Efficient Nonprofiled Side-Channel Attack Using Multi-Output Classification Neural Network

Van-Phuc Hoang, *Member, IEEE*, Ngoc-Tuan Do, and Van Sang Doan, *Member, IEEE*

***Abstract*—Differential deep learning analysis (DDLA) is the first deep-learning-based nonprofiled side-channel attack (SCA) on embedded systems. However, DDLA requires many training processes to distinguish the correct key. In this letter, we introduce a nonprofiled SCA technique using multi-output classification to mitigate the aforementioned issue. Specifically, a multi-output multilayer perceptron and a multi-output convolutional neural network are introduced against various SCA protected schemes, such as masking, noise generation, and trace de-synchronization countermeasures. The experimental results on different power side channel datasets have clarified that our model performs the attack up to 9–30 times faster than DDLA in the case of masking and de-synchronization countermeasures, respectively. In addition, regarding combined masking and noise generation countermeasure, our proposed model achieves a higher success rate of at least 20% in the cases of the standard deviation equal to 1.0 and 1.5.**

***Index Terms*—Deep learning (DL), embedded systems, multi-loss, multi-output, side-channel attacks (SCA).**

I. Introduction

SIDE-CHANNEL attacks (SCA) have become a serious threat to the cryptographic implementations on embedded systems. It has raised the awareness of the security research community to seek new techniques, which can be used to detect vulnerabilities [1] or counteract the SCA attacks [2]. However, researching new SCA attacks is critical to point out the potential threats. In this letter, we introduce a new SCA attack method using multi-output neural networks, which can reveal the secret key quickly in a nonprofiled context.

Our work is motivated by the previous work that was presented by Timon [3]. Accordingly, based on deep learning (DL) techniques, their proposal called differential DL analysis (DDLA) can reveal the secret key without any reference devices. However, DDLA requires the attacker repeatedly perform the training process to observe the training metrics, which are then used to determine the correct subkey byte. Recently, Kwon et al. [4] have investigated the aforementioned drawbacks of the DDLA technique and mitigated them by using a parallel neural network architecture. Kwon's work can be considered as a multilabel SCA approach as in [5]. Based on the binary cross entropy loss function, their models are optimized by minimizing a scalar loss value after training processes. Therefore, the accuracy metrics of key guesses are calculated by a custom function on each epoch, which separates the output and then matches the hypothesis values. The results of this function are then used to determine the correct key. Despite being a very fast attack technique, the parallel architecture requires a high memory usage. To mitigate the disadvantage of parallel architecture, the authors have introduced a shared-layer-based model that is reconstructed by the same DDLA model, except for the output layer. To the best of our knowledge, this is the first model that can predict 256 keys hypotheses in only one training process.

An alternative and often more effective approach in the DL domain is to develop a single neural network model that can learn multiple related tasks (i.e., outputs) at the same time, called multiple-output learning (MOL) [6]. From the point of view of the SCA domain, MOL is a promising technique that could increase the performance of the SCA evaluation process. In this letter, we propose a novel SCA attack based on multi-output classification, which can predict 256 values of the key hypothesis in a single training without any reference device. Specifically, a multi-output multilayer perceptron (MLP_{MO}) and a multi-output convolutional neural network (CNN_{MO}) are introduced. In which MLP_{MO} is used for breaking the boolean masking [7] and reducing the effect of noise-generation countermeasures [8], whereas CNN_{MO} is exploited to reveal the secret key from de-synchronization countermeasure [9]. Our approach exploits multiloss instead of using only binary cross entropy loss as in [4], [5]. Accordingly, a separate loss corresponding to each output is calculated in the training process. Therefore, the training metrics (loss and accuracy) of each key hypothesis can be achieved easily without any extra calculation. As a result, our proposal can perform attacks faster than a parallel network.

II. Proposed Multi-Output Deep Learning Model for Nonprofiled SCA

A. Data Preparation

To apply multi-output model in SCA domain, the data input (power traces) should be labeled by the values corresponding to the model outputs. We aim to predict all key hypotheses in one training process. Therefore, the number of network outputs is 256, which corresponds to 256 key guesses (0 to 255). To

Manuscript received 6 August 2022; revised 23 September 2022; accepted 30 September 2022. Date of publication 10 October 2022; date of current version 30 August 2023. This work was funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under Grant 102.02-2020.14. This manuscript was recommended for publication by O. Sinanoglu. *(Corresponding author: Van-Phuc Hoang.)*

The authors are with Institute of System Integration, Le Quy Don Technical University, Hanoi 100000, Vietnam, and also with the Faculty of Communications and Radar, Vietnam Naval Academy, Nha Trang 650000, Vietnam (e-mail: phuchv@lqdtu.edu.vn).

