Performance Metric for Differential Deep Learning Analysis

Daehyeon Bae and Jaecheol Ha*
Hoseo University, Asan-si, Chungcheongnam-do, Korea
noeyheadb@gmail.com, jcha@hoseo.edu

Received: April 9, 2021; Accepted: May 16, 2021; Published: May 31, 2021

Abstract

In recent years, deep learning has been actively applied to the field of side-channel analysis, and has focused on profiling scenarios in particular. The advent of the Differential Deep Learning Analysis (DDLA) technique allows the advantages of deep learning to be utilized in non-profiling scenarios, where for a fixed key, only a limited number of power traces can be analyzed. However, in most DDLA-related studies, training metric graphs for estimating performance are only provided, without specific numerical value. In this case, there are several difficulties, such as performance comparison and implementation of automatic attack scripts. In this paper, we propose a novel performance metric, the Normalized Maximum Margin (NMM), which takes into account the statistical characteristics of training metric values, such as accuracy and loss value. In addition, we present three experiments to verify that the NMM can be effectively used for attack success decision, performance evaluation, and so on. These experimental results show that the NMM can objectively represent the performance of DDLA, and whether the attack was successful. Furthermore, we confirm that the NMM can also be used as a key distinguisher, which allows the key to be extracted in the real world, where the correct key is unknown.

Keywords: Hardware Security, Side-Channel Analysis, Differential Deep Learning Analysis, Normalized Maximum Margin

1 Introduction

The side-channel attack, first introduced by P. Kocher as a timing attack, has been actively studied in the field of cryptography and hardware security [12]. There are several types of side-channel attacks, such as the timing attack, power analysis attack and ElectroMagnetic (EM) analysis attack. The timing attack analyzes difference in execution time of the cryptographic device caused by cache memory or branch statements according to the processing data. On the other hand, the power and EM analysis attacks exploit the dependency between side-channel leakage, i.e., power consumption and EM signal, and intermediate data processed in the cryptographic device [16].

Power analysis attacks are classified into two types according to the attack scenario: profiling attack, or non-profiling attack. In the profiling attack scenario, the adversary must have a profiling device, which is similar to the victim device, and is utilized to profile the power consumption characteristics of the victim device. First, the adversary who has full control of the profiling device makes a power consumption profile for all secret key candidates. Then by comparing the power trace with the power consumption profile, the adversary can extract the secret key used in the victim device. In contrast to the profiling scenario, in the non-profiling scenario, the adversary can access only the victim device. The numerous side-channel leakages measured from the victim device with fixed key can be analyzed using

Journal of Internet Services and Information Security (JISIS), volume: 11, number: 2 (May 2021), pp. 22-33
DOI:*10.22667/JISIS.2021.05.31.022*

*Corresponding author: Division of Computer Eng., Hoseo University, Asan-si, Chungcheongnam-do, 31499, Korea, Tel: +82-41-540-5991

statistical methods, such as Differential Power Analysis (DPA) [13], and Correlation Power Analysis (CPA) [3].

With the growth of deep learning technique, the research efforts applying it to side-channel analysis have recently been increasing [10]. However, most of these studies have focused on the profiling attack scenario [11, 5, 15]. Meanwhile, in 2018, B. Timon showed that the deep learning technique can also be applied to the non-profiling scenario to extract the fixed secret key [18, 19]. The non-profiled deep learning based side-channel attack, the so-called Differential Deep Learning Analysis (DDLA), exploits the difference in training speed and degree, due to the consistency between traces and labels generated from each hypothetical key. That is, the training metrics, e.g., accuracy, loss value and average rank of the correct key, that are recorded when assuming the correct hypothetical key, are distinguishable, compared to when assuming the wrong keys.

In most of the DDLA studies, as well as in Refs. [18, 19, 6, 1, 21], the success decision or evaluation of performance tends to rely on individual perspectives. That is, they provide only a training metric graph for all key candidates, excluding precise numerical values. As a result, there are some difficulties, such as performance comparison and the implementation of automatic attack scripts. Even though J. Han et al. proposed a decision ratio that can determine the success of DDLA, it cannot be used for performance comparison because the statistical characteristics of training metric values are not considered [8].

In this paper, we propose a performance metric for DDLA named the Normalized Maximum Margin (NMM) to solve these difficulties. We show that the NMM can be effectively used as a performance metric through three experiments: 1) finding a key in a real environment where the correct key is unknown, 2) comparing performance, and 3) finding the minimum number of power traces for DDLA. The experimental results show that the NMM can objectively represent the performance of DDLA, and whether the attack was successful. Furthermore, we can adopt the NMM as a key distinguisher that allows the key to be recovered in the real world, where the correct key is unknown.

