WHITE PAPER

Lowering the Bar: Deep Learning for Side Channel Analysis

Automating the signal analysis process in power side channel analysis

KEYSIGHT

Table of Contents

Abstract

Deep learning can help automate the signal analysis process in power side channel analysis. So far, power side channel analysis relies on the combination of cryptanalytic science and the art of signal processing. Deep learning is essentially a classification algorithm, which can also be trained to recognize different leakages in a chip. Even more so, we do this such that typical signal processing problems such as noise reduction and re-alignment are automatically solved by the deep learning network. We show we can break a lightly protected AES, an AES implementation with masking countermeasures, and a protected ECC implementation. These experiments show that where previously side channel analysis had a large dependency on the skills of the human, first steps are being developed that bring down the attacker skill required for such attacks. This paper is targeted at a technical audience that is interested in the latest developments on the intersection of deep learning, side channel analysis, and security.

Authors: Guilherme Perin, Baris Ege, Jasper van Woudenberg

Introduction

Deep learning is defined as an advanced machine learning technique that is able to learn by example. Among the main applications of deep learning we can refer to self-driving cars, advanced image recognition systems, and natural language processing. The main reason why deep learning is receiving so much attention is mainly attributed to the highly accurate classification results that were not achieved before with alternative machine learning methods.

Even if deep learning is able to provide higher levels of recognition accuracy, its application requires a large amount of labeled data for the learning procedure and substantially more computation power than well-known machine learning techniques. However, current applications benefit from the high availability of big data scenarios and GPUs for parallel computing.

How deep learning works

Behind the term *deep learning* there is another term which is very well known by the scientific community: *artificial neural networks.* A neural network is a structure composed of several layers. At least one input and one output layer are sufficient to define the neural network structure. Usually, classical application of neural networks considers the usage of hidden layers, which are placed between the input and output layers. Traditional structures are composed with 2 or 3 hidden layers. Deep learning, also known as deep neural networks, can go far beyond this value and implement structures with tens to hundreds of hidden layers.

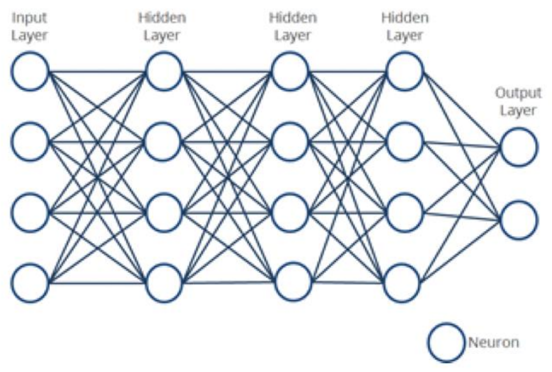

Figure 1. Basic neural network structure

The structure in Figure 1 is an example of a very basic neural network called Fully Connected Network or Multiple Layer Perceptron. It contains only one type of layer, conventionally called a dense or fully connected layer that is composed of several small structures called neurons. Every neuron from a hidden layer is fully connected to all the neurons of its precedent and subsequent layers.

Many different topologies for artificial neural networks exist and, among the most largely used ones, multiple layer perceptron and convolutional neural networks are one of the most popular approaches. In the field of side channel analysis, these two topologies already proved to be successful against first-order masking of AES as well as scenarios where the traces are desynchronized in time. In the last case, convolutional neural networks are particularly successful because they have the ability to identify leakages in different trace positions.

Multiple layer perceptron

Multiple layer perceptron (MLP) is a class of feed forward artificial neural network with a function F that is composed of multiple linear functions and some non-linear activation functions. Every layer of an MLP is composed of multiple neurons (or, as called in the past, perceptron). The neuron is the basic element and it is fully connected to all the neurons of the previous and subsequent layers. Therefore, this topology of neural networks is also known as a fully connected network.

The connection of every neuron to the previous or subsequent layer is defined by a connection weight. Additionally, every neuron has a bias value and an activation function. These are the parameters that are updated during the training of a multiple layer perceptron. The activation function usually is RELU (Rectifier Linear Unit), TANH (hyperbolic tangent) or Sigmoid. The following equation defines the output activation value of a neuron:

$$a = f\left(\sum_{i=1}^{n} \omega_i \, i_i + bias\right)$$

where $f()$ is the activation function, ω_i is the weight connection between the neuron to the neuron i in the previous layer, ii is the activation value of the neuron i in the previous layer, $bias$ is the bias value, and finally n is the number of neurons in the previous layer.

The structure of a multiple layer perceptron must contain at least the following layers:

- Input layer: the number of neurons in the input layer must be given by the number of samples (or points) in the input data;
- Hidden layers: these are the layers placed between the input and the output layers. The hidden layers are responsible for feature extraction and classification of input data;
- Output layer: the output layer maps the information from the last hidden layer to output neurons. The number of neurons in the output layer is equivalent to the number of classes in the input data set. Usually, the output value of a neuron in the output layer is given in terms of probabilities (between 0 and 1). For that, it is recommended the usage of *Softmax* as the activation function combined with the negative log-likelihood for the loss function in the output layer (these hyper-parameters will be explained later in the text).

