling operations followed by two fully-connected layers. Details about the network structure along with related training parameters are described in detail in Table 2 in the Appendix. As a preprocessing step ahead of training the CNN, we transformed the traces of all data sets to have zero mean and unit variance (sometimes referred to as data standardization).

Table 1. Overview of data sets

Data set	Sample points	Traces (Profiling)	KGE < 3
AES-Serial	1000	25 000	750
AES-Serial-Desync	950	50 000	100
ASCAD	700	50 000	500
AES-RSM	10 000	100 000	20

AES-Serial. AES-Serial denotes a set of power traces of an unprotected AES hardware design that have been acquired from a Xilinx ZYNQ UltraScale+ evaluation board. A single measurement contains 1000 data points representing approximately the time interval when the first AES round is calculated. Since it is commonly known that the most leakage in a hardware implementation is caused by register transitions, we have used the XOR of two consecutive S-Box outputs in the first round as target operation and consequently as labels for training. We have trained the network using 25 000 traces and subsequently calculated heatmaps with a subset of $N_{Attr} = 1000$ measurements for each of the attribution methods introduced in Sect. 2. N_{Attr} was set to this value since we observed no improvements when using more than 1000 traces for the calculation. The resulting heatmaps along with the corresponding correlation for the correct key hypothesis are shown in the first row of Fig. 1. From there, one can observe that the region around sample point 800 is considered as most informative by all three attribution methods as well as by the Pearson correlation. Interestingly, the saliency heatmap indicates a wider range of samples as important and additionally shows a second peak in the first half of the heatmap which

