On the Performance of Non-Profiled Differential Deep Learning Attacks against an AES Encryption Algorithm Protected using a Correlated Noise Generation based Hiding Countermeasure

Amir Alipour, Athanasios Papadimitriou, Vincent Beroulle, Ehsan Aerabi, David Hély
Univ. Grenoble Alpes, Grenoble INP, LCIS F-26000 Valence, France
{firstname.lastname}@lcis.grenoble-inp.fr

Abstract— **Recent works in the field of cryptography focus on Deep Learning based Side Channel Analysis (DLSCA) as one of the most powerful attacks against common encryption algorithms such as AES. As a common case, profiling DLSCA have shown great capabilities in revealing secret cryptographic keys against the majority of AES implementations. In a very recent study, it has been shown that Deep Learning can be applied in a non-profiling way (non-profiling DLSCA), making this method considerably more practical, and able to break powerful countermeasures for encryption algorithms such as AES including masking countermeasures, requiring considerably less power traces than a first order CPA attack. In this work, our main goal is to apply the non-profiling DLSCA against a hiding-based AES countermeasure which utilizes correlated noise generation so as to hide the secret encryption key. We show that this AES, with correlated noise generation as a lightweight countermeasure, can provide equivalent protection under CPA and under non-profiling DLSCA attacks, in terms of the required power traces to obtain the secret key.**

Keywords—Deep Learning, Side Channel Analysis, Profiling SCA, non-Profiling SCA, AES encryption algorithm, Hiding-based AES countermeasure

I. INTRODUCTION

Every year there is a greater demand for devices which process data which need to be secure. Side Channel Attacks (SCA) are a serious threat against these devices, as they can expose the secret encryption keys [1]. Deep learning methods have been proven potentially powerful for many different data analysis applications in which the analyzed data are chunks of data that are accompanied with distortions and noise [2]. Even though Deep Learning based Side Channel Analysis (DLSCA) is a fairly new field, it is very promising for performing powerful attacks against cryptographic implementations. DLSCA have been applied in two ways, profiling and non-profiling [3].

In profiling DLSCA the attacker uses a copy of the target device to record its power consumption during the execution of the target encryption algorithm (e.g. AES) [4]. In this case, the attacker has a full control over the copy of the device. The attacker therefore uses the experimentally obtained power traces of the copy of the device to train a neural network, and proceeds into constructing a template model which is able to identify the correlations between the intermediate values of the encryption algorithm and the power traces of the target device. Then, the trained model is used to analyze the power traces obtained from the actual target device, and to reveal its secret key. Profiled DLSCA has been proven capable to reveal secret encryption keys besides the presence of strong countermeasures, such as masking [5] and jitter-based countermeasures [11]. However, since this method assumes having access to a copy of the target device with full control over the implementation of the encryption algorithm as well as knowledge of the secret key, it is oriented towards evaluations. Such strong assumptions may lead to evaluations which are a lot stronger than possible attacks, which in turn may lead to over-constrained countermeasures and thus increased costs.

On the other hand, non-profiling DLSCA, can be used to evaluate secure devices in a more realistic way, due to less constraining assumptions. Other non-profiling SCA attacks include Correlation Power Analysis (CPA) and Differential Power Analysis (DPA) attacks. To apply these attacks, the attacker only uses the power traces captured from the actual target device using only one secret key. However, since the attacker has no knowledge of this secret key, he makes hypotheses for all the possible values of the part of the secret key under attack (e.g. one key byte). Using non-profiling SCA attacks, including non-profiling DLSCA, the attacker is able to use the power traces to reveal which key hypothesis is correct. For DLSCA, this analysis happens during the optimization (also known as training) of an Artificial Neural Network (ANN) model. Later in the coming sections we will explain how this analysis can lead to compromising the encryption algorithm and reveal the secret key using non-profiling DLSCA.