Digital Object Identifier 10.1109/LES.2022.3213443

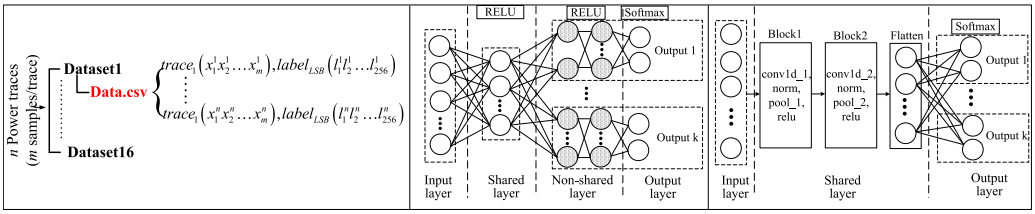

Fig. 1. Structure of reconstructed dataset and proposed multi-output models. (a) Multi-output dataset. (b) MLP_{MO} model. (c) CNN_{MO} model.

TABLE I
STRUCTURE OF RECONSTRUCTED DATASETS

Dataset	ASCAD		CW		Label
	Traces	Samples	Traces	Samples	
Dataset1	20000	700	-	-	LSB
Dataset2	20000	700	-	-	LSB(vector)
Dataset3	50000	700	-	-	LSB(vector)
Dataset4	-	-	10000	480	LSB
Dataset5	-	-	10000	480	LSB(vector)

benchmark our proposed architecture, we use the same LSB labeling technique as in previous works [3], [4], which is calculated by formula (1). As a result, the multi-output datasets used in this letter are constructed as depicted in Fig. 1(a)

$$l_j^i = \text{LSB}(\text{Sbox}(p_i \oplus k_j)) \tag{1}$$

where $p_i = \{1, n\}$ denotes the ith plaintext encrypted in AES-128 algorithm, n is the number of plaintexts. $k_j = \{0, 255\}$ is the key guess number j.

In order to evaluate the efficiency of the proposed models, we consider two SCA data, same as in [3], including ASCAD data [7] and the data captured from ChipWhisperer-lite (CW) board [10]. Regarding ASCAD data, the fixed key dataset is selected to perform attacks on the first-order masking countermeasure. The leakage model of ASCAD data is the output of the third Sbox with unknown mask values as described in [7]. In the case of CW data, we select 10 000 power traces with the size of 480 samples/trace, which correspond to the power consumption of the first Sbox output process. In addition, we simulate the de-synchronization countermeasure by using the same method as introduced in [4]. The structure of reconstructed datasets is shown in Table I.

B. MLP_{MO}

To solve the problem of DDLA, a feed-forward MO model based on the MLP architecture is proposed. As depicted in Fig. 1(b), the overall architecture of our proposed network consists of an input layer, a shared layer followed by k branches corresponding to k key hypotheses ($k = 256$). Each branch contains the same MLP architecture as MLP_{DDLA} (except the input layer) [3]. According to [11], we keep the number of layers and the number of nodes in each layer as the original MLP_{DDLA} model (hidden layer: 20×10-Relu, output layer: 2-Softmax). The input layer of the proposed model has the same size as the number of samples in the power trace.

The shared layer plays an important role in the proposed architecture. It can be utilized the max shared as in $\text{MLP}_{\text{max-shared}}$ [4]. However, using the same architecture of MLP_{DDLA} except output layer, their architecture only decreases the execution time without enhancing the success rate, especially in the case of noisy data. In contrast, our proposal aims to decrease the computation time as well as enhance the success rate. Therefore, we do not make the comparison to $\text{MLP}_{\text{max-shared}}$ in this work. In the case of the model without using the shared layer, the first hidden layer of each branch is fully connected to the input layer. Unlike MLP_{DDLA}, the network parameters of our model are updated for all key hypotheses in each iteration instead of updating for only one key guess as MLP_{DDLA} architecture.

Since the same structure is applied for all branches, the weights used for each branch are equivalent. Consequently, the loss function of the whole network is calculated as follows:

$$\mathcal{L}_{\text{total}} = \sum_{k=1}^{256} \gamma_k * \mathcal{L}^{[k]}(\theta) \tag{2}$$

where θ represents the set of all parameters of the model, γ_k is used as weighted factor of branch number kth and set as 1 for all branches (weights of each branch is equivalent), $\mathcal{L}^{[k]}$ denotes the loss results calculated for the kth branch. It is noted that the same loss function is used for all branches, which can be generally defined as follows:

$$\mathcal{L}^{[k]}(\theta) = -\frac{1}{N_s} \sum_{j=1}^{2} y_{\text{true}} \ln(z) \tag{3}$$

where y_{true} and z are the ground-truth and the predicted values, respectively. N_s denotes the number of training samples.