The structure of this paper is organized as follows: Sections 2 and 3 describe the underlying deep learning techniques and DDLA, respectively. Section 4 proposes the novel performance metric NMM to solve several problems. Section 5 describes 3 types of experiments we undertake to verify the efficiency of NMM: extracting the key in the real world in which the correct key is unknown, comparing the performance of DDLAs, and finding the minimum number of traces for a successful DDLA. Finally, Section 6 concludes this paper with the expected effect.

2 Deep learning for Side-Channel Analysis

Recent years have seen remarkable development of the deep learning technique, due to the appearance of big data and high-performance computing hardware, especially the Graphics Processing Unit (GPU). The deep learning technique is based on the mathematical model that performs learning and inference using artificial neural networks created by imitating biological neurons.

2.1 Artificial Neural Network

The Artificial Neural Network (ANN) is a fundamental component of most deep learning models. The ANN can be classified according to the detailed structure of the model and the feature extraction layer into the Multi-Layer Perceptron (MLP) [17], Convolutional Neural Network (CNN) [14], Recurrent Neural Network (RNN) [7], and so on. All neurons, so-called nodes, are connected to other neurons in the previous and next layers using weight values. The weight denotes the connection strength between the two neurons. The training of the deep learning model means updating these weights to infer a desired value using the gradient descent and error backward propagation algorithms. In other words, the training

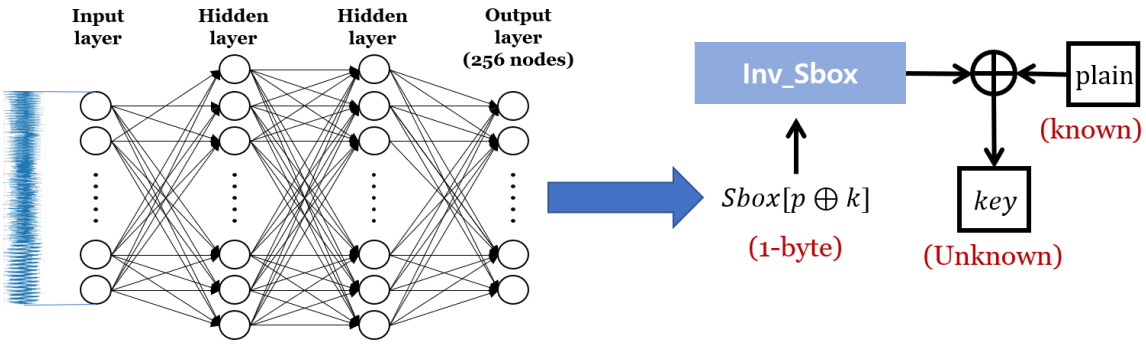


Figure 1: The simple MLP model for general profiling side-channel attacks.

model is a learning process to approximate a function that outputs a desired value according to an input data by updating the weight values. Each node receives and accumulates the multiplication values of the previous layer output and the weight. Then, the outputs of nodes are applied to the activation function, and entered to the next layer. The connection of these simple neurons enables complex inference of the deep learning model to perform complex inference.

2.2 Side-Channel Analysis Using MLP

As mentioned in Section 1, side-channel attacks using deep learning prior to DDLA have mostly focused on the profiling attack scenario. Therefore, in this subsection, we present a simple example of a profiling attack using MLP, which is the most basic artificial neural network model. In this case, each sample of the power trace is entered into input nodes, respectively. Then, an output value corresponding to the input power trace is inferred through the output nodes. Here, the output value (label) of the model is an Intermediate Value (IV) correlated with the power consumption trace. In other words, the model learns the power consumption leakage of the profiling and victim devices.

As an example, when attacking SubByte of AES encryption, the output value of the model is generally the result of SubByte. In this case, the secret key used in encryption can be extracted through an additional calculation on the output value of the model, as shown in Figure 1. Note that the model must be trained using a number of correct trace–IV pairs measured from the profiling device in advance.

3 Differential Deep Learning Analysis

DDLA allows the advantages of deep learning technique to be utilized for non-profiling attacks that can use only the limited number of power traces measured from the victim device. As a result, DDLA can outperform conventional non-profiling attacks, such as DPA and CPA, in terms of high-order and desynchronized power trace attacks. Concretely, DDLA can extract the secret key from the victim device executing the AES implementation, regardless of masking countermeasure, without any leakage combination pre-processing or knowledge about the detailed masking algorithm. Moreover, desynchronized power traces, which cannot be analyzed with conventional non-profiling attacks, are able to be analyzed without any pre-processing by using the CNN–DDLA model, due to the translation-invariance property of CNN [4].

