Executive Summary of the Thesis

Towards Efficient Training in Deep Learning Side-Channel Attacks

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Luca Castellazzi

Advisor: Prof. Luca Oddone Breveglieri

Co-advisors: Prof. Alessandro Barenghi, Prof. Gerardo Pelosi

Academic Year: 2021-2022

1. Introduction

Today's digital world highly relies on data and their security is one of the most crucial aspects. Side-Channel Attacks (SCAs) represent a serious threat to data security, offering a non-invasive way of retrieving secrets processed by a microcontroller during the execution of cryptographic algorithms. Recent research enhanced SCAs exploiting Deep learning (DL) networks to retrieve the secrets. The usage of multiple devices for training showed that DL-based SCAs can be effective even in real-world scenarios against an unprotected implementation of the Advanced Encryption Standard (AES), being able to successfully attack a device that is different from the ones used for training.
In addition to multiple devices, this work considers several training-phases with multiple encryption keys and different amounts of data, showing that these three variables affect the performance of DL-based SCAs in real-world scenarios. Device-Key Trade-off Analysis (DKTA) is introduced to study the trade-off between number of devices and number of keys, while Device-Trace Trade-off Analysis (DTTA) to investigate the trade-off between number of devices and number of traces. This work aims at providing efficient training-configuration to mount effective realistic DL-based SCAs against a software implementation of AES-128, both with and without masking countermeasures.

2. Background

2.1. Deep Learning

Deep Learning (DL) is the sub-field of Artificial Intelligence that aims at automatically extrapolating relevant features from raw data exploiting Artificial Neural Networks (ANNs).
The Multi-Layer Perceptron (MLP), depicted in Figure 1, is the ANN considered throughout this work.

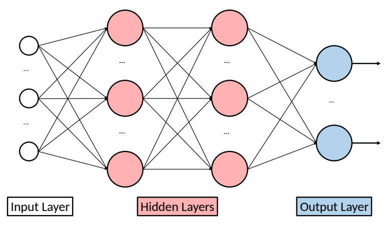

Figure 1: Multi-Layer Perceptron (MLP).

It is a fully-connected network that involves an input layer, containing as many neurons as the input features, multiple hidden layers of varying number of neurons and an output layer which contains exactly as many neurons as the num-

ber of possible output values.
An MLP can be used, as in this work, to solve complex classification problems, where one must identify which category, from a fixed set, a given input belongs to. In this case, the output layer features exactly as many neurons as the number of classes and the actual output of the network is the probability that the provided input belongs to each class.
The training process involves the automatic tuning of the *internal* parameters of the MLP in a *supervised* way: during training, the inputs are provided to the network with the related expected output, so that the network can learn their dependency.

2.2. Side-Channel Attacks

Side-Channel Attacks (SCAs) allow to retrieve a secret encryption key from a device in a non-invasive way. Statistical methods are exploited to extract the correlation (*leakage*) between device-related physical variables (e.g., Power Consumption, Electro-Magnetic emissions) and processed sensitive information, making it possible to retrieve the key. The physical variables are captured from the device through measurements, known as *traces*. Moreover, usually the exploited processed information is not the key itself, but rather an intermediate value of the targeted algorithm. Attacking AES, as performed in this work, the most common target is *sbox-out*, the output of the first round *SubBytes* operation, since it adds non-linearity to the algorithm. Moreover, it offers an easy way to recover the key, being *SubBytes* easily invertible. Figure 2 shows how to compute *sbox-out* from plaintext and key and how to compute the key from plaintext and *sbox-out*.

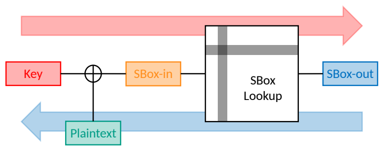

Figure 2: First Round *SubBytes* Operation, Forward (left-to-right) and Backward (right-to-left).