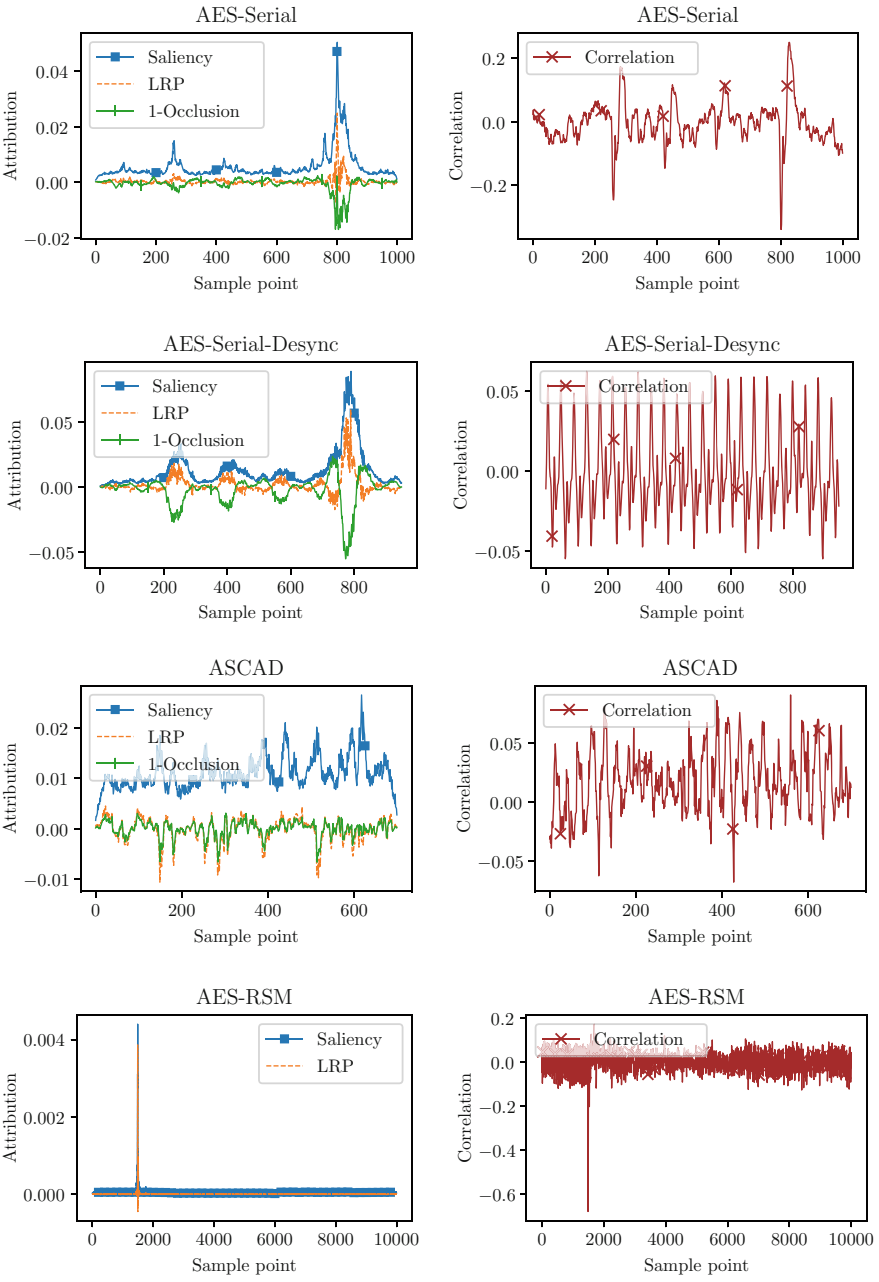

Fig. 1. Mean attributions for the four data sets (left column) and corresponding correlation analysis result (right column). Each curve has been calculated with 1000 traces.

is also visible in the correlation plot. We could easily backtrack by examining the implemented hardware layout that the found leakage is caused by unintended high routing fanouts between four registers of the state array. Considering this, it seems that correlation analysis gives better results than the attribution methods in this data set as four peaks are clearly visible in the correlation plot. We will see later in the paper that this assumption is correct.

AES-Serial-Desync. This data set is similar to the former one except that the traces have been artificially desynchronized in order to simulate jitter in the measurement setup. To this end, we have shifted each trace to the right with a random offset in the range $[0, 50]$. Cagli et al. demonstrated that CNNs are robust to such misalignments due to the spatial invariance property of the convolutional layers [8]. Here, our goal is to evaluate if the different attribution methods are still able to extract POIs in case of desynchronized traces. We examined during the experiments that our CNN architecture is indeed capable to learn a meaningful representation from the misaligned traces. However, more training traces have been required for a successful key recovery compared to the aligned AES-Serial data set (50 000 instead of 25 000).

Attribution and correlation heatmaps for the AES-Serial-Desync data set can be found in the second row of Fig. 1. One can notice that the attribution curves look similar to the curves for the aligned traces, while the correlation-based feature extraction totally fails in this case. Taking a closer look on the lefthand plots, it can be observed that the four expected peaks are even better visible than in the attribution heatmaps for the aligned traces. A second observation that can be made is that the peaks are broader compared to the curves plotted directly above. This due to the misalignment of the sample points and the fact that we have calculated then mean relevance over $N_{Attr} = 1000$ traces.

ASCAD is a public database of side-channel measurements and related metadata obtained from a first-order secured software AES implementation [25]. Each trace is composed of 700 sample points and the targeted intermediate result is the third byte of the masked S-Box output that is processed during the first round. The database is split in 50 000 training and 10 000 attack traces and we have used the complete former set for CNN training. Next, we calculated attribution heatmaps using $N_{Attr} = 1000$ measurements which are illustrated in the third row in Fig. 1. From there, it can be noticed that the heatmaps computed by LRP and 1-Occlusion are very similar in most areas, while the saliency heatmap shows a different characteristic with several peaks in regions where no attribution is found by the other techniques. However, the results look in general more noisy than for the former data sets. Considering the Signal-to-Noise Ratio (SNR) analysis that is given in [25], one would expect to see four regions with POIs: One for the processing of the masked S-box output in linear parts, one for the processing of the S-Box output masked with the output mask, and for the processing of the two masks each. This is mostly reflected by the LRP and 1-occlusion heatmaps. The example demonstrates that attribution methods may also help

to reverse-engineer internal structures of protected cryptographic implementations. As expected, correlation analysis without sample point combination fails to extract a meaningful pattern from the first-order secure implementation.

AES-RSM. The fourth data set we have analyzed is based on a secured software AES implementation which originates from the DPA Contest v4.2 [7]. It is equipped with two SCA countermeasures: a first-order secure masking scheme called *Rotating Sbox Masking (RSM)*, and shuffling of the S-Box execution order. All traces are composed of 10 000 sample points representing approximately the first one and a half rounds of an encryption operation and have been acquired on a ChipWhisperer-Lite board. Previous work showed that the implementation can be attacked very efficiently using a CNN with Domain Knowledge (DK), where the profiling is done directly regarding a byte of the secret key (i.e. key is used as label) and the related plaintext byte is given to the network alongside the trace [17]. We slightly adapted our CNN architecture used in the former experiments to this setting. The network was trained using 100 000 traces with random keys and once again, we calculated attribution heatmaps using a subset of 1000 measurements. Since the DeepExplain framework does not support occlusion analysis for DNNs having multiple inputs, we only report results for saliency and LRP. In the last row of Fig. 1, it can be seen that both methods consider only a small fraction of sample points as most important. Same applies to correlation analysis. However, the remaining part of the correlation curve looks very noisy compared to the attribution-based methods. When examining the pseudo code of the implementation that is given in [7], it becomes evident that the high peak in the plots represent the time window when the key is masked before the actual AES round transformation. The second smaller peak, which appears a bit later in the saliency heatmap, is likely due to the XOR of the plaintext with the masked key. In RSM, the mask values are fixed to carefully chosen constants which are rotated for every cipher execution. The results show that such a construction is not secure enough to resist DNN-based attacks. That is why we recommend to employ masking schemes that provide a higher level of entropy.