Since DDLA is a non-profiling attack, it is performed by assuming all key candidates in the range of the key space (typically 1-byte). Whenever a hypothetical key is assumed, new labels of power traces are generated based on that key. The training and validation phases are then performed using a deep learning

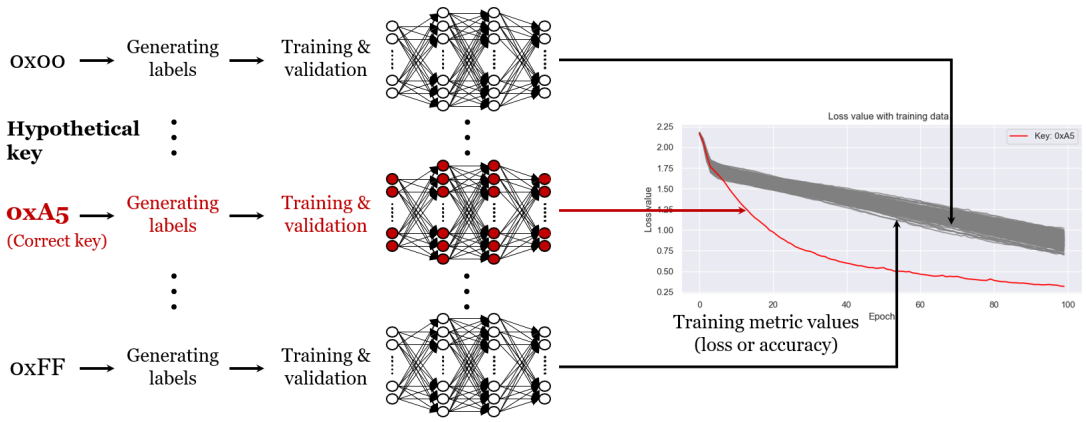


Figure 2: The overall process of DDLA to find a 1-byte fixed key.

model. On the way through all key candidates, if a correct key is assumed, the generated label and power trace are relatively correlated. As a consequence, the deep learning model that trained with labels with the correct hypothetical key will be better trained than the other candidates. That is, the deep learning model with the correct key has a relatively small loss value, and high accuracy. Thus, by observing the figure of training metric values, such as loss and accuracy, the correct key can be recovered, as shown in Figure 2.

References [18, 19] maintain that three types of label are available, i.e., Most Significant Bit (MSB), Least Significant Bit (LSB) and Hamming Weight (HW), excluding identity (ID) label. Here, the ID label means a 1-byte value by itself. All types of labels are applied to 1-byte intermediate values. However, in 2020, J. Han et al. showed that the ID label can be used in DDLA when not applying the one-hot-encoding method [8]. In their research, the number of output nodes is one, instead of 256. Since one-hot encoding does not consider whether the inferred value is far from or close to the correct label, the loss value of the misinferred output is always calculated as the same value. As a result, even though an ID label has a linear feature with neighboring ones, one-hot-encoding causes the feature of labels to disappear.

4 Performance metric for DDLA

4.1 The Need for Performance Metric

As mentioned in Section 1, the success decision or performance evaluation of DDLA in most research tends to depend on individual perspectives using visual graphs, excluding precise numerical values. In that case, problems may arise in the following situations:

- Comparison of attack performance
- Implementation of automatic attack scripts
  - Extracting the key in the real world where the correct key is unknown
  - Finding the minimum number of power traces required for a successful DDLA

When comparing attack performance, DDLAs cannot be exactly compared, due to various reasons, such as ambiguous difference of the y-axis scale, and the limitations of visual representation. In addition,

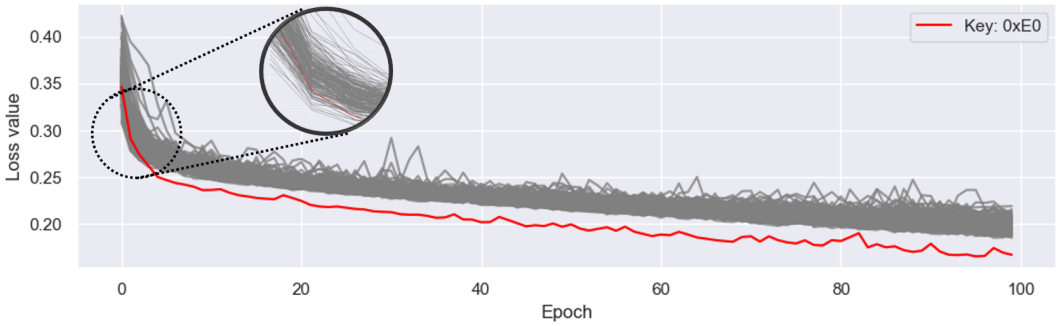


Figure 3: The loss metric values of DDLA.

the subjectivity of the researcher cannot be completely excluded. On the other hand, when writing automatic attack scripts, such as extracting the secret key and finding the minimum number of power traces to decide the success of DDLA, a precise numerical value, such as the Pearson correlation coefficient of the CPA, is required. To solve the above problems, we propose a novel performance metric for DDLA, named the NMM which deeply takes into account the statistical characteristics of training metric values, such as loss and accuracy.

