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, 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 try to close this gap by applying attribution methods that aim for interpreting 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 Interests (POIs) in side-channel traces. We show by experiments with three separate data sets that Layer-wise Relevance Propagation (LRP) proposed by Bach et al. provides the best result in most cases. Finally, we demonstrate that attribution can also serve as a powerful side-channel distinguisher 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 [13], power consumption [12], and Electromagnetic (EM) emanation [3] have been investigated to reveal secret parameters. In order to decrease the information leakage of cryptographic implementations, researchers and industry came up with dedicated countermeasures which can be roughly classified into *Masking* and *Hiding* [15]. However, more powerful attacks demonstrated that even side-channel protected implementations may still be vulnerable [16].

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 [10], natural language processing [23], and medicine [8]. In the context of SCA, especially Convolutional Neural Networks (CNNs) have shown to be advantageous over standard analyzing tools like Template Attacks (TAs) in different settings (for example in case of de-synchronized traces or an unknown leakage model) [7,11,14,22].

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 a 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 specific, we investigate *saliency maps* [20], *occlusion* [24], and *LRP* [5] 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 pre-processing step ahead of the actual attack [17]. Here, we take another perspective and show a technique to compute the relevance of sample points in side-channel traces after the profiling step. This can be seen as a constructive method for evaluators to identify the operations of the implementation under test which caused the highest leakage. Furthermore, we demonstrate that attribution methods can also be used as a distinguisher in DNNs-based SCA.

1.1 Contribution & Structure of the Paper

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 to quantitatively asses how good the selection of POIs is done.
3. We compare three attribution methods on three different data sets: an unprotected hardware and two protected software implementations of the Advanced Encryption Standard (AES). Our results indicate that the LRP approach is most suitable for finding POIs.

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.

The structure of the paper is as following: In Section 2, we shortly recap DL-based SCA and give an introduction in DNN attribution methods. In Section 3, we presents our approach for POI visualization and apply them to three data sets for leakage analysis. In Section 4, we evaluate the quality of side-channel heatmaps. In Section 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 that 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:

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

The k-th entry in score vector $\mathbf{d}^{LL}$ corresponds to the correct key candidate [18]. 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 $\mathcal{D}_{Attack}$ [21]. It is computed by performing a ranking of $\mathbf{d}$ after the evaluation of each attack trace.

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 as to 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 a 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 [9].

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.

Saliency Maps Saliency maps were introduced by Simonyan et al. in 2013 to highlight class discriminative areas of images [20] captured by CNNs. 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 [19].

Layer-wise Relevance Propagation (LRP) 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 terms of the separate input dimensions [5]:

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

, where $r_i > 0$ can be interpreted as positive evidence for the presence of a structure, 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 redistribution 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.

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 [19]. We will see later in the paper that this property can be exploited to construct a side-channel distinguisher.

Occlusion 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 [24]. 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 is 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 those 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 [17].

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 three 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 contains the most information [7,14]. 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 the POIs has to be done manually as 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 [25].

In this section, we go one step further and describe a way to extract the POIs from a trained 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 Section 2. C_k represents the output class under the correct key hypothesis, 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 three data sets for the experiments of the paper: An unprotected hardware (denoted as AES-Serial), and two protected software implementations of the AES (denoted as ASCAD and AES-RSM). For all data sets, we have created attribution heatmaps according to 6 using the Python frameworks Keras [2]