For successful training, the DL algorithm needs to find the optimal value to minimize the loss function $\mathcal{L}_{\text{total}}$. The network is trained in a series of iterations. In each iteration, the gradient of the loss function $\nabla\mathcal{L}_{\text{total}}$ is computed for updating the network. In our study, the popular optimization algorithm called adaptive moment estimation (ADAM) with default setting is employed to train the proposed model.

C. CNN_{MO}

Timon [3] has introduced a CNN model (CNN_{DDLA}) to reveal the secret key from the de-synchronized countermeasure. Similar to MLP_{DDLA}, their model needs to be trained repeatedly to determine the correct key. To mitigate this disadvantage, we introduce a multi-output model based on the CNN architecture (CNN_{MO}), which can break de-synchronized countermeasure in a single training process. Our CNN_{MO} consists of an input layer, share layers, and an MO layer, as

TABLE II
DL HYPERPARAMETERS OF PROPOSED MODELS

Model	MLP-MO	CNN-MO
Input size	700	480
Shared layer	0/50/ 200/400 -Relu	conv1d_1 (4 32 × 1 filters), pool_1 (2 × 1), norm, relu conv1d_2 (4 16 × 1 filters), pool_2 (4 × 1), norm, relu
Branch	256	256
Hidden layer/branch	20x10-Relu	0
Output layer/branch	2-Softmax	2-Softmax
Batch	1000	50
Initializing	He_uniform	

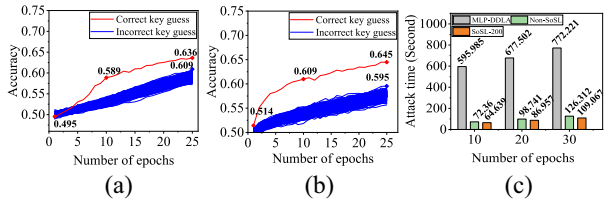

Fig. 2. Experimental results on masking countermeasure. (a) and (b) Accuracy of proposed model with and without shared layer, respectively. (c) Comparison of attack time.

depicted in Fig. 1(c). The shared layer consists of two blocks. Each block includes a 1-D convolutional (*conv1d*) layer, a average pooling (*pool*) layer, a batch normalization (*norm*) layer, and a rectified linear unit (*relu*) layer. These layers are placed in order as follows *conv1d-norm-pool-relu*. In the training phase, with the equivalent weights used for all branches, the loss function of the whole network is calculated by the formula (2), same as in MLP_{MO}.

The details of the proposed models are presented in Table II. It is worth noting that the MLP_{MO} is experimented with four variants of the shared layer, including a nonshared layer (0 node) and one shared layer of 50, 200, and 400 nodes, for comparison with MLP_{DDLA}. Regarding CNN_{MO}, to simplify, we choose the simplest model based on CNN_{DDLA} model, except for the output layer. It is worth noting that attacker is able to apply other hyperparameters to our proposed architecture to enhance the success rate of SCA attacks.

III. EXPERIMENTAL RESULTS

A. Masking

All experiments were performed by the Keras framework on a personal computer with Intel Core i5-9500 CPU, DDR4 24-GB memory. Our first experiment is performed on the Dataset2. We conduct various training processes with different sizes of the shared layer. Regarding the proposed network with a shared layer, we choose the size of the shared layer equal to 200 and denote it as SoSL-200 model. Our choice is motivated by the fact that it provides good results in terms of attack time and accuracy. The results of SoSL-200 using accuracy metrics are plotted in Fig. 2(a). Accordingly, an increasing trend of accuracy metrics of all key guesses can be seen. However, only the correct key (red) achieves stable and highest accuracy in most epochs (from epoch 5th) with a clear gap (0.636 and 0.609) at the end. It means that only the output of the correct key contributes a stable update to the appropriate branch; other outputs make the weights of others chaotic. In the case of nonshared layer model (Non-SoSL), we carry out further experiments using Dataset2. The results show that Non-SoSL can discriminate the correct key very early and more gap between correct and incorrect ones (0.645 and 0.595 at epoch 25th), as depicted in Fig. 2(b). The reported results have clarified the efficiency of our proposed methods for masking protected devices.