DNN attribution mechanisms are especially interesting in combination with the DK approach, since here no specific assumption about the leakage behavior of the implementation under test is assumed. This means, an evaluator using the method out-of-the-box is only able to validate if the implementation is vulnerable to such kinds of attacks. Attribution-based leakage analysis supports this process by identifying which parts of the implementation need to be fixed in order to increase the SCA resistance.

4 Evaluating Side-Channel Heatmaps

As discussed in the previous section, side-channel heatmaps of the same data set can vary a lot depending on the used attribution method. A natural question is therefore which technique for computing DNN attributions is most suitable in

the context of SCA for leakage analysis. In image classification tasks, heatmaps are often evaluated qualitatively by human experts. This is supported through highlighting the important pixels in the ground truth. It is trivial to see that this process cannot be applied for side-channel traces, as it is not always possible to judge whether a 1D heatmap indicates the 'important' sample points only by visual inspection. Because of that, we introduce two novel quantitative metrics in the following to assess the quality of side-channel heatmaps.

Given an attribution heatmap $\bar{\mathbf{r}}$, we can derive an ordered sequence $\mathbf{s} \in \mathbb{N}^D = [s_1, \ldots, s_D]$ that sorts the values of $\bar{\mathbf{r}}$ according to its relevance such that the property holds:

$$(i < j) \Leftrightarrow (|\bar{r}_i| > |\bar{r}_j|) \tag{7}$$

That means, values at the beginning of $\mathbf{s}$ indicate the sample points with the highest relevance, while values at end of the vector can be considered of less importance. We use absolute values for the comparison since a side-channel heatmap can also contain negative attribution values as illustrated in Fig. 1. However, the sign of the attribution can be disregarded in this case since both positive as well as negative evidence can be considered as important for POI detection. Based on the ordering $\mathbf{s}$, we can define our heatmap metrics called Key Rank Perturbation Curve (KRPC) and Zero-Baseline Key Guessing Entropy (ZB-KGE).

4.1 KRPC

The KRPC is inspired by the region perturbation method proposed in [26] and measures how the key rank calculated in the recovery phase of a profiled attack increases when we progressively replace sample points in the traces with Gaussian noise. Algorithm 1 summarizes the procedure to compute the KRPC.

Algorithm 1. KRPC

Inputs: Sorted heatmap indices $\mathbf{s}$, attack (sub-)set $\mathcal{D}_{Attack}$, correct key k, trained DNN f, number of perturbation (replacement) steps N_{Pert}

1: Initialize perturbation counter: $i = 1$
2: **while** $i < N_{Pert}$ **do**
3: Get index of sample points to perturb: $ip = \mathbf{s}[i]$
4: **for all** $\mathbf{x} \in \mathcal{D}_{Attack}$ **do**
5: Replace sample point with Gaussian noise: $\mathbf{x}[ip] = \mathcal{N}(0, 1)$
6: **end for**
7: Calculate key rank with updated traces: $\mathbf{kr}[i] = \mathbf{d}^{LL}(\mathcal{D}_{Attack}, f)[k]$
8: Increase perturbation counter: $i = i + 1$
9: **end while**

return: Key rank vector $\mathbf{kr}$ for key k

We have decided to use a Gaussian noise with mean $\mu = 0$ and standard deviation $\sigma = 1$ (denoted as $\mathcal{N}(0, 1)$ in Algorithm 1) as perturbation procedure,