Convolutional neural networks

Convolutional neural networks (CNN) perform automatic feature extraction from input data. The original idea (Y. Lecun, 1998) appeared as an advanced solution for object recognition in images. However, their application to 1D data (like time-series) is also very appropriate.

The basic structure of a CNN consists of one or more convolution layer, a fully connected network (dense layers), and an output layer. Figure 2 shows an example of a CNN with two convolution layers.

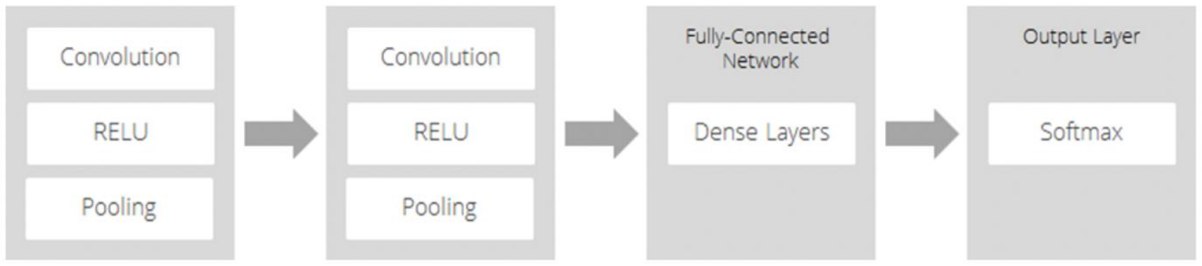

Figure 2. Example of a convolutional neural network structure

The convolution layer contains three basic inner structures: a convolution operation, a RELU (Rectified Linear Unit) as activation function, and a pooling layer. The pooling layer is responsible for down-sampling operation (dimensionality reduction). This convolution layer contains internal weights and biases for its elements. It implements a fixed number of convolution filters, and all of them are defined with same (1D or 2D) kernel size and a stride to define the step of the convolution over the input. The convolution operation (which is the sum of dot products) is performed between the filter and the input data. Therefore, it is also assumed that the output result from a convolution layer is a filtered result or a feature. After the convolution operation, the result is applied to an activation function and, optionally, to a pooling operation.

The convolution filter parameters (weights and bias) are updated for every iteration. The updates are based on the back-propagation algorithm, and it is directly affected by the defined learning rate, regularization, and updater method.

Deep Learning for Side Channel Analysis

The application of deep learning requires a careful analysis of the problem and the configuration of the neural network. Side channel analysis requires a neural network to identify leakages in traces. This leakage can be understood as the dependency between the power consumption (or electromagnetic emanation) and the intermediate states being processed by the cryptographic algorithm. The values of intermediate states depend on the input (plaintext or cipher-text) and the key material. Usually, the amount of leakage embedded in side channel traces is limited by inherent noise from acquisitions.

First, it is important to have in mind what is the entire framework of a deep learning application for side channel attacks. The scenario described here is supervised learning, meaning that an open sample (with a configurable key), or at least a closed sample with a known key, is required for the learning or training phase. After the acquisition phase, the key and inputs are known and are associated to each side channel trace in order to label them. Next, the trace set is split into training and validation sets (see Figure 3). Ideally, a new trace set is measured from another and identical device (closed sample), containing an unknown key. This last trace set is then used as a test set. Because deep learning is very computationally expensive, the training phase used to be very time-consuming, while the validation and test phases are usually fast.

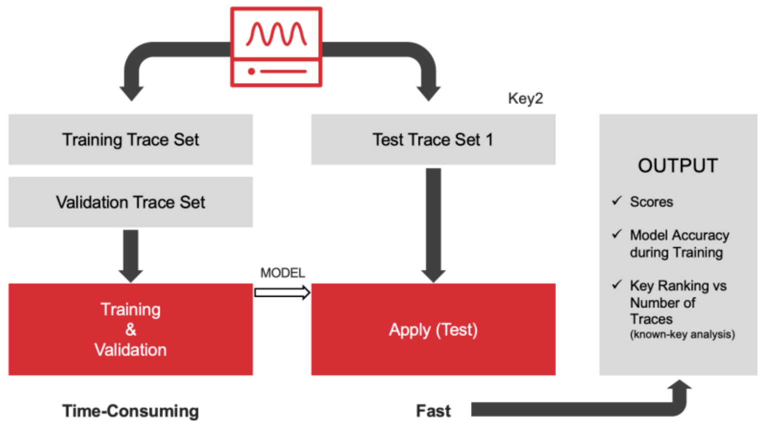

Figure 3. Basic side channel framework for deep learning

As do conventional side channel analysis methods (such as differential power analysis and template attacks), deep learning also requires the definition of a *leakage model*. The application of multiple layer perceptron or convolutional neural networks means that the evaluation is done in a supervised setting. Consequently, the traces in the training and validation phases need to be labeled according to the defined leakage model. In this procedure, the attacker also defines the number of classes that a neural network must be able to identify (and eventually classify) from the input trace set. As an example, if the leakage model is built over the S-box output of the first AES round, the number of classes that a neural network expects to see in the training data is:

- 9, if Hamming weight (HW) model is selected;
- 256, if identity (ID) model is selected;
- 2, if the leakage model refers to a single bit in the S-Box output.