Next, we compare the efficiency of proposed networks with and without any shared layer (SoSL-200 and Non-SoSL, respectively) using Dataset2 and MLP_{DDLA} using Dataset1. Fig. 2(c) presents the execution time of attacks using selected models on different numbers of epochs. Overall, the execution time of DDLA is the highest in all cases. Interestingly, the execution time of our proposed network decreases dramatically about eight times and nine times (from 595.98 to 72.36 and 64.639 s), corresponding to Non-SoSL and SoSL-200 over ten epochs, respectively. Similar results can be seen in the case of training 30 epochs. The execution time of Non-SoSL and SoSL-200 decreased significantly about six and seven times (from 772.221 to 126.312 and 109.067) compared to MLP_{DDLA}, respectively. Compared to MLP_{PL}, the multiloss-based model in this work provides the training metric separately on each output. It leads to lower complexity than MLP_{PL} using a custom function. Indeed, MLP_{PL} reduces the execution time approximately 2.81 times (from 1950.9 to 693.8 s) compared to MLP_{DDLA} [4], whereas Non-SoSL decrease the execution time approximately 5.62 times (from 1025.2 to 182.1 s) compared to MLP_{DDLA}. These results have clarified our assumption and demonstrated that the proposed model outperforms both MLP_{DDLA} and MLP_{PL} in the computation time.

B. Noise-Generation Hiding Countermeasure

To simulate the noise-generation countermeasure, each sample of power traces of ASCAD dataset is added different levels of Gaussian noise as follows:

$$t_{\text{noise}}(i, m) = t(i, m) + \sigma \times \text{randn}(1, m) + \text{mean} \tag{4}$$

where *randn* returns a vector of numbers drawn from the standard normal distribution, σ and mean are the standard deviation and mean value (mean = 0), respectively. Consequently, the datasets called DatasetX-N1, DatasetX-N2, and DatasetX-N3 (correspond to $\sigma = 0.5$, 1.0, and 1.5, respectively) are reconstructed as the same technique as DatasetX, where X = 1, 2, 3. In this case, we consider MLP_{DDLA} is more reliable than $\text{MLP}_{\text{max-shared}}$ on noisy data. Therefore, only the comparison between Non-SoSL, SoSL-200, and MLP_{DDLA} is performed. By repeating the attacks using Non-SoSL, SoSL-200, and MLP_{DDLA} 50 times, we calculate the percentage of successful attacks over total attacks. The comparison of the success rate between MLP_{DDLA}, Non-SoSL, and SoSL-200 is shown in Fig. 3(a). Evidently, all models achieve good performance (100%) with the presence of a small level of additive noise ($\sigma = 0.5$). However, in the case of higher noise ($\sigma = 1.0$), the number of successful attacks drops from the

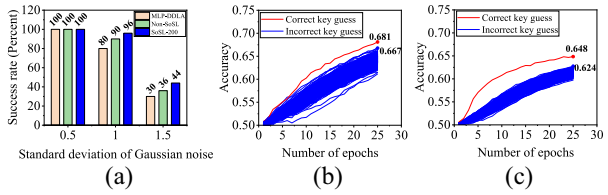

Fig. 3. Experimental results on combined masking and noise generation. (a) Success rate of MLP_{DDLA} and MLP_{MO} models on different levels of noise using 20 000 power traces. (b) SoSL-200 on 20 000 power traces, $\sigma = 1.5$. (c) SoSL-200 on 50 000 power traces, $\sigma = 1.5$.

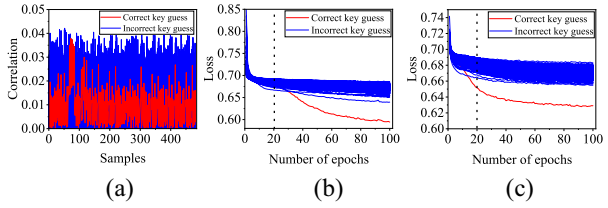

Fig. 4. Attack results on de-synchronized power traces using CPA, CNN_{DDLA}, and CNN_{MO}. (a) CPA. (b) CNN_{DDLA}. (c) CNN_{MO}.

100% to 80% and 90% corresponding to MLP_{DDLA} and SoSL model, respectively. Interestingly, the success rate of SoSL-200 only decreases slightly from 100% to 96%. A similar trend can be seen at the higher level of Gaussian noise ($\sigma = 1.5$). The success rate goes down significantly because the models provide poor discrimination, as illustrated in Fig. 3(b), in the case of SoSL-200 (0.681 and 0.667). However, our network still achieves better results than MLP_{DDLA} (44% and 36% compared to 30%). We perform further attacks using SoSL-200 on a larger size of dataset (Dataset3-N3). A clear gap between correct and incorrect keys (0.648 and 0.624) can be seen in Fig. 3(c). More interesting, the success rate is 100%. It indicates that by using the reasonable value of SoSL, the proposed network can mitigate the effect of the additive noise better. In addition, the attacker can perform DDLA attacks with reasonable epochs, a larger number of traces, or more hyperparameters, which will, in turn, improve the success rate.