Profiled-SCAs extends traditional SCAs considering the attacker able to exploit a sample of the target device to produce a model of the leakage. The model is generated during the so-called Profiling Phase, while the key is recovered from the actual target-device during the Attack Phase.
SCAs can be thwarted by several countermeasures, which aim at either reducing the leakage or making it uncorrelated with the secret. In this work, a masking countermeasure [5] of order 1 is considered: it splits the secret x into 2 shares x_0, x_1 such that $x_0 \oplus x_1 = x$. In particular, x_0 is the masked value, while x_1 is the random mask. This simple countermeasure exponentially increases the complexity of the attack, since the attacker, to recover the secret, needs to exploit not only a single leakage point, but two, one related to the mask and one related to the masked value.

2.3. Deep Learning based Side-Channel Attacks

One of the emerging trends in SCA field is the use of DL networks to produce a model of the leakage and then retrieve the secret [4]. In particular, the SCA is treated as a classification problem where the input to be categorized is a measurement taken from the target-device and the classes are all the possible values of the targeted intermediate value. A final key-recovery step allows to retrieve the secret key.
The absence of assumptions about the distribution of the noise in the measurements and the ability of correctly retrieving the secret with few attack samples are the main advantages of DL-based SCAs over traditional Profiled-SCAs. In addition, the ability of ANNs to perform automatic feature extraction from data allows to completely automate the Profiling Phase, especially targeting a masked implementation of AES, as shown in [4].
On the other hand, the usage of ANNs in real-world scenarios makes this approach affected by the so-called *Portability Problem*: if the network focuses only on recognizing noise and details from the sample-device rather than learning its general behavior, then it would provide wrong predictions during the Attack Phase against a different device. Therefore, the goal is to build a network that is general enough to successfully attack a device which is different from the one used during training. The Multiple Device Model (MDM) [1] is the state-of-the-art solution

(validated only on 8-bit CPUs) to the Portability Problem. It suggests the usage of at least 2 devices during the Profiling Phase. Indeed, additional devices allow to add diversity to the training set, making the network more general and able to attack a different, unseen device.
This work aims at providing efficient training-configurations to mount effective realistic DL-based SCAs. In order to do so, three main variables are considered: the number of train-devices, the number of keys used during the Profiling Phase and the number of train-traces. Therefore, differently from [1], this work addresses the Portability Problem not only in terms of number of devices, but also considering the impact of other two variables. Moreover, for the first time, this work applies the MDM both on 32-bit CPUs and against a masked implementation of AES.

3. Methodology

3.1. Dataset Construction

All the data used during the experiments, including traces and multiple keys, were generated throughout this work. In particular, the power consumption traces exploited to perform the attacks were collected by an oscilloscope at 500MHz and then re-sampled at 168MHz, the clock-frequency of the targeted devices. This re-sampling technique allows to capture all needed information and, at the same time, reduce the number of samples per trace, since it is the number of neurons of the input layer of the MLP. The capture of the traces is followed by a labeling phase, needed to apply the *supervised* approach of MLP training: this procedure is performed emulating AES as shown in Figure 2 (forward step). Traces are also scaled to 0-mean and unit-variance to simplify MLP convergence.

3.2. Hyperparameter Tuning

For simplicity, a single MLP is trained to retrieve a single byte of the secret key, so a total of 16 models is needed to extract the full key. The training process is preceded by the definition of the *hyperparameters*, all the *external* parameters of the MLP (e.g., number of hidden layers) that define its structure and control its training. The following Genetic Algorithm, based on [3], was considered to find their optimal configuration.

Algorithm 1 Genetic Algorithm for Hyperparameter Tuning

```
$pop \leftarrow$ POPULATE($hpSpace$, $popSize$)
for i $\leftarrow 0$ to $nGen - 1$ do
    $sortedPop \leftarrow$ EVALUATE($pop$)
    $parents \leftarrow$ SELECT($sortedPop$, $selPerc$)
    $pop \leftarrow$ EVOLVE($parents$, $mutProb$)
end for
$bestHPs \leftarrow sortedPop[0]$
return $bestHPs$
```

Algorithm 1 is based on the *Natural Selection Principle*: starting from an initial population of random hyperparameters, for a fixed number of generations, the population is evolved considering, for the next population, only the hyperparameters that lead to the best MLP performance. Random mutations are added to the process to increase the diversity of the populations. Throughout this work, Algorithm 1 is run every time a train-device, the targeted algorithm or the targeted byte are changed, to obtain the best MLP possible for the given situation.