While proven as a successful attack, non-profiling DLSCA has been applied on only few designs and types of countermeasures. The attack has been applied so far on simulated datasets, on unprotected hardware-based AES and on software-based AES protected by masking [6][7]. In this study, we present the results of attacking a software-based AES, running on a 32-bit ARM-based Micro-Controler Unit (MCU), by means of a non-profiling DLSCA attack, which is also called Differential Deep Learning Analysis (DDLA) attack. The AES algorithm is protected by a hiding countermeasure based on complementary memory reads and writes and a protection based on an additional correlated noise generation. This countermeasure was initially described in [8] for AES hardware implementations and then applied to AES software implementations [9]. This countermeasure is a low overhead solution, so as to protect modern MCU applications concurrently against SCA and Fault attacks. In this work, by attacking this AES countermeasure, we show that non-profiling DLSCA attack, described so far in the literature, can be approximately equally strong as a CPA attack. While on the contrary, according to [6], when a non-profiling DLSCA attack is performed against a masked countermeasure, the attack is very strong and reveals the secret key with only a few thousand power traces.

The rest of the paper will be as follows: In section II, we will explain the Differential Deep Learning Attack (DDLA) as the core method of non-profiling DLSCA attack. In section III, we show our experimental results with DDLA against different implementations of AES. Finally, in section IV we conclude the work.

II. METHODOLOGY OF DIFFERENTIAL DEEP LEARNING ATTACKS

First, the main goal of non-profiling DDLA, as mentioned in the introduction, is to compute the training accuracy for each key hypothesis and identify the secret encryption key as the key guess having the maximum accuracy. Take note that accuracy refers to the correctness of classification of a reduced characterization, such as the Most Significant Bit (MSB), of an intermediate value in AES [7]. In more details, to perform a DDLA attack, the attacker must initially provide a list of key-byte hypotheses *KH*. For a byte oriented cipher, we have:

$$KH \in \{0,1,\dots,255\} \quad (1)$$

In the next step, the attacker moves forward to the labeling process. Labels, in DDLA, are values assigned to each power trace and characterize it according to one of its properties. In our case, potential candidates for these labels are the values obtained by applying a power model to the cipher's hypothetical intermediate values. Possible power models include, the Hamming Weight (HW), the Most Significant Bit (MSB) and the Least Significant Bit (LSB) [6]. Therefore, by taking MSB as our power model, we can label each power trace according to the following formula:

$$L_{i\_\{K,j\}} = MSB\left(Sbox\left(P_i, KH_j\right)\right) \quad (2)$$

Where Pi is the plaintext for the byte under attack and KH_j is the key hypothesis for the same byte. This way the attacker can compute one label for each power trace (0 to N) and each key hypothesis (0 to 255) and use it for training his ANNs, as shown in (3):

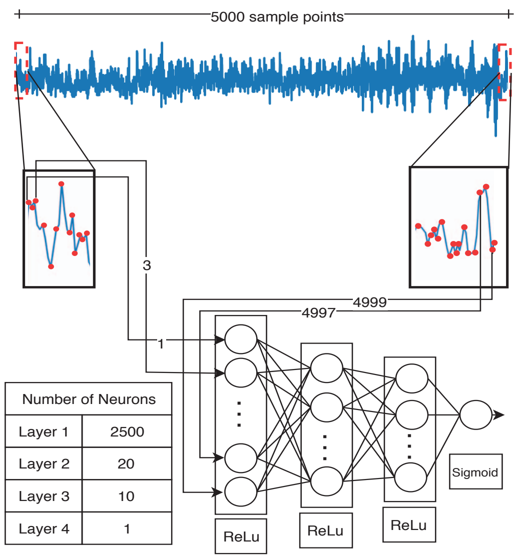
$$KH_{0_{Label_{List}}} = \{L_{1,k0}, L_{2,k0}, L_{3,k0}, \dots, L_{N,k0}\}$$

$$KH_{1_{Label_{List}}} = \{L_{1,k1}, L_{2,k1}, L_{3,k1}, \dots, L_{N,k1}\}$$

…

$$KH_{255_{Label_{List}}} = \{L_{1,k255}, L_{2,k255}, L_{3,k255}, \dots, L_{N,k255}\} \quad (3)$$

Wherein N is the total number of power traces in the dataset. The next step for the attacker is to design a model of each ANN he will train in order to perform the attack with each KH_i Label List (3). For this purpose, several options exist, such as modeling a Multi-Layer Perceptron (MLP), or Convolutional Neural Network (CNN) [10]. In common cases, MLPs are types of ANN which are relatively faster to train compared to CNNs.