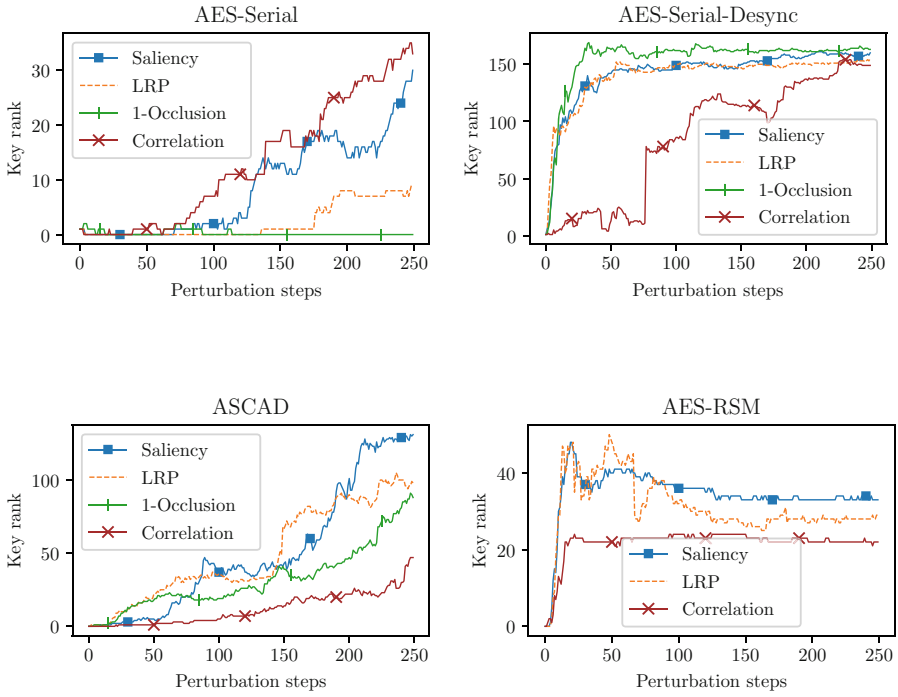

Fig. 2. Mean KRPC curves of four data sets

since the injected values lie within the same distribution as our standardized inputs. The KRPC can be interpreted as noise that is present in the attack traces, but not in the training traces. Replacing the most sensitive samples first (i.e. the sample points containing the most information regarding the classification) should imply a fast decrease of the key rank.

We have computed KRPC curves for all four data sets with $N_{Pert} = 250$ perturbation steps. This number was chosen since it represents at least 25% of the sample points in three of the four data sets. For computational reasons, and since we did not see a substantial change in the KRPC curves set when applying more than 250 perturbation steps, we used the same value of N_{Pert} for the AES-RSM data set. Additionally, we have restricted the number of attack traces that are used in Algorithm 1 to a value that led to a stable key rank below three without perturbation (The exact numbers are given in Table 1). In order to decrease the bias that is induced by a fix choice of the attack traces, we have repeated each experiment five times and used a different subset of the attack traces for every run. Finally, we calculated average KRPC curves which are illustrated as a function of perturbation steps in Fig. 2.

From there, we can generally observe that the correlation analysis reaches the highest key rank after perturbing 250 sample points in the unprotected hardware setting (AES-Serial), whereas the attribution methods perform better in

case of desynchronized traces (AES-Serial-Desync) and the protected software implementations (ASCAD, AES-RSM). Looking at the three attribution methods exclusively, one can see that there is no clear winner over all data sets. For example, the heatmaps computed by LRP and 1-occlusion better identifies the most relevant POIs in the ASCAD data set in the beginning, while saliency performs far better than the two other techniques on the unprotected hardware implementation. We assume that this is due to the fact that saliency is only able to provide local explanations and thus is less suitable for POI detection in settings with highly multivariate leakage (i.e. implementations with masking countermeasures). The unprotected hardware implementation, in contrast, exhibits several independent leakage locations due to its serial architecture, which can be better detected by the saliency method.

Comparing the plots of AES-Serial with AES-Serial-Desync, it is notable that the results for the jitter-based setting are almost similar, while there are significant differences between the attribution methods for the aligned traces. It seems furthermore that the injected jitter is beneficial for identifying the most relevant areas in the traces as the corresponding curves rise stronger. However, we cannot exclude that this is solely due to the misalignment in the traces forcing the DNN to learn spatially invariant features, or just the fact that we had to use more traces in the training phase.

Results for AES-RSM again look very similar for Saliency and LRP, which is not surprising when looking at the corresponding attributions in Fig. 1. Although AES-RSM is also equipped with a lightweight masking countermeasure, the exploited leakage is rather of univariate nature since there is only a single peak visible in the heatmaps that is detected by saliency and LRP likewise. Same applies to a certain degree also for correlation, but the corresponding KRPC curve runs a bit flatter than the previous two mentioned.