3.3. Key-Recovery and Attack Evaluation

The considered DL-based SCAs differ from classical classification problems: the MLP is trained to provide the value of a single byte of *sbox-out*, while the target of the attack is a single byte of the key. For this reason, an additional step is needed to recover the key from MLP predictions. It computes the value of the key-byte starting from the predicted *sbox-out* and the plaintext, as shown in Figure 2 (backward step).
Moreover, in DL-based SCAs, the network provides a prediction for each single attack-trace, but the key-byte to be recovered is actually shared among all of them. Therefore, in this work, the MLP predictions are combined to generate a ranking of all possible key-bytes: the key-byte in position 0 (counting 0-based) is the actual result of the attack, being the most probable outcome given by the network.
This ranking is useful to evaluate the attack performance. The Guessing Entropy (GE), defined as the average position of the correct key-byte in the final ranking over on multiple independent experiments is used as metric. In this work, positions (ranks) range from 0 to 255 (since a single byte is targeted), meaning that an attack able to retrieve the correct key-byte is expected to reach $GE = 0$. Given multiple attacks, the best one is

the attack that achieves $GE = 0$ (or the lowest GE) using the smallest number of attack-traces.

3.4. Trade-off Analysis of DL-SCAs

This work identifies the number of devices, the number of keys and the number of traces as the three main variables of the Profiling Phase of a DL-based SCA. To study the influence that these variables have on the attack performance and consequently provide efficient training-configurations to mount effective realistic attacks, this work introduces the Device-Key Trade-off Analysis (DKTA) and the Device-Trace Trade-off Analysis (DTTA). These techniques allow to analyze the trade-off between the three main variables tracking the performance of multiple attacks in terms of GE, where the number of keys (for DKTA) or the number of traces (for DTTA) is gradually increased. The experiments are repeated for increasing number of devices (at most 2). In this way, it is possible to completely characterize the Profiling Phase with respect to its most important variables, highlighting their influence on the attack performance. The provided results are robust, averaging the outcome of attacks performed with all possible device-combinations available with the considered setup.
Moreover, DKTA and DTTA allow to highlight possible scenarios where the number of keys and the number of traces, respectively, can enhance the attack performance in case of limited number of devices, providing possible alternative solutions to the Portability Problem.

4. Experimental Evaluation

4.1. Experimental Setup

Table 1 summarizes the capturing setup used for collecting the traces.
The code relative to all experiments was developed, from scratch, in `Python 3.8.10`. In particular, DL scripts were implemented with `Keras` considering a `Tensorflow` backend. A Nvidia GeForce RTX 2080 Ti GPU with 11GB of dedicated DDR6 memory provided the computing power needed to perform DL training. The collected traces are available on Zenodo [2] while the code can be found at `https://github.com/luca-castellazzi/EfficientTraining_DLSCA`.

Capturing Setup

Instrument	Name	Description
Target Device	Riscure Piñata v2.3.1	STM32, Cortex-M4 CPU, 168MHz clock-frequency
Oscilloscope	Tektronix MS58	Max sample-rate of 6.25GS/s
Capturing Application	Riscure Inspector v2022.1	Trace collection, Attack simulation

Table 1: Capturing Setup for Trace Collection.

4.2. Results Targeting Unprotected-AES

Figure 3 shows a compressed representation of the results of DKTA targeting byte 5 with one and two train-devices. Each single curve is the GE obtained with a specific amount of keys, while the color indicates the number of devices involved.

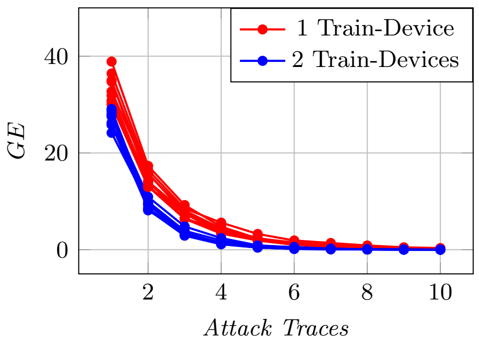

Figure 3: DKTA Results Targeting Byte 5 against Unprotected-AES.