Number of Neurons	
Layer 1	2500
Layer 2	20
Layer 3	10
Layer 4	1

Figure 1. Diagram of a binary classifier MLP.. Mentioned ReLu and Sigmoid on the diagram are activation functions defined for each layer.

In addition, , CNNs are suitable models for detailed feature extraction and handling complicated data classification models, MLPs on the other hand are for simpler classification applications. Also relative to our context, on one hand, CNN have the advantage of being resilient to temporal distortion, on the other hand MLPs are susceptible to it [11]. In this work, we assume that our traces are free from temporal distortion, and therefore MLP networks are a suitable and efficient choice. Fig. 1 shows a binary classifier MLP which is modelled for DDLA and Table 1 shows the optimization parameters of the model. This specific model can be trained to predict given a specific power trace of an encryption, whether its Sbox output has an MSB equal to one or zero. This type of classification is referred to as binary classification [12]. During constructing our MLP, our aim was to reconstruct the same model used in [7]. Since each of the power traces has 5000 sampling points, in order to reduce the size of our DDLA model and thus reduce its training duration we reduced by 50% the sampling points of each power trace. In detail, from each two neighboring sample points, the first one is kept, while the second is omitted (Fig. 1).

Table 1. Table showing the optimizer parameters for the given MLP

Optimizer	Learning Rate	Loss Function	Metric
Adam	0.001	Binary Crossentropy	accuracy

During the training phase, the attacker trains a new model for each of the KH_i Label Lists and the corresponding power traces. Noting that for each model the attacker starts each train-

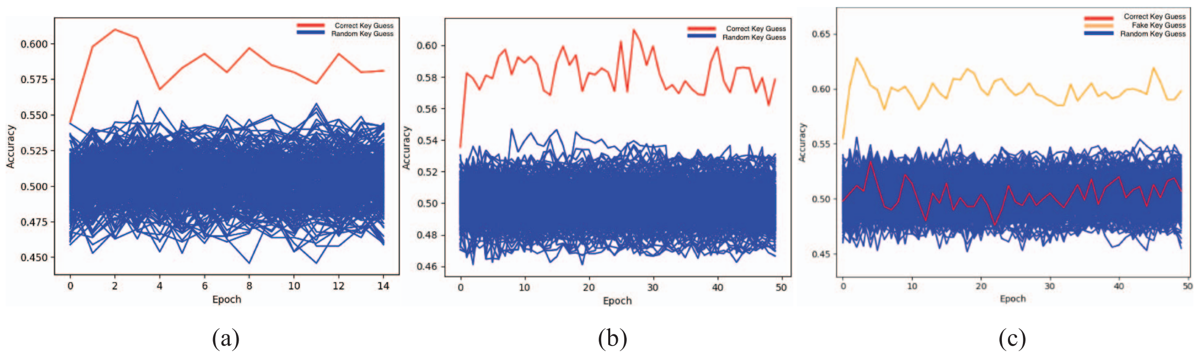


Figure 2. Accuracy graphs of DDLA attack, (a) belonging to DDLA against AES unprotected, (b) belonging to DDLA against AES with masking and (c) to DDLA against AES with fake-key

ing with an un-trained model. In the end the attacker uses the classification accuracies so as to determine the secret encryption key. In DDLA, the trained network with the highest accuracy (between all the 256 networks) will correspond to the secret encryption key [6]. Since the assigned labels for this trained network are the ones which have been calculated using the correct secret key, the key hypothesis used to build these labels will be the most correlating with the power traces. In the coming section, we will show our experimental results by performing DDLA attacks.

III. EXPERIMENTAL RESULTS AND DISCUSSION

Our experimental results include performing DDLA against an unprotected AES, an AES protected with masking, and one AES protected by a hiding countermeasure utilizing complementary memory writes and correlated noise generation.