An obvious observation is that more classes require more traces in the training phase. The neural network must process a minimum sufficient number of examples per class to be able to recognize the classes. Otherwise, the neural network will not be able to achieve satisfactory accuracy and generalization. The usage of identity leakage model should be avoided if the training and validation keys are the same, as usually happens when a closed sample (with a known key) is available for the training phase. In this case, the Hamming weight leakage model is a better choice.

Deep learning is, therefore, able to classify side channel traces according to the labels derived from the leakage model. For symmetric-key algorithms, like DES or AES, the attacker can implement a divide-and-conquer process, where one key byte is attacked at a time. Therefore, the number of times that a neural network must be trained is equivalent to the number of bytes in the key. However, the classification of traces into separate classes does not directly allow the attacker to reveal the key. For that, a key enumeration process is necessary. The output of the neural network provides the classification probability for each class. These class probabilities are then associated to key byte hypothesis in order to extract the likelihood for each key byte candidate.

Related works

Several works already investigated the performance of deep neural networks in side channel analysis. In (H. Maghrebi, 2016), the authors compared the performance (in terms of number of traces) of multiple layer perceptron and convolutional neural networks against other supervised methods, like machine learning and template attacks. The work proposed in (E. Cagli, 2017) evaluates the performance of convolutional neural networks in scenarios where side channel traces are misaligned due to countermeasures or hardware-related effects (clock jitter). The authors demonstrate that convolutional neural networks are able to suppress the misalignment effect if combined with regularization techniques like data augmentation. A detailed analysis about neural network hyper-parameters for side channel analysis is given in (E. Prouff, 2018). The authors apply different hyper-parameters combinations on unprotected and protected AES scenarios.

Advantages of deep learning for side channel analysis

Deep learning is a very computationally expensive technique that usually requires more time and memory resources to be deployed. However, it has clear advantages for side channel analysis, like:

- Points of interest selection are no longer necessary. The automatic feature extraction of convolutional layers and dense layers can identify the features related to the labeled traces;
- Convolutional layers can extract features independently of their position in the data. Therefore, deep learning should be able to bypass jitter-based effects from unstable clock domains or even random delays countermeasures;
- Because deep learning is a highly parametric model, the classification accuracy, and consequently the success rate of a side channel analysis, can be optimized based on hyper-parameter optimization;
- Deep neural networks can implement highly complex functions. Consequently, deep learning might be able to break widely adopted countermeasures like masking or shuffling.
- Section 4 provides some application examples of deep learning on AES and ECC. We also compare deep learning against classical side channel methods.

Training a Neural Network for SCA

There are a number of factors that need to be taken into consideration when training a neural network for doing side channel analysis. In this section, we lay out the basic principles of training a neural network and achieving good generalization performance when training an artificial neural network.

Training accuracy

The number of traces used for training a neural network is, of course, related to the actual problem. Side channel analysis requires a large number of traces in the training process due to the very small statistical differences from one class to the other. Different from other deep learning application scenarios (e.g., images, where the features can be, in most cases, easily visualized), classic side channel attacks usually disclose the features in power or electromagnetic traces (or, simply, the leakage) using statistical methods that average out the noise (DPA) or model the noise in order to identify classes (Template attacks). On the other hand, neural networks learn by example where the layered structure implements a complex function that is able to identify the features presented in the training data. Therefore, the training phase must cover a wide variety of traces for every given class in the training set.

Furthermore, the number of internal parameters in a neural network (which are derived from the number of layers and neurons) must be sufficient to fit all the input features provided with the training set. A neural network will implement a function that will be able to map the input data (with a specific label class) to output probabilities that indicate “how much” the input data corresponds to each class. If the training set is too small, we can easily overfit the parameters. It means that the trained neural network is very capable of classifying the small training set with very high accuracy (close to 100%); however, it is unable to generalize to new data. To avoid such a scenario, the training set must have a minimum adequate number of traces with respect to the number of parameters in the neural network (weight and biases).