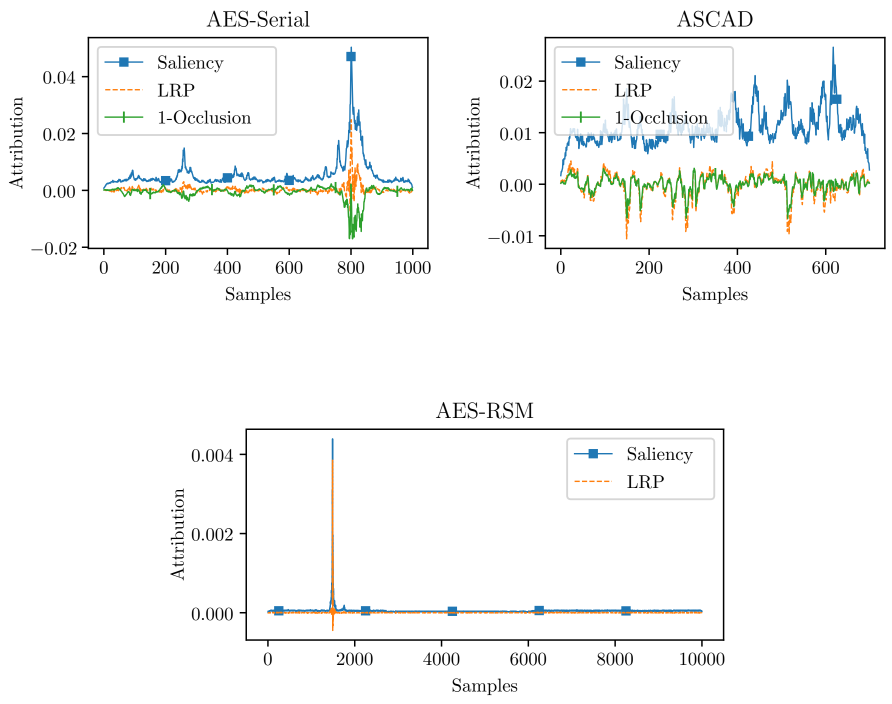

Fig. 1: Mean attributions for three different data sets. Each curve has been calculated with $N_{Attr} = 1000$ traces according to formula 6.

and DeepExplain [1]. We have kept the same DNN architecture throughout all experiments in order to allow an unbiased evaluation. The employed network is a CNN which consist of four convolution blocks followed by two fully-connected layers. Details about the network structure along with related training parameters are described in detail in Table 1 in the Appendix. As a preprocessing step, we transformed the traces of all data sets to have zero mean and unit variance (sometimes referred to as data standardization).

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 Section 2. The result is plotted in the upper left corner 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. Interestingly, the saliency heatmap indicates a wider range of samples as important and additionally shows a second peak in the first half of heatmap. One can now use this information in order to increase the SCA resistance of the implementation. Indeed, by examining the implemented hardware layout, we could easily backtrack that the found leakage is caused by an unintended high routing fanout between the state array and the mix columns operation.

ASCAD ASCAD is public database of side-channel measurements and related meta-data obtained from a first-order secured software AES implementation [18]. 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 upper right corner of 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. Considering the Signal-to-Noise Ratio (SNR) analysis that is given in [18], 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 heatmap. The example demonstrates that attribution methods may also help to reverse-engineer internal structures of protected cryptographic implementations.

AES-RSM The third data set we have analyzed is based on a secured software AES implementation which originates from the DPA Contest v4.2 [6]. 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 and the related plaintext byte is given to the network via a second input path [11]. 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 lower part of Fig.1 one can see that both methods consider only a small fraction of sample points as important. When examining the pseudo code of the implementation that is given in [6], it becomes evident that these sample points 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 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 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 possible to judge whether a 1D heatmap indicates the ”important” sample points 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:

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

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 [19] and measures how the key rank calculated in the recovery phase of a profiled attack

Algorithm 1 KRPC

```
Inputs: Sorted heatmap indices $\mathbf{s}$, attack (sub-)set $\mathcal{D}_{Attack}$, trained DNN $f()$, number
    of perturbation steps $N_{Pert}$
 1: Initialize perturbation counter: $i = 0$
 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 $k$
```

increases when we progressively inject Gaussian noise into the traces. Algorithm 1 summarizes the procedure to compute the KRPC.

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, since the injected values lie within the same distribution as the other sample points in the trace (induced by our preprocessing). 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 should imply a fast decrease of the key rank.