Table 2. Table of Training Dataset specifications

Dataset name	Encryption Algorithm	Number Of traces	Size of One trace
AES unprotected	AES-128bit	10,000	5,000 sample points
AES with hiding countermeasure	AES-128bit with hiding countermeasure	10,000	5,000 sample points
ASCAD	AES-128bit with masking	20,000	700 sample points

The masked AES set of power traces we used is part of the publically available ASCAD database [4]. We have performed the power trace acquisition for the other two power trace sets of the unprotected and hiding based countermeasures. To obtain the power traces we have used a Teledyne-LeCroy WaveRunner 640Zi oscilloscope, set at a sampling rate of 2.5GSPS. The target contained an MCU of the 32-bit ARM Cortex M3 family, operating at a clock frequency of 72 MHz. To capture the power traces we have connected a resistor of 1 Ohm in series between a GND pin of the MCU and the PCB ground and cut all other pins connected to the ground.

The specifications of each set of power traces are presented in Table 2. For both the unprotected and the hidden AES, we collected 10,000 traces. For the masked AES, we used the 20,000 traces available in the ASCAD database. For our MLP model, we implemented the model we discussed in section II. Noting again that our goal was to reconstruct the same model discussed in [7]. The only difference is we used the Keras library instead of Pytorch. In Fig. 2, we present the results of our DDLA. In each graph the horizontal axis contains the training number of epochs, while the vertical axis represents the accuracy of each trained network. Noting in deep learning terms, one epoch is when an entire training dataset is passed, only once, through the neural network during the training phase [2]. In DDLA, for each epoch, all the available traces (10k or 20k) are used to train the networks, resulting to the depicted accuracies. In Fig. 2a, we can see that the attack is capable to find the secret key after the first epoch, using the 10k traces. Fig. 2b shows that the masked AES is also leaking the key after the first epoch using the 20k traces. On the other hand, in Fig. 2c we see the results for the hiding countermeasure which computes each intermediate value using a fake key (the AES operation of which is on purpose leaking more than the one with secret key), in parallel with the computation based on the correct secret key. This countermeasure exposes after all the epochs using the 10k traces the fake key only and hides the secret key byte inside the remaining key byte hypotheses. While first order CPA attack works against the hiding countermeasure, the DDLA is not efficient against this countermeasure.

After obtaining these results, we decided to further investigate the hiding countermeasure since it was the only one which was still resilient using 10k traces. The goal of our second experiment was to obtain as many traces as necessary, so as to break the hiding countermeasure by performing a DDLA attack. Furthermore, for this set of power traces we will also present a first order CPA attack and its computed correlation coefficients for each sampling point of the oscilloscope. Noting for CPA we considered 5000 sample points. In Fig. 3, we see the results of the DDLA attack using 100k power traces which were needed for the attack to clearly distinguish the correct secret key from the remaining keys which were considered as noise. We can consider that the countermeasure is no longer secure when this happens, even though the highest accuracy is the one of the fake key, since eventually the attacker may understand the use of a fake/correct key scheme. Therefore, this countermeasure is compromised after training the network for 50 epochs using 100k traces. A CPA attack which is shown in Fig. 4 on the other hand needs at least 65k traces to compromise the countermeasure. Furthermore, we can also see that the CPA attack is capable just after 10k traces to include the correct secret key byte among the most likely choices of an attacker (who has understood the way the countermeasure operates).