4.2 The Normalized Maximum Margin (NMM)

The success decision or performance evaluation of DDLA depends on whether the training metric values for the correct key and wrong keys can be distinguished. To determine whether two groups of correct and wrong keys are separated, the distance between the two groups should simply be calculated, and this is called the margin. In the case of loss metric, the loss values calculated with the correct key must be smaller than all the metric values of the wrong keys, as shown in Figure 3. Therefore, the margin between the two groups should be calculated by $\mathrm{MIN}(mk'_e) - mk_e$, where mk_e is a metric value calculated with the guessed key, and mk'_es are the 255 values for the other keys, for all epochs e. Here, the k is guessed as the correct key, while the 255 k' values mean the other keys. Since the accuracy metric is the opposite of the loss metric, it is calculated by $mk_e - \mathrm{MAX}(mk'_e)$.

The performance cannot be perfectly determined only by the size of the margin between the two groups (the metric values of a guessed key, and the other keys). It is necessary to consider the statistical characteristics of training metric values. From what we have seen, the group of metric values, such as loss and accuracy, nearly follows a Gaussian distribution for each epoch. To show this, we visualize the distribution of the metric values used in Figure 3, after setting the average of the loss values to 0 according to each epoch, as shown in Figure 4. Consequently, we have confirmed that both the loss metric and the accuracy metric nearly follow a Gaussian distribution for each epoch. If the probability distributions are different from each other, it is difficult to compare the performance, even though the size of the margin is the same. As a result, the margin should be normalized by dividing it by the standard deviation (σ) for each epoch.

In some special situations where the number of power traces is small, the distribution can be slightly biased in one direction, such as the zoomed part of (b) in Figure 4. The distinction can be seen by comparison with the zoomed part of (a). Nevertheless, such phenomena can be ignored, because this bias does not significantly affect the probability distribution. In addition, the training metric values has the following additional characteristics:

- The training metric values along the epoch do not sharply fluctuate.

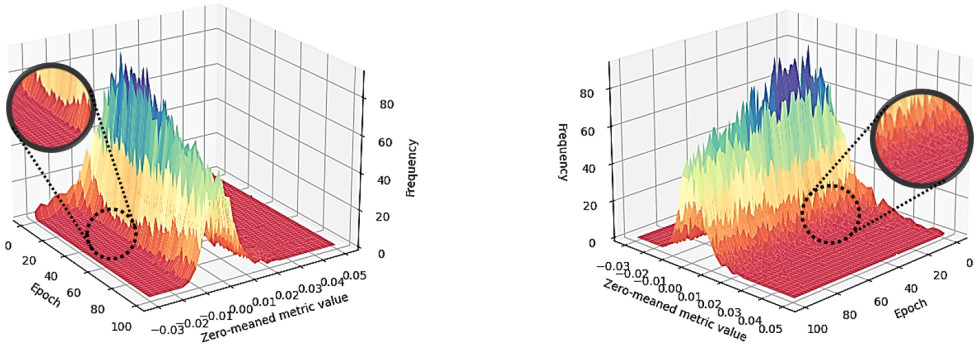


Figure 4: Distribution of zero-meaned loss values for each epoch.

- As the epoch increases, the size of the margin can be reduced due to overfitting and the universal approximation theorem [9].

For this reason, only a maximum margin should be calculated. Consequently, the Normalized Margin (NM) and the NMM can be calculated with the following formula Eq. (1)-(2), by selecting the maximum value of the normalized margin for a given hypothetical key k and all epochs $e \in \{1,2,...,E\}$:

$$\mathbf{NM_e} = \begin{pmatrix} \frac{mk_e - \mathbf{MAX}(mk'_e)}{\sigma(mk'_e)} & \text{if } M = acc \\ \\ \frac{\mathbf{MIN}(mk'_e) - mk_e}{\sigma(mk'_e)} & \text{if } M = loss \end{pmatrix} \tag{1}$$

$$\mathbf{NMM} = \mathbf{MAX}(\{\mathrm{NM}_1, \mathrm{NM}_2, ..., \mathrm{NM}_E\}) \tag{2}$$

Equation (1)-(2) can be calculated by a simple Python function using NumPy [20] package, as shown in Listing 1:

```python
def calculate_NMM(M_all, k, training_metric_type):
    # M_all : shape=(256, epoch)
    selector = np.arange(0, 256)
    M_k, M_not_k = M_all[k], M_all[selector != k]
    sigma = np.std(M_not_k, axis=0)
    if training_metric_type == "acc":
        NM = (M_k - np.max(M_not_k, axis=0)) / sigma
    elif training_metric_type == "loss":
        NM = (np.min(M_not_k, axis=0) - M_k) / sigma
    return np.max(NM)
```

Listing 1: A simple python script to calculate the NMM.