C. De-Synchronized Traces

Finally, we consider other protected datasets containing de-synchronization power traces. To simulate this countermeasure, we randomly shift each power trace of Dataset4 and Dataset5 in a maximum of 20 samples. Consequently, two new datasets called Dataset4-sh20 and Dataset5-sh20 are used for training CNN_{DDLA} and CNN_{MO}, respectively. In this experiment, we used the loss metric to reveal the correct key. First, a CPA attack is performed on Dataset4 to validate the efficiency of this countermeasure. As depicted in Fig. 4(a), the secret key can not be revealed. In contrast, a good result in detecting the correct key can be seen in Fig. 4(b) and (c). These results demonstrate that the CNN model can break the de-synchronization countermeasure based on the translation-invariance property. However, the attack time of CNN_{MO} is shorter by approximately 30 times compared to CNN_{DDLA} (703.65 s compared to 20792.43 s). In addition, CNN_{MO} provides a clear distinction at very early epoch compared to that of CNN_{DDLA}. These results have clarified that the proposed model outperforms previous work in terms of attack time.

IV. Conclusion

In this letter, two multi-output models called MLP_{MO} and CNN_{MO} were introduced, which could perform SCA attacks better than the single-output approach on different protected schemes. Specifically, MLP_{MO} reduces the execution time up to nine times compared to the MLP_{DDLA} in the case of masking countermeasure. In addition, Non-SoSL model outperforms the state-of-the-art MLP_{PL} in term of execution time. Regarding hiding countermeasures, two common hiding countermeasures, such as noise-generation and de-synchronization, were used to evaluate the proposed models. The experimental results have shown that MLP_{MO} can reduce the effect of noise and achieves a higher success rate of at least 20% compared to MLP_{DDLA} in the case of $\sigma = 1.0$ and 1.5. The experimental results have also clarified that CNN_{MO} can break the de-synchronization countermeasure. More interesting, our proposed model performs SCA attacks faster, up to 30 times compared to CNN_{DDLA}. However, by using a fixed number of epochs, the attack results are not optimized. This problem will be investigated in the future work.

References

[1] P. Chakraborty, J. Cruz, C. Posada, S. Ray, and S. Bhunia, "HASTE: Software security analysis for timing attacks on clear hardware assumption," *IEEE Embedded Syst. Lett.*, vol. 14, no. 2, pp. 71–74, Jun. 2022.

[2] I. M. Delgado-Lozano, E. Tena-Sánchez, J. Núñez, and A. J. Acosta, "Gate-level design methodology for side-channel resistant logic styles using TFETs," *IEEE Embedded Syst. Lett.*, vol. 14, no. 2, pp. 99–102, Jun. 2022.

[3] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 107–131, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7387

[4] D. Kwon, S. Hong, and H. Kim, "Optimizing implementations of non-profiled deep learning-based side-channel attacks," *IEEE Access*, vol. 10, pp. 5957–5967, 2022.

[5] L. Zhang, X. Xing, J. Fan, Z. Wang, and S. Wang, "Multi-label deep learning based side channel attack," in *Proc. Asian Hardw. Orient. Security Trust Symp. (AsianHOST)*, Dec. 2019, pp. 1–6.

[6] D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong, and X. Shen, "Survey on multi-output learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2409–2429, Jul. 2020.

[7] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, "Study of deep learning techniques for side-channel analysis and introduction to ASCAD database," IACR Cryptol. ePrint Arch., Lyon, France, Rep. 2018/53, 2018.

[8] N. Kamoun, L. Bossuet, and A. Ghazel, "Correlated power noise generator as a low cost DPA countermeasures to secure hardware AES cipher," in *Proc. 3rd Int. Conf. Signals Circuits Syst. (SCS)*, Nov. 2009, pp. 1–6.

[9] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Cryptographic Hardware Embedded System (CHES)*, C. Clavier and K. Gaj, Eds. Berlin, Germany: Springer, 2009, pp. 156–170.

[10] C. O'Flynn and Z. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Constructive Side-Channel Analysis and Secure Design*. Cham, Switzerland: Springer Int., 2014, pp. 243–260.

[11] K. Kuroda, Y. Fukuda, K. Yoshida, and T. Fujino, "Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM," in *Proc. 5th Workshop Attacks Solutions Hardw. Security*, 2021, pp. 29–40.