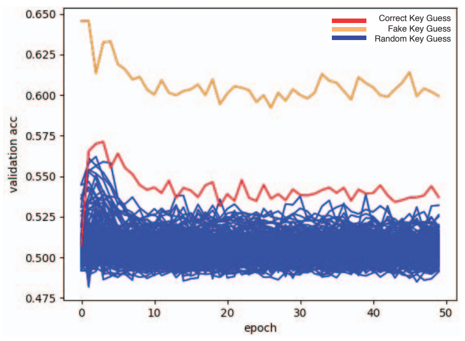


Figure 3. Accuracy graph of DDLA attack against fake key with 100,000 traces

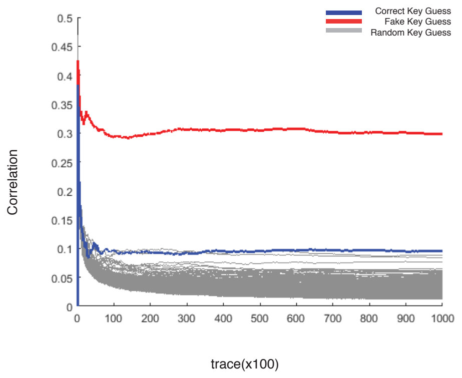


Figure 4. Correlation diagram of CPA attack against fake key with 100,000 traces

Comparing the time necessary for each attack, we needed approximately 14 hours in order to perform the DDLA attack and 12 hours to perform the first order CPA attack for 100k traces. Even though these durations seem to show that the two attacks take approximately the same amount of time to compromise the hiding countermeasure, we also need to take into account the different computational platforms used for each attack. In the case of DDLA we have used a computer equipped with a GeForce GTX 1080Ti Graphics card which is very efficient for Deep Learning applications. On the other hand, for performing the CPA attack we have used a computer with an Intel i7 processor without any graphics accelerator. Therefore, in order to fairly compare the durations of the attacks we would have to implement the CPA attack by taking advantage of the same graphics card accelerator, as for the DDLA attack.

IV. Conclusion

Our results add to the state of the art an evaluation of a hiding-based SCA countermeasure by means of DDLA and compare it with a classic CPA attack. In [6] the authors point out that a countermeasure which may offer high protection against first order CPA attacks, such as masking, may not be powerful against DDLA attacks. Our experiments show that a hiding countermeasure may provide higher protection against non-profiled deep learning side channel attacks. Therefore, in order to implement countermeasures which are resilient against modern non-profiling DDLA attacks we may need to use methodologies which can disturb their training procedure. One perspective is to further investigate if there exist disturbances we can induce into the power traces of an encryption algorithm which can disturb even more the training procedure of DDLA attacks. Furthermore, we plan to investigate countermeasures which can protect equally against DDLA and CPA attacks.

V. Acknowledgement

This work is supported by the French National Research Agency in the framework of the “Investissements d’avenir” program (ANR-15-IDEX-02).

VI. References

[1] Standaert, François-Xavier. "Introduction to side-channel attacks." Secure Integrated Circuits and Systems. Springer, Boston, MA, 2010. 27-42.
[2] L.Deng, “A tutorial survey of architectures, algorithms, and applications for deep learning,” APSIPA Transactions on Signal and Information Processing, vol. 3, 2014.
[3] H. Maghrebi, “Deep Learning based Side Channel Attacks in Practice,” Tech. Rep. 578, 2019.
[4] J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. Springer Science & Business Media, Mar. 2013. Google-Books-ID: fNaoCAAAQBAJ.
[5] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, “Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database,” IACR Cryptography ePrint Archives, p. 45, 2018.
[6] B. Timon, “Non-Profiled Deep Learning-Based Side-Channel Attacks,” Tech. Rep. 196, 2018.
[7] B. Timon, “Non-Profiled Deep Learning-based Side-Channel attacks with Sensitivity Analysis,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 107–131, Feb. 2019.
[8]Kamoun, Najeh, Lilian Bossuet, and Adel Ghazel. "Correlated power noise generator as a low cost DPA countermeasures to secure hardware AES cipher." 2009 3rd International Conference on Signals, Circuits and Systems (SCS). IEEE, 2009.
[9] E. Aerabi, A. Papadimitriou, and D. Hély, “On a side channel and fault attack concurrent countermeasure methodology for MCU-based byte- sliced cipher implementations,” 2019.
[10] A. Kamilaris and F. X. Prenafeta-Bold, “Deep learning in agriculture: A survey,” Computers and Electronics in Agriculture, vol. 147, pp. 70–90, Apr. 2018.
[11] E.Cagli, C.Dumas, and E.Prouff, “Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures,” in cryptographic Hardware and Embedded Systems CHES 2017 (W. Fischer and N. Homma, eds.), Lecture Notes in Computer Science, pp. 45–68, Springer International Publishing, 2017.
[12] S. Fanelli, M. D. Martino, and M. Protasi, “An efficient algorithm for the binary classification of patterns using MLP-networks,” in IEEE International Conference on Neural Networks, pp. 936–943 vol.2, Mar. 1993.