Now, the NMM can be calculated for all hypothetical keys using the loss values shown in Figure 3. Figure 5 shows the normalized margin for all keys, and NMM calculated with the correct key 0xE0. If the normalized margin values, calculated with the correct key, are in a positive range, the correct key is distinguishable; whereas, if these values are in a negative range, it is impossible to distinguish the correct

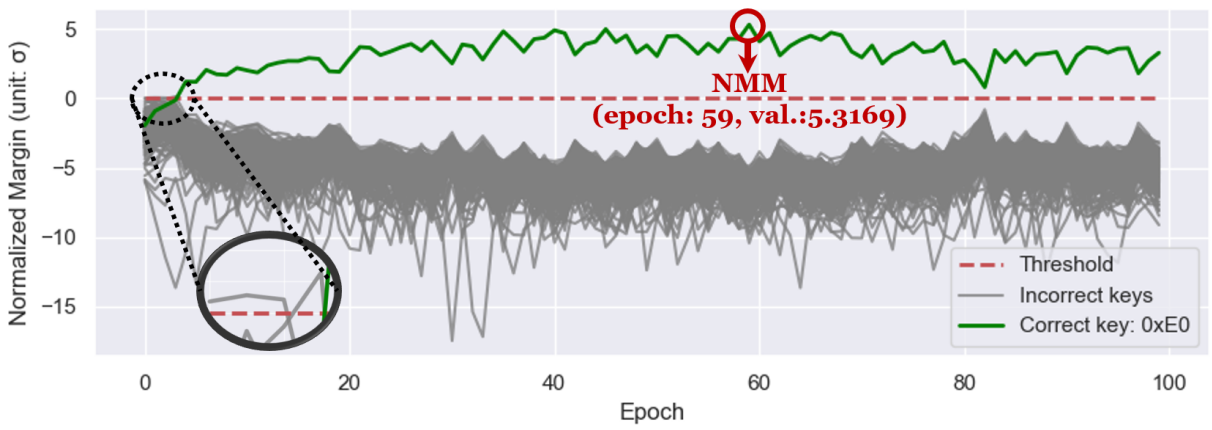


Figure 5: The normalized margin calculated with all key candidates and the NMM of the correct key 0xE0.

key. In fact, the normalized margin, calculated with the correct key 0xE0, is always in a positive range, except for epochs in the range (0–3), where the correct key cannot be distinguished, as shown in the zoomed part of Figures 3 and 5. The maximum of all normalized margin values is at the 59-epoch point of the correct key 0xE0. As a result, the maximum value 5.3169 is finally determined as the NMM.

5 Experimental Validation

To demonstrate the efficiency and usage of NMM, we show three types of experiments. First, we automatically find the secret key using NMM in the real world, where the correct key is unknown. Next, we compare the performance of the two DDLAs according to the labeling method; and finally, we find the minimum number of power traces required for successful DDLA using NMM.

5.1 Experimental Setup

All experiments in this paper are conducted using ASCAD database [2]. The ASCAD, introduced by Prouff et al. in 2018, is an open database in the field of side-channel analysis to enable the perfect reproducibility of experiments. They measured the power consumption of first-order masked AES in an 8-bit microcontroller. Their database includes the power traces and metadata, such as plain texts, cipher texts, keys, and mask values.

As an experimental example, we target the 3rd byte of the first-order masked SubByte conducted with a fixed secret key, i.e., 2nd-order DDLA. As described in Refs. [18, 19], the label of each trace is calculated without considering the mask value. That is, the label is calculated using only the result of $\mathrm{Sbox}\,(p\,[3] \oplus k\,[3])$, where p and k are the plain text and the secret key, respectively, without any mask values, such as message mask, or input and output mask of Sbox.