We have computed KRPC curves for all three data sets with $N_{Pert} = 250$ perturbation steps. For computational reasons, 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. However, 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 function of perturbation steps in Fig.2. From there, we can observe that saliency analysis reaches the highest key rank after perturbing 250 sample points in all investigated data sets. However, there are notable differences between the three attribution methods when examining only the first 100 perturbation steps. One can see, for example, that the heatmaps computed by LRP and 1-occlusion better identifies the most relevant POIs in the ASCAD data set, while saliency performs best 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. Results for AES-RSM are almost similar, 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

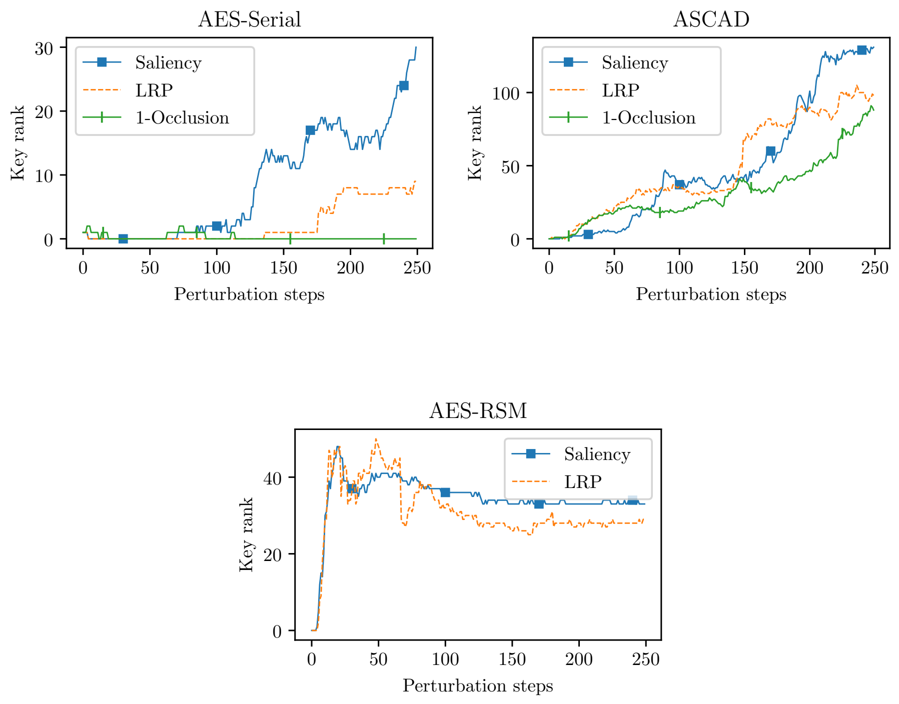

Fig. 2: Mean KRPC curves for three data sets

univariate nature since there is only a single peak visible in both heatmaps that is detected by saliency and LRP likewise.

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. Fig.3 displays the ZB-KGE as function of the number of added POIs for the three 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, the LRP and 1-occlusion ZB-KGE curves for the ASCAD data set drop faster in the beginning, but eventually gets passed by the curve for the saliency method. What we find interesting is the fact that 1-occlusion identifies equally good relevant sample points in the ASCAD data set as LRP, but not in the AES-Serial data set. This is a strong 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. The 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.

Algorithm 2 ZB-KGE

```
Inputs: Sorted heatmap indices $\mathbf{s}$, attack (sub-)set $\mathcal{D}_{Attack}$, trained DNN $f()$, number
    of sample points to add $N_{Add}$
 1: Initialize status counter: $i = 0$
 2: Initialize zero-baseline attack set: $\mathcal{D}_{Baseline}$
 3: while $i < N_{Add}$ do
 4:     Get index of sample points to add: $ia = \mathbf{s}[i]$
 5:     for all $\mathbf{x}^A \in \mathcal{D}_{Attack}, \mathbf{x}^B \in \mathcal{D}_{Baseline}$ do
 6:         Replace zero sample point with actual value: $\mathbf{x}^B[ia] = \mathbf{x}^A[ia]$
 7:     end for
 8:     Calculate key rank with updated traces: $\mathbf{kr}[i] = \mathbf{d}^{LL}(\mathcal{D}_{Baseline}, f())[k]$
 9:     Increase status counter: $i = i + 1$
10: end while
return: Key rank vector $\mathbf{kr}$ for key $k$
```

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 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 following:

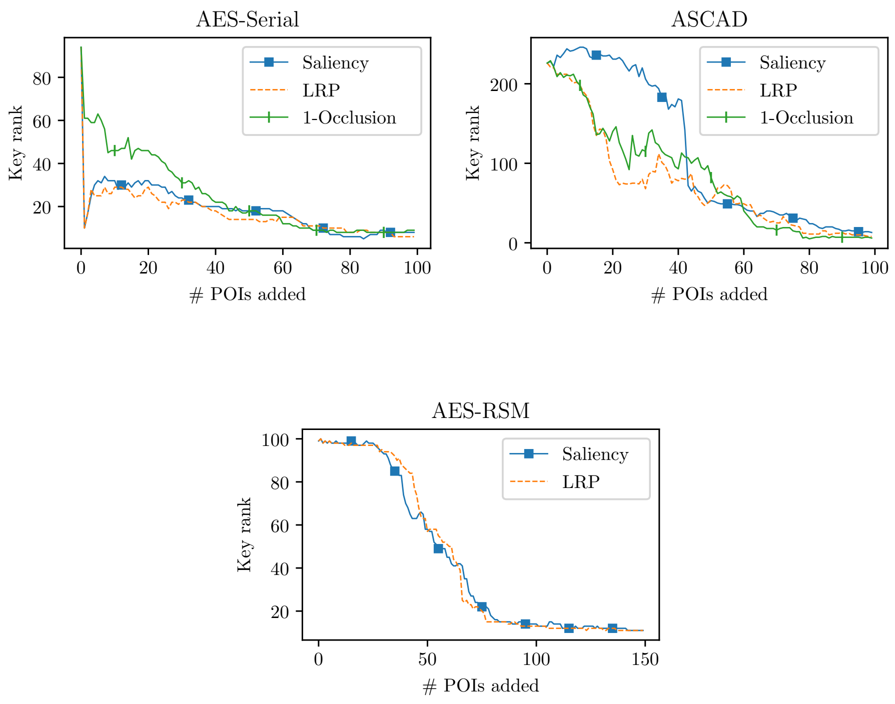

Fig. 3: Mean ZB-KGE curves for three data sets

1. Perform DNN training as in a usual profiled attack according to Section 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 $\mathcal{D}_{Attack}$ 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 remaining parameters as well as the DNN architecture have been the same as in the previous experiments. Fig.4 shows the evolution of the average key rank as function of the number of attack traces computed from five independent attacks. For comparison, we have done the key recovery also

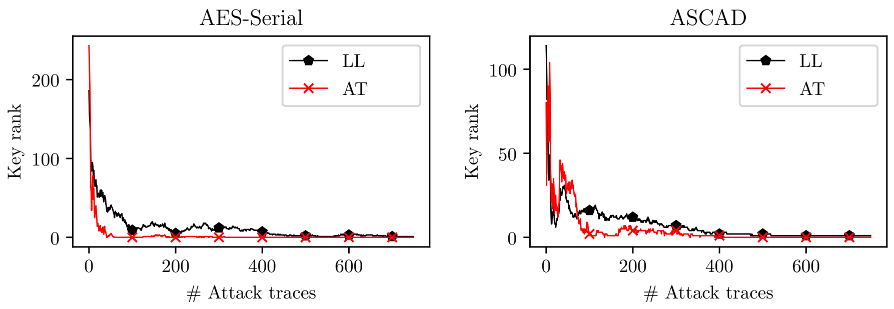

Fig. 4: Mean key ranks for the unprotected hardware AES (left) and the protected software AES (right). The attribution-based attack (AT) needs less traces for a successful key recovery than the LL-based attack in both setups.

according to 1 using same trained DNN models and exactly the same 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 roughly 100 traces to get to key rank one for the first time and stabilizes after 300 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 below ten with approximately 100 traces while the attack based on LL distinguisher needs around 300 traces more to pass that mark.

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 recovery than the standard LL approach. 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 10 minutes which is still practical. This further confirm that our attribution-based distinguisher is an interesting alternative when performing profiled SCA.

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 additionally proposed two metrics to evaluate the quality of side-channel heatmaps and used them to compare saliency analysis, LRP and 1-occlusion for their suitability to detect sensitive sample points in side-channel traces. Although the results indicate no clear advantage for one of the methods, we tend to recommend LRP as most appropriate due to its good performance on the first-order secured implementations. Furthermore, as demonstrated in the paper, there is also the opportunity to build an effective distinguisher for key recovery using LRP.