4.2 ZB-KGE

Using ZB-KGE, we are able to determine how fast the key rank estimated with a zero-baseline attack set $\mathcal{D}_{Baseline}$ (i.e. an attack set where all sample points in the traces are set to zero) converges when we continuously add relevant sample points from the actual attack set $\mathcal{D}_{Attack}$ to $\mathcal{D}_{Baseline}$. The procedure for calculating a ZB-KGE curve is described in Algorithm 2. Intuitively, the steeper a ZB-KGE graph decreases, the more POIs have been identified by the related side-channel heatmap. Since the ZB-KGE simulates the absence of features, it furthermore provides insights on how many POIs should approximately be conserved in case of a dimensionality reduction.

Figure 3 displays the ZB-KGE as function of the number of added POIs for the four data sets. As in the previous experiment, we have calculated mean curves over five independent subsets of $\mathcal{D}_{Attack}$. From Fig. 3, it can be noticed that the results are close to, but not equivalent to those computed with Algorithm 1. For instance, correlation analysis again performs best on the AES-Serial dataset but is defeated by the attention methods for the remaining three data sets. What we find interesting is the fact that 1-occlusion identifies almost equally good relevant

sample points as LRP and Saliency in the AES-Serial-Desync and ASCAD data sets, but a bit worse in the AES-Serial data set. This is an indicator that the information contained in a single sample point of the unprotected hardware traces is rather small. A greater occlusion factor might be more suitable in such cases where the univariate leakage is distributed over a large range of connected sample points. Furthermore, it can be noticed, that the random shifting of the data points in the AES-Serial-Desync data set has a very positive effect on 1-occlusion. Intuitively, the random shifting induces a varying occlusion factor which seems to be beneficial taking the results of the two previous sections into account. This makes the technique interesting for different setups where traces are not perfectly aligned, be it due to an unstable measurement setup, or because of some delay-based countermeasure (e.g. [12]).

The attribution curves for the AES-RSM data set are again very similar and show that roughly 150 out of 10 000 data points are sufficient to reveal the correct key. In contrast, the correlation curve decreases very slowly for this data set. We assume this is due to the noisy result of the correlation analysis shown above.

Algorithm 2. ZB-KGE

Inputs: Sorted heatmap indices $\mathbf{s}$, attack (sub-)set $\mathcal{D}_{Attack}$, correct key k, trained DNN f, number of sample points to add N_{Add}

Initialize status counter: $i = 1$
Initialize zero-baseline attack set: $\mathcal{D}_{Baseline}$
while $i < N_{Add}$ **do**
  Get index of sample points to add: $ia = \mathbf{s}[i]$
  for all $\mathbf{x}^A \in \mathcal{D}_{Attack}, \mathbf{x}^B \in \mathcal{D}_{Baseline}$ **do**
    Replace zero sample point with actual value: $\mathbf{x}^B[ia] = \mathbf{x}^A[ia]$
  end for
  Calculate key rank with updated traces: $\mathbf{kr}[i] = \mathbf{d}^{LL}(\mathcal{D}_{Baseline}, f)[k]$
  Increase status counter: $i = i + 1$
end while

return: Key rank vector $\mathbf{kr}$ for key k

4.3 Limitations

We have seen that DNN-based attribution methods are superior to classical correlation analysis (except for the unprotected data set). However, there is a necessary precondition one have to consider: Meaningful POIs can only be extracted from the network when the training was successful, i.e., when the network was able to learn the target function f of the training set and generalize to new data. Since we were able to drive a successful key recovery on all data sets, we were certain about the network's performance. However, performing key recovery is not necessary to evaluate the training procedure. In order to monitor the learning