Another reason to ensure a good size for the training set is related to the minimum necessary classification accuracy. Once the trace set is classified by a neural network according to a specific leakage model, the classified traces are applied to a key enumeration or key recovery algorithm. This algorithm considers the output probabilities from the output layer in order to rank and return best key candidates. Even if the classification accuracy of the validation (or test) set is relatively small, the key recovery phase can still be successful. The reason for this relies on the fact that the attack phase on symmetric algorithms (AES, DES) needs to process several traces with the same key material. For this reason, a key enumeration process is able to eliminate wrong key guesses and make the correct key hypothesis to converge. Increasing the size of the training set may lead to higher classification accuracy. Consequently, successful key recoveries are possible with fewer traces in the validation and test phases.

The training accuracy is an important metric during the training process. By observing the evolution of the training accuracy after every processed epoch, the user is able to conclude whether the amount of training traces is sufficient for the neural network to learn and generalize. Furthermore, the training accuracy evolution indicates whether the back-propagation algorithm is stabilizing the correct weight and bias values after some time. The learning rate is a very important hyper-parameter that affects back-propagation algorithm steps during the training phase. Figure 4 shows an example of training accuracy evolution when the learning rate is too large to keep the model stabilized after the processing of, approximately, 250 epochs.

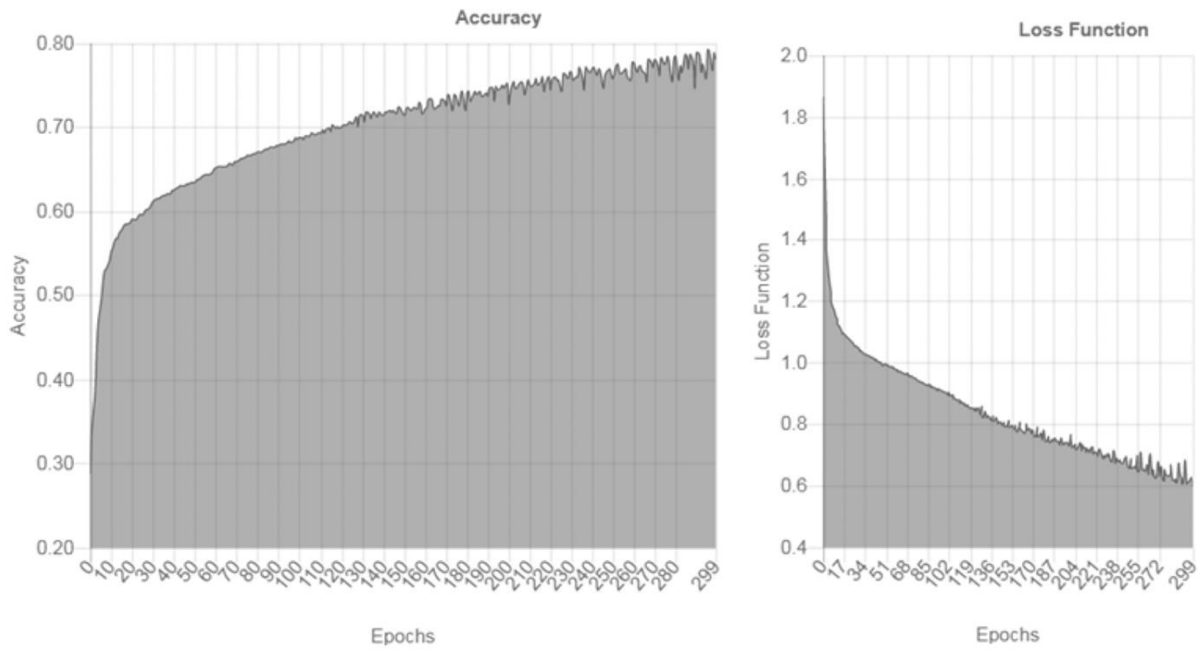

Figure 4. Training accuracy evolution — learning rate is too high

One of the solutions to prevent such a scenario of instability is to use learning rate decay rate during training. In this case, the learning rate is updated (and mostly reduced) during training. The consequence can be observed in Figure 5 in the accuracy and in the loss function evolution, where both become more stable after processing more than 250 epochs.