The attacks are successful even when a single train-device is considered, retrieving the correct key-byte in 9 attack-traces, but the usage of an additional one allows to reduce this number to just 6. This results shows that, with the considered setup, the Portability Problem is minimal. The number of keys appear not to influence the attack, since all GE curves are almost overlapped and converge to $GE = 0$ at the same time. This motivates the choice of not differentiating the single curves with several colors.
Further analysis targeting byte 0, 11 and 14 highlighted that the number of keys influences

only the specific case of attack against byte 0 with two train-devices, since considering at least 2 keys ensures convergence to $GE = 0$ in less than 10 attack-traces. On the other hand, the number of devices always ensures performance-boost, decreasing the number of needed attack-traces. In addition, it was found that targeting byte 5 is easier than targeting byte 14, which is itself easier than targeting byte 11: this is due to the *jitter effect*, a partial misalignment of the traces more visible towards the end of the measurements. Byte 14 and byte 11 are more affected by this problem than byte 5 because they are computed later during the execution of *SubBytes* and, in particular byte 5 is computed before byte 14, which is computed before byte 11. On the other hand, byte 0 is not affected by the *jitter effect*, but it shows anyway a quite bad performance. This happens because it is computed first, being affected by the *warm-up problem* of the instruction cache: when byte 0 is computed, the instruction cache of the processor is empty (*cold*) and the instructions are taken directly from the flash memory, while from the second iteration on, the cache is full of instructions (*warmed-up*).
Figure 4 shows the results of DTTA in terms of minimum number of attack-traces needed to achieve $GE \leq 0.5$, a good approximation for $GE = 0$.

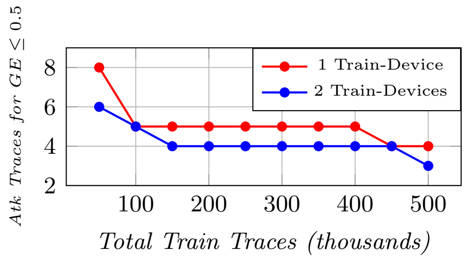

Figure 4: DTTA Results Targeting Byte 5, Unprotected-AES.

The results highlight how the number of train-traces allows to gain a 2× performance improvement by adding 450,000 traces to the starting training set. Such an influence of the number of train-traces over the attack is independent of the number of devices, which still increases the performance allowing, in the best scenario, to recover the key-byte in only 3 attack-traces.

4.3. Results Targeting Masked-AES

Figure 5 shows the results of DKTA targeting byte 5 against masked-AES: the upper figure shows the results obtained with a single train-device, while the lower one presents the outcome obtained with two train-devices.

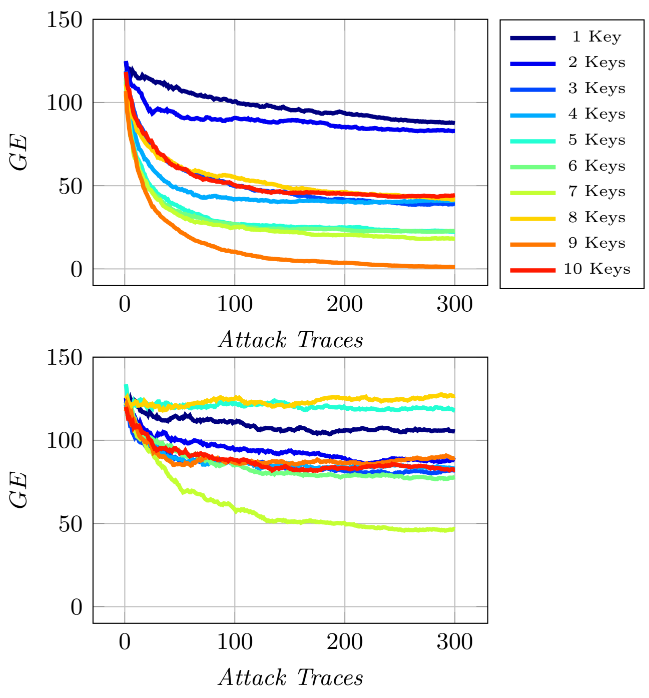

Figure 5: DKTA Results Targeting Byte 5, Masked-AES.