In our research, the simple MLP model was used for all DDLA experiments, i.e., MLP–DDLA. The model consists of 700 input nodes, 1 hidden layer composed of 30 nodes with ReLU function, and an output layer with Softmax function. We adopted mean squared error as a loss function, and Adam optimizer with a learning rate of 0.001. In addition, we used 20,000 power traces, and made the average of traces zero. For training phase, power traces were separated into a ratio of 0.75:0.25 for training and validation, respectively, and the batch size of 2,000 was used. For DDLA, we select the LSB label, unless anything is noted. That is, the LSB value of the Sbox result, $\mathrm{LSB}\,(\mathrm{Sbox}\,(p\,[3] \oplus k\,[3]))$, is actually used

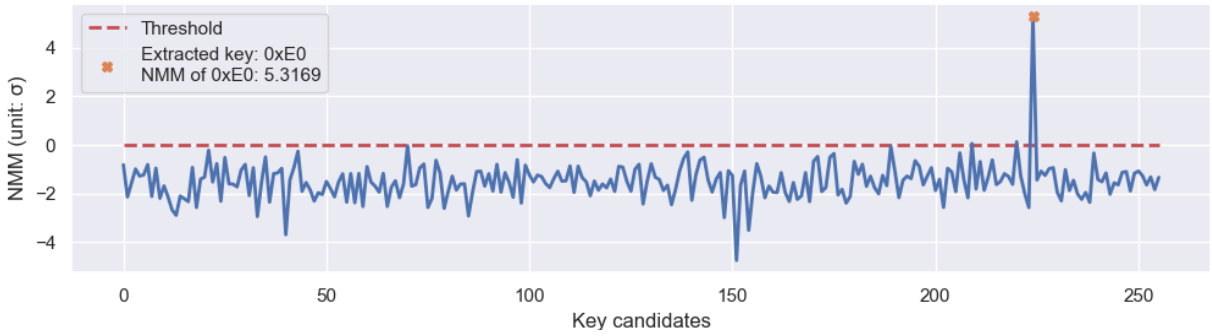


Figure 6: NMM calculated with all key candidates and the correct key 0xE0.

for training. Furthermore, all previous experiments performed in Figures 3-5 were also conducted in the above environments.

5.2 Case 1: Extracting the key in the Real-World

The NMM can be used to automatically extract the key from training metric values in the real world in which the correct key is unknown. In this case, the adversary can calculate the NMM for all key candidates, and precisely determine the one with the highest NMM value as the correct key as described in Algorithm 1. Figure 6 shows that the NMM (=5.3169) calculated at the correct key 0xE0 surpasses all other key candidates. As a result, the NMM can also be used as a key distinguisher, such as the Pearson correlation coefficient of CPA, to find the correct key automatically, even if the key is unknown.

Algorithm 1 Key extraction using the NMM (when the correct key is unknown)

Input: Training metric values **M** of size 256×epoch
Output: Key

1: **function calc_NMM**(m, k) ▷ Training metric values m, a given hypothetical key k
2: Calculate and return NMM for a given key k
3: NMMs ← [0.0, 0.0, ..., 0.0] ▷ Initialize an array of length 256
4: **for** $i = 0 \rightarrow 255$ **do**
5: NMMs[i] ← **calc_NMM**(M, i) ▷ NMM is calculated for all hypothetical keys (0-255).
6: **return** argmax(NMMs)

5.3 Case 2: Performance Comparison of DDLAs

To show an example of the performance evaluation using NMM, we compare the performance of the LSB label and the HW-based binary label, which is a new proposal we make in this paper. The HW-based binary is a simple label generation method that is presented to show an example of a performance comparison. This method generates 0 label if HW is less than 4, and 1 label if HW is greater than 4. Since the performance comparison is performed while the correct key is already known, the NMM should be calculated with only the correct key, not for all keys, as in the experiment of case 1. Note that the computation of NMM for all keys should be performed only in a real environment where the correct key is unknown.

According to the two label generation methods, Figure 7 shows the experimental results of case 2. Here, the (a) and (b) are loss values calculated with training data, while (c) is a normalized margin for the