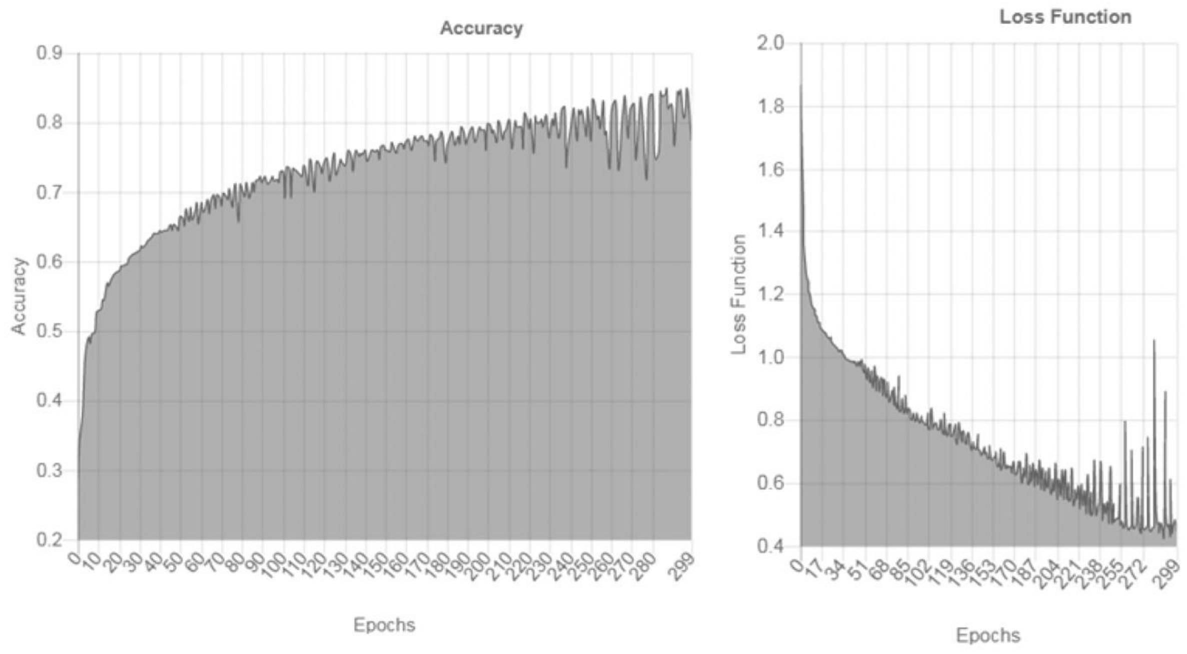

Figure 5. Training accuracy evolution — correct learning rate

Regularization parameters for generalization

The selection of hyper-parameters is the most important step in deep learning and neural networks. Different hyper-parameters have different influences in the neural network performance. The required training time to achieve satisfactory classification accuracy is influenced by the learning rate, mini-batch size, and number of epochs. Furthermore, some of the hyper-parameters affect generalization.

There are many solutions to improve generalization in deep learning. Solutions proposed by experts in the field suggest the usage of:

- Dropout;
- Early stopping;
- Regularization L1;
- Weight decay (similar to regularization L2);
- Data augmentation;
- Batch normalization (even if this method is not directly implemented for this purpose).

Regularization L1 and L2 are considered as a penalty in the weight updates in order to prevent them from being too large. Typically, the overfitting happens when the neural network is trained for quite a long time and the weights become too large. Therefore, regularizers, such as L1 and L2, prevent this effect. The L1 regularization penalizes the sum of the absolute values of the weights. The L2 regularization penalizes the sum of the squared values of the weights. Furthermore, L1 regularization also helps in the feature selection by pruning unneeded features by setting their associated weights to zero. As a consequence, by setting optimal regularization L1 and L2 values, the training accuracy will not reach 100% (because the model cannot perfectly fit the input data anymore) and the generalization can happen more easily. A similar effect can be observed when the number of training traces is increased.

Dropout is a technique that turns off a random subset of the neurons from the layers during training. This method can prevent a single neuron from overshadowing all the other neurons in a layer during training.

Early stopping tries to prevent the overfitting by returning the best achieved model within a pre-defined training time. The metric for the early stopping should be the validation accuracy.

Data augmentation, as the name suggests, uses the original data set to produce additional modified data in order to increase the capability of the neural network of dealing with different features in the data.

The *mini-batch* size used for training is also a determining factor in generalization. The stochastic gradient descent algorithm and its variants are employed in a mini-batch regime (batch sizes of 32, 64… 512). According to (N. S. Keskar, 2017), large batch sizes result in a bigger generalization gap. By increasing the batch-size for a specific problem, the analysis will find a threshold after which the quality of the model will deteriorate. What is then observed is a drop in the testing accuracy. It is also important to mention that, unlike the training with the entire batch (full training set), the cost function does not decrease very smoothly when using mini-batches. An alternative idea is to initialize the training with a small batch (called warm-starting) and then gradually increase the batch size during epochs.

In addition, small batch sizes offer an inherent regularization effect, maybe because they add noise effects to the learning process. Training with very small batch sizes also requires a very small learning rate (e.g., 0.001 or smaller) in order to maintain stability.

The number of hidden layers

Deep learning actually receives this denomination because the neural network usually contains tens of hidden layers. However, a large number of hidden layers is not always needed for a successful side channel attack. The reason for that is because a very small accuracy is, sometimes, sufficient to execute a successful key recovery. The size of the training set, and the number of input samples, can be related to the number of hidden layers.