The results show that the considered MLP is able to *break* even a masked implementation of AES-128, since at least a GE curve reaches 0. The number of keys influences the attack: higher number of keys are preferred, even if the performance improvement is not monotonic. In particular, $GE = 0$ is achieved with 9 keys and 300 attack-traces. On the other hand, considering an additional device, on average, makes the attack unfeasible, causing the GE to diverge.
Figure 6, instead, shows the results of DTTA obtained with a single train-device (solid lines) and with two train-devices (dashed lines) in case of masking countermeasures. Also in this case, the addition of a device makes it impossible, on average, to retrieve the correct key-byte, even with 300 attack-traces. On the other hand, the performance of the attack increases monotonically with the number of traces. Indeed, the best result is obtained with 350,000 traces, since GE reaches 0 in only 150 attack-traces. The num-

ber of used keys is 9, relying on the results of DKTA.

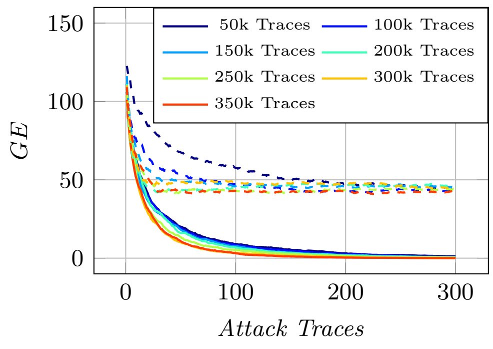

Figure 6: DTTA Results Targeting Byte 5, Masked-AES.

The results shown in Figure 5 and Figure 6 are *averages* over the attacks performed with all possible combinations of train-devices and target-device. A further analysis of the single attacks revealed that it is possible to recover the key with even fewer attack traces and sometimes also with two train-devices. At the same time, some attacks involve a network which acts like a random classifier, causing the divergence of the average. In particular, the attacks where the train-devices coincide with the ones used during Hyperparameter Tuning achieved the best performance: this demonstrates that the Portability Problem can manifest itself not only in terms of difference between the train and target device, but also in terms of choice of the train-devices.

5. Conclusions and Future Work

This work offers a complete characterization of the Profiling Phase of a DL-based SCA targeting AES-128, both with and without masking countermeasures and running on a 32-bit CPU, in terms of trade-off between its most important variables. When targeting unprotected-AES, the number of devices and the number of traces highly influence the attack performance, because adding a device improves the attack in all scenarios, validating for the first time the MDM [1] on 32-bit CPUs, and a bigger training set (up to 500,000 traces) always allows to perform better attacks. The number of keys appears to be much less relevant, boosting the performance only in the specific case of attack against byte 0 with two train-devices. When targeting masked-AES, all variables influence the attack: an high number of keys (9) and traces (350,000) allows to perform better attacks, while the addition of a device makes them unfeasible, on average. In addition, this work demonstrates that the Portability Problem, with the considered setup and when masking countermeasures are in place, can manifest itself in terms of choice of the train-devices, rather than in terms of difference between train-devices and target-device: if the train-devices coincide with the ones used during Hyperparameter Tuning, then the attack performs better.
Even if this work focuses entirely on AES, the proposed methodology can be applied to any encryption algorithm. Therefore, future works can involve DKTA and DTTA over an hardware implementation of AES, AES with a different mode of operations or pipelined-AES. Moreover, this work can be further extended considering Electro-Magnetic (EM) emissions as side-channel, relaxing the implicit assumption of physical access to the pins of the target-device, needed to connect the current probe, at the cost of additional noise in the measurements.

6. Acknowledgements

A special thanks for the collaboration goes to Prof. Luca Oddone Breveglieri, Prof. Alessandro Barenghi, Prof. Gerardo Pelosi, and to Dr. Paolo Amato, Dr. Danilo Caraccio, Dr. Niccolò Izzo.

References

[1] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. Mind the portability: A warriors guide through realistic profiled side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 661, 2019.

[2] Luca Castellazzi. Towards Efficient Training in Deep Learning Side- Channel Attacks, April 2023. `https://doi.org/10.5281/zenodo.7817187`.

[3] Matt Harvey and Norman Heckscher. Evolve a neural network with a genetic algorithm. `https://github.com/harvitronix/neural-network-genetic-algorithm`, 2017. Published under MIT License.

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

[5] Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.