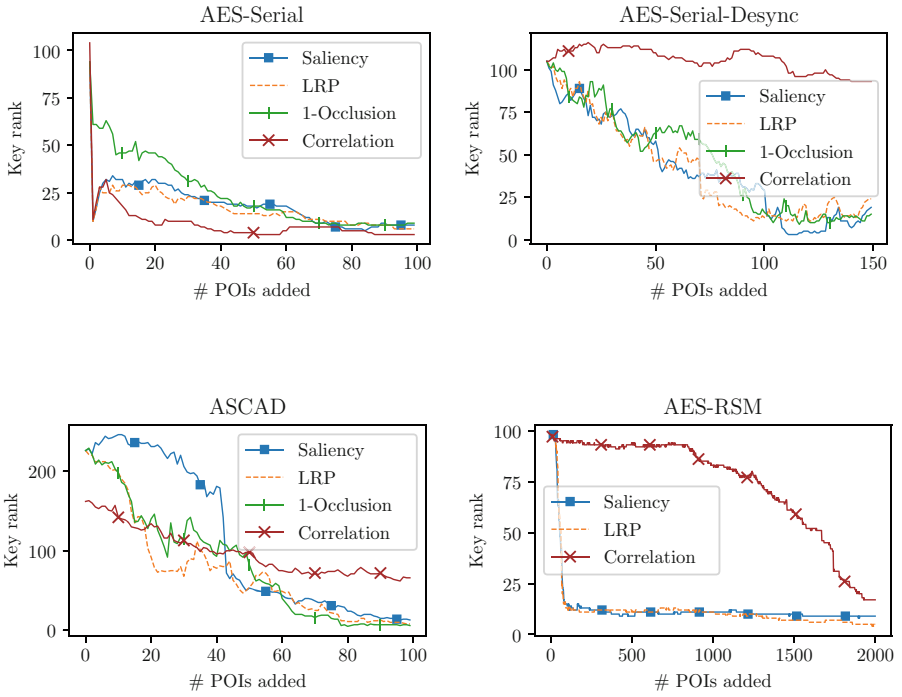

Fig. 3. Mean ZB-KGE curves for the four data sets.

progress, usually a small amount of the training set is used to verify the model's performance on unseen data (i.e. validation set) at the end of each epoch (i.e. an iteration over the complete training dataset [15]). An increasing training and validation accuracy along with a decreasing training and validation loss indicates that the DNN model is able to approximate f. If this is not case, the network either underfits the data (meaning model is not able to obtain a sufficiently low error on training and validation set), or runs into overfitting (model performs well on the training set, but not on the validation set). Overfitting and underfitting is primarily influenced by the number of parameters of the model, which reveals another issue of DNN-based POI visualization compared to parameterless methods like correlation or DOM: A suitable network architecture has to be determined which encompasses, amongst others, the number of layers, the number of neurons per layer, etc. For this paper, we derived our final network mainly from related work [17,25]. However, there are several methods available to automatically search for a suitable set of parameters like genetic algorithms or Bayesian optimization [13]. This facilitates also non-experts in the field an easy access to DNN-based methods.

5 Attribution as a Distinguisher

As explained earlier in the paper, LRP provides signed explanations that allow to distinguish between input features that support the classification decision, and features speaking against the prediction result. This property is very helpful in image classification tasks as LRP heatmaps can be easily interpreted, e.g., to debug which pixels of an image led to a misclassification. In this section, we exploit the ability of LRP to provide negative and positive evidence to distinguish between correct and incorrect key hypothesis in the key recovery phase of a profiled attack. The basic idea is that there should be a measurable difference between heatmaps calculated with the attack traces under the correct key guess, and heatmaps for which the wrong output neuron of the DNN (i.e. label) has been chosen. Furthermore, the difference should be most distinct in areas which have been identified as relevant during profiling. The procedure of the complete attack is as follows:

1. Perform DNN training as in a usual profiled attack according to Sect. 2.1 in order to build device model f.
2. Create side-channel heatmap $\bar{\mathbf{r}}$ using (6) and a subset of $\mathcal{D}_{Train}$. Next, build ordered sequence $\mathbf{s}$ that fulfills (7).
3. For each key hypothesis $k^* \in \mathcal{K}$, calculate attribution vector $\mathbf{r}^{C_{k^*}}$ using LRP and sum up those values that correspond to the N_{POI} highest ranked components in $\mathbf{s}$. Repeat for complete attack set $\mathcal{D}_{Attack}$ composed of N_A attack traces such that:

$$\mathbf{d}^{AT}(\mathcal{D}_{Attack}, f) = \sum_{i=1}^{N_A} \sum_{j=1}^{N_{POI}} \mathbf{r}^{C_{k^*}}(\mathbf{x}_i, f)[\mathbf{s}[j]] \tag{8}$$

   The attack is successful if $k = \arg\max(\mathbf{d}^{AT})$

We have performed the attack on the AES-Serial and ASCAD data sets with $N_{POI} = 50$ (The number of relevant POIs was roughly estimated by inspecting the corresponding plots in Fig. 1). The remaining parameters as well as the DNN architecture have been the same as in the previous experiments. Figure 4 shows the evolution of the average key rank as a function of the number of attack traces computed from ten independent attacks (using 1000 traces per attack). For comparison, we have done the key recovery also according to (1) using the same DNN models and exactly the same attack traces. From Fig. 4, one can see that our proposed attribution-based attack converges faster to a key rank of one than the LL-based attack in both data sets. More concretely, for the unprotected hardware AES, our method needs less than ten traces to enter a key rank below five and stabilizes after roughly 50 attack traces. The LL-based attack, in contrast, reaches a stable key rank of one only after 750 traces. Results for the protected software AES differ not to such an extent. However, the attribution-based attack manages a stable key rank of one using approximately 85 traces while the attack based on LL distinguisher needs around 500 traces more to pass that mark.