It is important to note that with more hidden layers, the neural network is more capable of learning from a training set. This is a benefit when the training set is large enough. Of course, increasing the size of the neural network directly increases the number of internal parameters and more time and computation power are necessary to properly train all these parameters. The direct consequence of having more hidden layers is that the loss function value decreases faster once the neural network starts converging. In this case, the training phase may require fewer epochs to achieve satisfactory results or successful key recovery. On the other hand, as already mentioned, if the number of parameters in a neural network is too big for the input data, the network can easily overfit the training set. Note that the necessary time to process an entire epoch can be very long if the number of hidden layers is too big.

Automated search for hyper-parameters

Neural networks are highly parametric models. The initial configuration of the hyper-parameters requires (in most cases) advanced knowledge about neural networks. Some of the hyper-parameters can be defined based on the target under evaluation and leakage model. Typically, the number of hidden layers and their number of neurons are selected based on the target side channel traces and leakage model. On the other hand, regularization hyper-parameters, learning rate, number of epochs, mini-batch size and activation functions are usually defined based on the observed training and validation accuracy/recall. For that, the neural networks must be trained several times and the user must adapt the hyper-parameters one by one, making the model to converge to local or global minima in the hyper-parameters “landscape”.

The search for an optimized group of hyper-parameters requires a lot of tests, which requires a lot of computation power and time. The reason for that is because for every selected group of hyper-parameters the neural networks need to be re-trained. Among the available hyper-parameter search solutions, we have:

- Grid search;
- Random search;
- Optimized search.

Random and grid search are the most basic hyper-parameter search techniques. These two methods search for hyper-parameters inside pre-defined ranges and do not use any optimization technique. The best achieved model (based on accuracy, recall, loss function) is assumed as being the best candidate. The literature has shown that random search performs better than grid search. Optimized searches consider the usage of advanced methods, like evolutionary (or genetic) algorithms, Bayesian optimization, simulated annealing, etc.

Practical Experiments

This section summarizes the experiments done on using deep learning methods to recover cryptographic keys from implementations featuring various countermeasures.

Bypassing misalignment with CNNs

Convolutional neural networks (CNNs) can deal with misaligned traces, as proposed in (E. Cagli, 2017). The results presented in this work demonstrate that CNNs can indeed overcome misalignment and jitter-based countermeasures with the application of data augmentation techniques.

The target is a software AES implementation, where strong misalignment is present, as shown in Figure 6. The displayed interval represents the processing of the S-box in the first round. In total, 45~000 traces are considered for training and 5~000 are considered for validation.

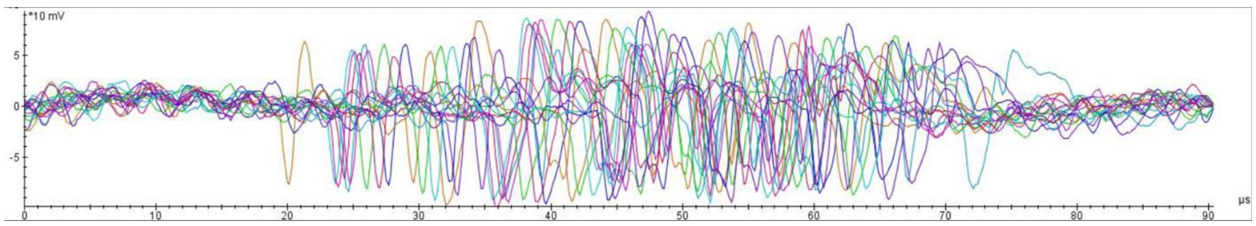

Figure 6. AES traces with misalignment

The trained convolutional neural network contains two convolution layers and three dense layers. The leakage model is Hamming weight of S-box output, meaning that 9 different classes are defined in the training set.

Figure 7 shows the key ranking results for convolutional neural network, correlation power analysis, and template attacks. As we can observe, CNN is the only scenario where the key recovery is possible against the evaluated target.

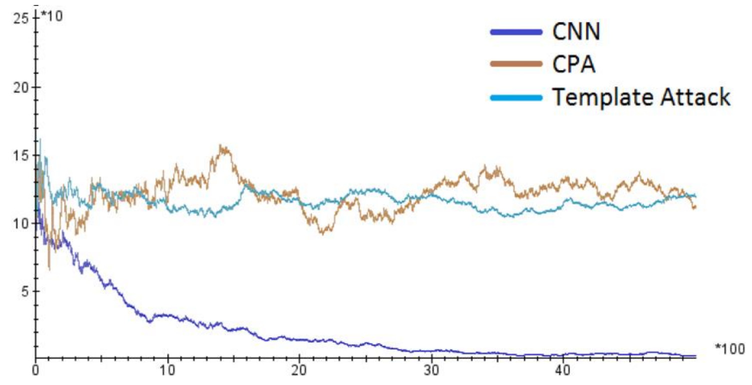

Figure 7. Key ranking evolutions for CNN, CPA and template attack

The comparison of key ranking evolutions demonstrates that the trained convolutional neural network is able to recover the key. Correlation power analysis and template attacks show no key convergence. This basic example shows that convolutional neural networks can efficiently deal with misalignment effects in side channel traces. Furthermore, jitter-based effects are present in most hardware devices due to instabilities in the clock domain or as an intentional countermeasure. Deep learning shows strong indications to bypass this effect by using convolution filters in neural networks.