Future work might investigate other DNN attribution methods in the context of SCA, such as prediction difference analysis [26] or Deconvolution [24]. Another interesting path could be to explore the usage of DNN visualization techniques for network debugging and architecture optimization.

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 SideChannel(s). In: Cryptographic Hardware and Embedded Systems - CHES 2002, pp. 29–45. Springer, Berlin, Heidelberg (Aug 2002), dOI: 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 (Nov 2017)
5. Bach, S., Binder, A., Montavon, G., Klauschen, F., Mller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLOS ONE **10**, 1–46 (07 2015). https://doi.org/10.1371/journal.pone.0130140, https://doi.org/10.1371/journal.pone.0130140
6. 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.) Security, Privacy, and Applied Cryptography Engineering. pp. 201–218. Springer International Publishing, Cham (2014)
7. Cagli, E., Dumas, C., Prouff, E.: Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures, pp. 45–68. Springer International Publishing, Cham (2017)
8. Ching, T., Himmelstein, D.S., Beaulieu-Jones, B.K., Kalinin, A.A., Do, B.T., Way, G.P., Ferrero, E., Agapow, P.M., Zietz, M., Hoffman, M.M., Xie, W., Rosen, G.L., Lengerich, B.J., Israeli, J., Lanchantin, J., Woloszynek, S., Carpenter, A.E., Shrikumar, A., Xu, J., Cofer, E.M., Lavender, C.A., Turaga, S.C., Alexandari, A.M., Lu, Z., Harris, D.J., DeCaprio, D., Qi, Y., Kundaje, A., Peng, Y., Wiley, L.K., Segler, M.H.S., Boca, S.M., Swamidass, S.J., Huang, A., Gitter, A., Greene, C.S.: Opportunities and obstacles for deep learning in biology and medicine. Journal of The Royal Society Interface **15**(141) (2018). https://doi.org/10.1098/rsif.2017.0387
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
10. 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)

11. 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, August 15-17, 2018, Revised Selected Papers. pp. 479–498 (2018). https://doi.org/10.1007/978-3-030-10970-7_22, https://doi.org/10.1007/978-3-030-10970-7_22
12. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis, pp. 388–397. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
13. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. pp. 104–113. CRYPTO '96, Springer-Verlag, London, UK, UK (1996), http://dl.acm.org/citation.cfm?id=646761.706156
14. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking Cryptographic Implementations Using Deep Learning Techniques, pp. 3–26. Springer International Publishing, Cham (2016)
15. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer Publishing Company, Incorporated, 1st edn. (2010)
16. Moradi, A., Guilley, S., Heuser, A.: Detecting hidden leakages. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security. pp. 324–342. Springer International Publishing, Cham (2014)
17. 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
18. 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
19. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.: Evaluating the visualization of what a deep neural network has learned. IEEE Transactions on Neural Networks and Learning Systems **28**(11), 2660–2673 (Nov 2017). https://doi.org/10.1109/TNNLS.2016.2599820
20. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. arXiv:1312.6034 [cs] (Dec 2013), http://arxiv.org/abs/1312.6034, arXiv: 1312.6034
21. Standaert, F.X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
22. Timon, B.: Non-profiled deep learning-based side-channel attacks. Cryptology ePrint Archive, Report 2018/196 (2018), https://eprint.iacr.org/2018/196
23. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing [review article]. IEEE Computational Intelligence Magazine **13**(3), 55–75 (Aug 2018). https://doi.org/10.1109/MCI.2018.2840738
24. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. CoRR **abs/1311.2901** (2013), http://arxiv.org/abs/1311.2901
25. 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.) Information and Communications Security. pp. 200–214. Springer International Publishing, Cham (2015)
26. Zintgraf, L.M., Cohen, T.S., Adel, T., Welling, M.: Visualizing Deep Neural Network Decisions: Prediction Difference Analysis. arXiv:1702.04595 [cs] (Feb 2017), http://arxiv.org/abs/1702.04595, arXiv: 1702.04595

A Network Parameters

Table 1: 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) or 0.001 (ASCAD & AES-RSM).