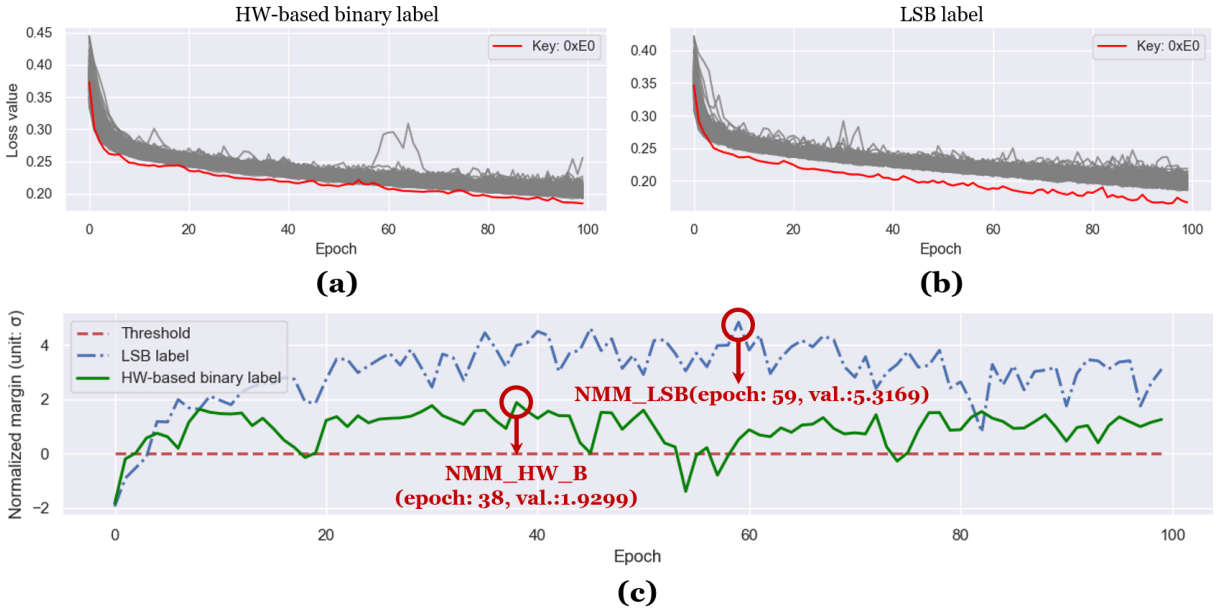


Figure 7: Training metric values and NMM according to label generation methods.

two label methods, respectively. Since (b) has a larger margin than (a), it can be guessed intuitively that (b) represents better performance. In fact, the NMMs of (a) and (b) are about 1.9 and 5.3, respectively. Therefore, the NMM objectively can reflect the reliability of DDLA, and be adopted for performance evaluation.

5.4 Case 3: Finding the Minimum Number of Traces Required for DDLA

When we find the minimum number of power traces required for DDLA, only the NMM calculated with the correct key should be used as the performance comparison in subsection 5.3. This is why the success decision of DDLA is determined based on the NMM of the correct key. That is, DDLA was successful only when the NMM of the correct key is in the positive range. Therefore, only the NMM of the correct key should be used in our analysis. Figure 8 shows the series of NMM calculated with the correct key 0xE0 according to the number of traces. In this case, we performed using power traces from (100 to 10,000) in increments of 100. Consequently, we confirmed that in our experimental environment, the minimum number of traces for successful DDLA is 2,800. Finally, the summary of three cases for the use of the NMM is shown in Table 1.

Table 1: Summary of three cases (1∼3) for the use of the NMM

	Correct key	NMM calculation	Success decision of DDLA
Case 1	Unknown	All hypothetical keys	*(Whether the extracted key is correct)*
Case 2	Known	Only a correct key	Whether the NMM is positive number
Case 3	Known	Only a correct key	Whether the NMM is positive number

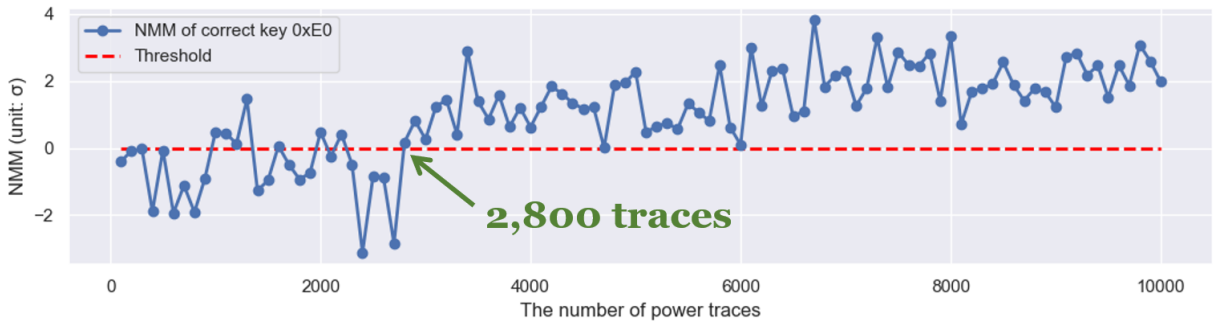


Figure 8: The minimum number of traces required for DDLA.