Breaking masked AES

DPA Contest v4 is a public database that we consider as an application case. This trace set is collected from a target that implements AES with first order masking countermeasure. Such a countermeasure eliminates dependency between predictable intermediate states in a cryptographic algorithm and the power consumption (or the electromagnetic emanation). The database consists of 40,000 power side channel traces measured from a software implementation of AES-256. The implementation is protected with rotating S-box masking (RSM) scheme, also called a low-entropy masking scheme that provides resistance to first order attacks.

An overview of the DPA Contest V4 trace is shown in Figure 8.

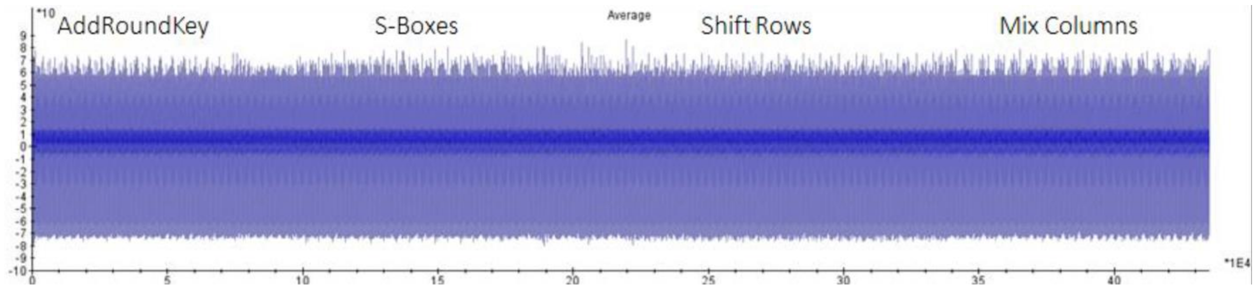

Figure 8. DPA Contest V4 side channel trace

The full trace represents the power consumption during the processing of the first AES round. The raw traces contain 435,000 samples each. The operations that we are interested in are the s-boxes and shift rows. This results in an interval of approximately 200,000 samples to apply the black-box deep learning analysis. This amount of input samples per traces renders the training process unfeasible due to memory and time constraints. The solution is to split the input trace into several sub-parts and then train the neural network for all the intervals. Off course, this procedure could be sped up with identification of the moment when the offset values (masks) are processed in the first round. This is possible because the mask values are provided together with the database.

The neural network is trained multiple times in a black box setting for different intervals. By doing so, we identify the interval where the validation accuracy is relatively high for the target key byte. Although the achieved validation accuracy is not sufficient to successfully recover the key, it was enough to identify the interval that has the most leaking samples. Training the neural network on the identified interval and after optimizing the training led to successful key recovery. Due to the limited amount of traces (40,000) and the selected short interval (1,000 samples), we were able to recover the key bytes by training a convolutional neural network with one convolution layer and three fully connected layers. In the key enumeration phase, on average 10 traces are sufficient to have the correct key byte to be ranked as the first key candidate.

In profiled side channel analysis, the key for training and validation (or test) should be different; otherwise the key recovery phase is not realistic. In fact, if the same key is used for training and validation, it is uncertain whether the neural network is actually learning from the leakage or some other features from the input trace. To confirm that this situation is not happening in the current scenario, we train and test the neural network on correct and on wrong (random) key byte values.

Figure 9 illustrates the training and validation recalls for correct and incorrect labels. The recall is the average of the individual accuracies per class. It is a more realistic metric when classes are unbalanced.