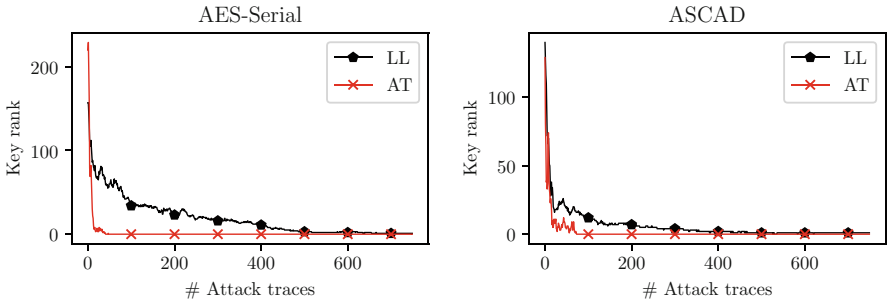

Fig. 4. Mean key ranks for the unprotected hardware AES (left) and the protected software AES (right). The attribution-based attack (AT) needs less traces for a successful key recovery than the LL-based attack in both setups.

In summary, our experiments demonstrate that attribution methods and especially LRP are able to use the information that is captured during DNN training more efficiently for key recovery than the standard LL approach. This is because to the fact that the attribution-based attack considers only the crucial parts in the traces during the attack phase, while complete traces are used to calculate the LL scores. Drawback of the method is the increased time complexity due to the need of computing attributions over all key hypothesis. However, we stress that time is often not a limiting factor for an adversary. We were able to do a successful key recovery on a single Nvidia GeForce GTX 1080 GPU in under 5 min (compared to approximately 15 seconds using the LL approach) which is still practical. This further confirms that our attribution-based distinguisher is a promising alternative when performing profiled SCA especially in settings where the adversary is able to acquire only a limited number of attack traces.

6 Conclusion

In this work we have studied DNN attribution methods as a tool for leakage analysis in DL-based side-channel attacks. In particular, we have presented a technique to compute heatmaps of side-channel traces in order to find leaking operations in unprotected and protected cryptographic implementations. We proposed two metrics to evaluate the quality of side-channel heatmaps and used them to assess saliency analysis, LRP, and 1-occlusion for their suitability to detect sensitive sample points in side-channel traces. Furthermore, we have compared the three methods with the widely-used Pearson correlation for POI analysis. As a summary, we can conclude that the attribution methods are beneficial especially with regard to secured implementations and in case of desynchronized traces. For standard unprotected settings, there seems to be no advantage over standard Pearson correlation especially when taking in consideration that training a DNN is much more time and computation intensive than a standard CPA. However, as also demonstrated in the paper, the LRP attribution method can also be used to build an effective distinguisher for key recovery.

Future work might investigate other DNN attribution methods in the context of SCA, such as prediction difference analysis [34] or Deconvolution [32]. Another interesting path could be to explore the usage of DNN visualization techniques for network debugging and architecture optimization.

A Network Parameters

Table 2. Network configuration of CNN

Layer Type	Hyperparameters
Trace input	-
Convolution 1D	filters = 8, filter length = 8, activation = ReLU
Max-pooling	pool length = 2
Dropout	$P_{Drop} = 0.3$
Convolution 1D	filters = 16, filter length = 8, activation = ReLU
Batch normalization	-
Max-pooling	pool length = 2
Dropout	$P_{Drop} = 0.3$
Convolution 1D	filters = 32, filter length = 8, activation = ReLU
Batch normalization	-
Max-pooling	pool length = 2
Dropout	$P_{Drop} = 0.3$
Convolution 1D	filters = 64, filter length = 8, activation = ReLU
Batch normalization	-
Max-Pooling	pool length = 2
Dropout	$P_{Drop} = 0.3$
Flatten	-
(optional) Domain input	neurons = 256
(optional) Concatenate	-
Fully-connected	neurons = 20, activation = ReLU
Batch normalization	-
Dropout	$P_{Drop} = 0.2$
Output	neurons = 256

In all experiments, we trained the network using Adam optimizer and a learning rate of 0.0001 (AES-Serial & AES-Serial-Desync) or 0.001 (ASCAD & AES-RSM).