6 Conclusion

DDLA is an innovative analysis method that enables the advantage of deep learning algorithms to be utilized, even in the non-profiling scenario that only a limited number of power traces can be used. It is noteworthy that DDLA can perform high-order attacks without consideration of masking algorithm, and analyze desynchronized power traces. That is, DDLA can be effectively used to defeat random masking and jitter-based hiding countermeasures. However, in most DDLA-related studies, only training metric graphs are roughly provided as research results, because there is no obvious performance metric. Therefore, a numerical performance metric, such as the Pearson correlation coefficient of CPA, is needed. In this paper, we proposed the NMM, which is a novel performance metric for DDLA to solve several problems. We confirmed that the NMM can objectively represent the performance of DDLA, and indicate whether the attack was successful or not. Moreover, the NMM can be used as a key distinguisher that allows the secret key to be recovered in the real world, where the correct key is unknown. Consequently, the NMM can be used effectively, not only in the performance evaluation, but also for further DDLA research. Future work can include designing new deep learning models to minimize the number of traces required for DDLA and to defeat high-level countermeasures.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2020R1F1A1074358).

References

[1] A. Alipour, A. Papadimitriou, V. Beroulle, E. Aerabi, and D. Hély. On the performance of non-profiled differential deep learning attacks against an aes encryption algorithm protected using a correlated noise generation based hiding countermeasure. In *Proc. of the 25th Design, Automation & Test in Europe Conference & Exhibition (DATE'20), Grenoble, France*, pages 614–617. IEEE, June 2020.

[2] 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 Cryptology ePrint Archive*, 2018:53–98, 2018.

[3] E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In *Proc. of the 6th International workshop on cryptographic hardware and embedded systems (CHES'04), Cambridge, MA, USA*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag, August 2004.

[4] E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In *Proc. of the 19th International Conference on Cryptographic Hardware and Embedded*

Systems (CHES'17), Taipei, Taiwan, volume 10529 of *Lecture Notes in Computer Science*, pages 45–68. Springer-Verlag, September 2017.

[5] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-deepsca: Cross-device deep learning side channel attack. In *Proc. of the 56th Annual Design Automation Conference (DAC'19), Las Vegas, NV, USA*, pages 1–6. IEEE, August 2019.

[6] N.-T. Do, V.-P. Hoang, and V.-S. Doan. Performance analysis of non-profiled side channel attacks based on convolutional neural networks. In *Proc. of the 16th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'20), Ha Long, Vietnam*, pages 66–69. IEEE, December 2020.

[7] F. A. Gers, J. A. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451—-2471, October 2000.

[8] J.-S. Han, B.-Y. Sim, H.-S. Lim, J.-H. Kim, and D.-G. Han. Design of an effective deep learning-based non-profiling side-channel analysis model. *Journal of The Korea Institute of Information Security & Cryptology*, 30(6):1291–1300, December 2020.

[9] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[10] S. Jin, S. Kim, H. Kim, and S. Hong. Recent advances in deep learning-based side-channel analysis. *ETRI Journal*, 42(2):292–304, February 2020.

[11] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179, May 2019.

[12] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Proc. of the 16th Annual International Cryptology Conference (CRYPTO'96), Santa Barbara, California, USA*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag, August 1996.

[13] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Proc. of the 19th Annual international cryptology conference (CRYPTO'99), Santa Barbara, California, USA*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag, August 1999.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[15] H. Maghrebi. Deep learning based side channel attacks in practice. *IACR Cryptology ePrint Archive*, 2019:578–627, May 2019.

[16] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*. Springer Science & Business Media, 2007.

[17] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. *Parallel distributed processing*. MIT press Cambridge, MA, 1986.

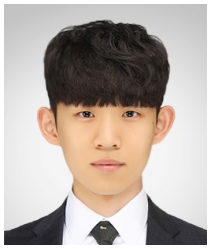
[18] B. Timon. Non-profiled deep learning-based side-channel attacks. *IACR Cryptology ePrint Archive*, 2018:196–229, February 2018.

[19] B. Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, February 2019.

[20] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, March 2011.

[21] Y.-S. Won, D.-G. Han, D. Jap, S. Bhasin, and J.-Y. Park. Non-profiled side-channel attack based on deep learning using picture trace. *IEEE Access*, 9:22480–22492, February 2021.

Author Biography



Daehyeon Bae received the BS degree in information security engineering from Hoseo University, Rep. of Korea, in 2021. He is currently working as a master's course student in the department of information and security at Hoseo University. His research interests include side-channel analysis, cryptography, hardware security and machine learning.

Jaecheol Ha received the BE, ME, and PhD in electronics engineering from Kyungpook National University, Rep. of Korea, in 1989, 1993, and 1998, respectively. He is currently a full professor of the division of computer engineering at Hoseo University, Asan-si, Rep. of Korea. During 1998 to 2006, he also worked as a professor in the department of information and communication at Korea Nazarene University, Cheonan, Korea. In 2014, he was a visiting researcher at Purdue University, USA. He is working as a vice-president in the Korea Institute of Information and Cryptography (KIISC). His research interests include post-quantum cryptography, network security, hardware security, and side-channel attacks.