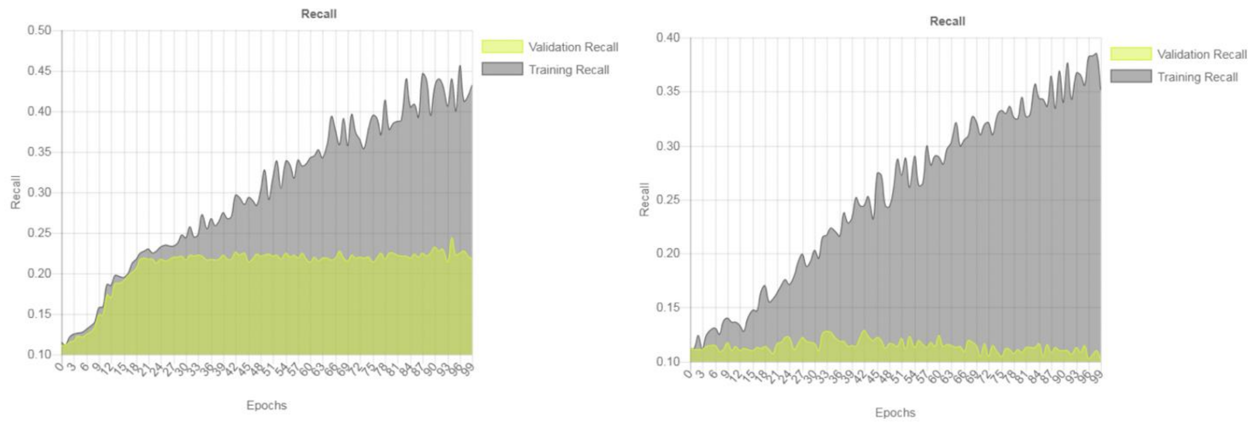

Figure 9. Training and validation recalls for the correct labels (left) and wrong labels (right)

As we can see in the figure, the validation recall (and generalization) is very low (around 11%) when incorrect labels are used for training the network; meaning that our model is actually learning the leakage from the traces. On the other hand, the validation recall for the correct labels achieved approximately 22%, which turned out to be sufficient for a successful key recovery.

Therefore, this experiment showed that deep learning is able to break slightly protected AES implementations.

Attacking protected ECC

State-of-the-art hardware implementations of elliptic curve cryptography consider several countermeasures in order to protect against private key extraction through side channel analysis. Among these countermeasures, point randomization, coordinate randomization and scalar blinding prevent the application of classical side channel analysis methods. In this experiment, the target is a Montgomery ladder implementation of the scalar multiplication, protected with scalar blinding and coordinate blinding. The implementation is based on Curve 25519 (Bernstein, Public Key Cryptography - PKC 2006).

Side channel analysis on protected public-key implementations (including RSA) is conducted in a different way than previously discussed in the paper. Instead of processing a trace set with the same key material, every trace contains measurements collected from an execution that uses a different (randomized) private key. Therefore, the attacker must work around this limitation by attacking a single trace. If an attacker can recover the randomized private key from a single trace, then an equivalent of the real private key can be obtained using basic calculus. This is done through a special type of side channel attacks called horizontal attacks.

The procedure for applying horizontal attacks can be summarized in the following way. The trace representing the power consumption of a scalar multiplication is divided into sub-parts, where each part represents the processing of one bit of the scalar. In the case of Montgomery ladder, every sub-part contains the processing of a point addition followed by a point doubling.

Even when horizontal attacks are applied in a supervised setting, the presence of misalignment directly reduces the points of interest identification performance. Convolutional neural networks are efficient in detecting leakages even when side channel traces are de-synchronized in the time domain. Another advantage here is that a convolutional neural network does not need to be informed about the location of points of interest. The network can discover these points of interests by itself.

In a supervised setting, deep learning is able to recover 100% of the scalar bits if the traces are properly aligned in time domain. The same performance can be achieved over the attacked trace set if profiled template attacks are considered. For the case when the traces are not aligned, supervised template attacks are only able to recover approximately 60% of the scalar bits in the test phase. However, a convolutional neural network can recover approximately 90% of the scalar bits.

To improve the classification accuracy, an optimal set of hyper-parameters must be identified. Another way to improve classification accuracy is to adopt data augmentation as a regularization method. To do so, we augment the original training set by inserting artificial (random) shifts in the traces. This mechanism provides more examples to the network, helping it to learn from a larger variety of misaligned traces. After data augmentation, the classification accuracy in the test phase raised to 99.4%. The rest of the bits can be recovered using a brute force strategy.

Conclusion

This paper presented an overview of deep learning for side channel analysis. We provided an overview of the methodology that is commonly applied to side channel attacks when neural networks are considered as the model for the key recovery. The paper also covered the details for training a neural network for SCA. We applied convolutional neural networks on different targets and demonstrated that CNNs can efficiently bypass trace misalignment and recover the key bytes. Furthermore, we also applied a convolutional neural network on slightly protected AES (DPA Contest V4) as a black-box attack. Finally, a CNN is also trained to bypass misalignment in ECC traces. The applied method achieved 99.4% of accuracy when applied on single ECC traces.

Bibliography

Bernstein, D. J. (Public Key Cryptography - PKC 2006). *Curve25519: New diffie-hellman speed record.* 2006.

E. Cagli, C. D. (2017). *Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing.* Cryptographic Hardware and Embedded Systems.

E. Prouff, R. S. (2018). *Study of deep learning techniques for side-channel analysis and introduction to ASCAD database.* IACR Cryptology ePrint Archive 53.

H. Maghrebi, T. P. (2016). *Breaking cryptographic implementations using deep learning techniques*. Lecture Notes in Computer Science, Springer.

N. S. Keskar, D. M. (2017). *On large-batch training for deeplearning: Generalization gap and sharp minima..* CoRR.

Y. Lecun, L. B. (1998). *Gradient-based learning applied to document recognition.* Proceedings of the IEEE.