References

1. DeepExplain: attribution methods for Deep Learning. https://github.com/marcoancona/DeepExplain
2. Keras Documentation. https://keras.io/
3. Agrawal, D., Archambeault, B., Rao, J.R., Rohatgi, P.: The EM side—channel(s). In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 29–45. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_4
4. Ancona, M., Ceolini, E., Öztireli, C., Gross, M.: Towards better understanding of gradient-based attribution methods for Deep Neural Networks. ArXiv e-prints, November 2017
5. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template attacks in principal subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_1
6. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE **10**, 1–46 (2015)
7. Bhasin, S., Bruneau, N., Danger, J.-L., Guilley, S., Najm, Z.: Analysis and improvements of the DPA contest v4 implementation. In: Chakraborty, R.S., Matyas, V., Schaumont, P. (eds.) SPACE 2014. LNCS, vol. 8804, pp. 201–218. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12060-7_14
8. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) CHES 2017. LNCS, vol. 10529, pp. 45–68. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66787-4_3
9. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
10. Ching, T., et al.: Opportunities and obstacles for deep learning in biology and medicine. J. R. Soc. Interface **15**(141), 20170387 (2018). https://doi.org/10.1098/rsif.2017.0387
11. Cooper, J., Goodwill, G., Jaffe, J., Kenworthy, G., Rohatgi, P.: Test vector leakage assessment (TVLA) methodology in practice. In: International Cryptographic Module Conference (ICMC). Holiday Inn Gaithersburg, Gaithersburg (2013)
12. Coron, J.S., Kizhvatov, I.: An efficient method for random delay generation in embedded software. Cryptology ePrint Archive, Report 2009/419 (2009). https://eprint.iacr.org/2009/419
13. Elsken, T., Hendrik Metzen, J., Hutter, F.: Neural Architecture Search: A Survey. arXiv e-prints arXiv:1808.05377, August 2018
14. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006). https://doi.org/10.1007/11894063_2
15. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016). http://www.deeplearningbook.org
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016

17. Hettwer, B., Gehrer, S., Güneysu, T.: Profiled power analysis attacks using convolutional neural networks with domain knowledge. In: Selected Areas in Cryptography - SAC 2018–25th International Conference, Calgary, AB, Canada, 15–17 August 2018, Revised Selected Papers, pp. 479–498 (2018). https://doi.org/10.1007/978-3-030-10970-7_22
18. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
19. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9
20. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) SPACE 2016. LNCS, vol. 10076, pp. 3–26. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_1
21. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards, 1st edn. Springer, Boston (2010). https://doi.org/10.1007/978-0-387-38162-6
22. Masure, L., Dumas, C., Prouff, E.: Gradient visualization for general characterization in profiling attacks. In: Polian, I., Stöttinger, M. (eds.) COSADE 2019. LNCS, vol. 11421, pp. 145–167. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16350-1_9
23. Moradi, A., Guilley, S., Heuser, A.: Detecting hidden leakages. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 324–342. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07536-5_20
24. Picek, S., Heuser, A., Jovic, A., Batina, L., Legay, A.: The secrets of profiling for side-channel analysis: feature selection matters. Cryptology ePrint Archive, Report 2017/1110 (2017). https://eprint.iacr.org/2017/1110
25. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Dumas, C.: Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. Cryptology ePrint Archive, Report 2018/053 (2018). https://eprint.iacr.org/2018/053
26. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.: Evaluating the visualization of what a deep neural network has learned. IEEE Trans. Neural Networks Learn. Syst. **28**(11), 2660–2673 (2017). https://doi.org/10.1109/TNNLS.2016.2599820
27. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 618–626, October 2017. https://doi.org/10.1109/ICCV.2017.74
28. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs], December 2013
29. Standaert, F.-X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_26
30. Timon, B.: Non-profiled deep learning-based side-channel attacks. Cryptology ePrint Archive, Report 2018/196 (2018). https://eprint.iacr.org/2018/196
31. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing [review article]. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018). https://doi.org/10.1109/MCI.2018.2840738

32. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR abs/1311.2901 (2013). http://arxiv.org/abs/1311.2901
33. Zheng, Y., Zhou, Y., Yu, Z., Hu, C., Zhang, H.: How to compare selections of points of interest for side-channel distinguishers in practice? In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) ICICS 2014. LNCS, vol. 8958, pp. 200–214. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21966-0_15
34. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. arXiv:1702.04595 [cs], February 2017