Still Making Noise

Improving Deep-Learning-Based Side-Channel Analysis

Kim, Jaehun; Picek, Stjepan; Heuser, Annelie; Bhasin, Shivam; Hanjalic, Alan

DOI
10.1109/MDAT.2024.3510421

Publication date
2025

Document Version
Final published version

Published in
IEEE Design and Test

Still Making Noise: Improving Deep-Learning-Based Side-Channel Analysis

Jaehun Kim
Pandora
Oakland, CA 94612 USA

Stjepan Picek
Faculty of Science
Radboud University
6500 GL Nijmegen, The Netherlands

Delft University of Technology
2628CD Delft, The Netherlands

Annelie Heuser
CNRS, Inria, IRISA
Université de Rennes
35000 Rennes, France

Shivam Bhasin
Temasek Laboratories
Nanyang Technological University
Singapore 639798

National Integrated Center for Evaluation (NiCE)
Nanyang Technological University
Singapore 639798

Alan Hanjalic
Delft University of Technology
2628CD Delft, The Netherlands

Editor's notes:
Side-channel attacks have been undermining cryptosystems for almost three decades. Advances in machine learning techniques have shown great promise in improving the performance and efficiency of side-channel attacks, even on systems with countermeasures. This article provides a systematic approach to applying ML techniques for side-channel attacks.
—Jeyavijayan Rajendran, Texas A&M University, USA

Implementation attacks are those attacks that do not aim at the weakness of the algorithm but at the implementation. A common division of implementation attacks is into side-channel and fault injection attacks. Side-channel attacks (SCAs) are powerful, noninvasive attacks that exploit unintentional leakages of secret information from electronic devices. During cryptographic executions, the devices leak information through different side channels, such as power consumption, electromagnetic emission, sound, or timing. Depending on the application, SCAs revealing secret information can have serious consequences, which makes protecting them a mandatory step. Numerous attacks have been proposed over the years to evaluate the security of implementations. One traditional division of SCAs is into direct and two-stage (or profiling) attacks.

Digital Object Identifier 10.1109/MDAT.2024.3510421
Date of publication: 2 December 2024; date of current version: 22 January 2025.

Direct attacks assume that the adversary is collecting a (large) number of measurements from the device under attack and uses statistical techniques to infer secret information. Well-known examples of direct attacks are simple analysis and differential analysis. To break the target, direct attacks may require millions of measurements. Profiling attacks assume that the adversary possesses a clone device identical (similar) to the device under attack. The attacker uses that clone device to build the model of a device and attack the target device. The first phase, where the attacker builds a model, is commonly called the profiling phase. The phase when the attacker uses the model to break the device under attack is called the attack phase. While profiling attacks assume a more powerful attacker, if the model is well built, the attack can succeed with a small number of attack traces (even only one trace), making this kind of analysis the worst-case security analysis. Examples of profiling attacks are template attacks [2] and machine learning-based attacks. While a template attack is the most powerful from the information-theoretic perspective, it relies on several assumptions. For instance, a template attack assumes Gaussian distribution, and it has an unlimited number of profiling traces. Moreover, to make the attack efficient, one needs to conduct feature engineering and reduces the dimensionality of traces. Finally, breaking protected targets was still not possible.

Machine learning-based SCA (MLSCA) started to attract the attention of the SCA community around 2010. While the results could rival (or even outperform) template attacks, one still needed to conduct the feature engineering step, and protected targets were still impossible to break. Moreover, a new problem dimension appeared: the need to conduct hyperparameter tuning for machine learning algorithms. Deep learning-based SCA (DLSCA) appeared in 2016 [3] and immediately drew significant interest from the SCA community [4]. DLSCA has the advantage of requiring less feature engineering but at the expense of more complex hyperparameter tuning. Moreover, DLSCA also demonstrated that it is possible to break targets protected with masking and hiding countermeasures [5]. Such results made DLSCA also find its place in standardization efforts.[1]

[1]There is a recent document providing guidelines for machine learning-based SCA, which mostly concentrates on deep learning SCA; see Guidelines for Evaluating Machine-Learning Based SCA Resistance.

However, while DLSCA is commonly considered the most powerful profiling SCA approach, there are certain considerations one needs to take. First, DLSCA is commonly a black box, making the internal workings of an algorithm not explainable. This means that the evaluator will often not know why the deep learning-based attack has failed. Moreover, there will be no guidance on how to improve the security of a target in case it is broken by DLSCA. The second issue is connected with the hyperparameter tuning, which can be significant and computationally expensive [6], but can also lead to overfitting if not done properly. Overfitting represents the phenomenon when a deep neural network fits the training measurements perfectly but cannot generalize to previously unseen measurements. In this article, we will briefly discuss the common methodologies in DLSCA to fight overfitting, emphasizing data preprocessing and hyperparameter tuning.

Background

Side-channel analysis

SCA has represented a well-known threat since the 1990s and seminal works by P. Kocher. Today, such attacks are widely studied by researchers from both industry and academia. SCAs rely on the fact that implementations leak information in the physical world. The attacker (or security evaluator) can combine the physical observation of a specific internal state within computation and a hypothesis on the data being manipulated and recover the intermediate state processed by the device. By doing so, the attacker can obtain secret information, i.e., break the target's security.

Countermeasures

As already discussed, the attacker can obtain secret information by observing leakages from the device. Countermeasures are commonly used to make the attack more difficult. Countermeasures aim to break the statistical link between intermediate values and traces. There are two main categories of countermeasures for SCA: masking and hiding. In masking, a random value (mask) is generated to conceal every intermediate value. Common examples of masking are Boolean, multiplicative, additive, and affine. In hiding, one makes measurements appear random or constant. For that, it is possible to make changes in the

amplitude domain by, e.g., adding Gaussian noise to the signal or, in the time domain, by adding desynchronization or jitter.

Performance metrics

To evaluate the performance of SCA, it is common to use metrics such as key rank and guessing entropy. Let us assume that there are Q attack traces. The attack results in a key guessing vector g, ordered in the decreasing order of probability, where g_1 denotes the most likely and $g|\mathcal{K}|$ the least likely key candidate. The position of the correct key in the key guessing vector is the key rank. Guessing entropy is the average key rank where one repeats multiple independent evaluations and averages the results (to avoid the influence of randomness).

Deep learning-based side-channel analysis

A neural network is a mapping $f: \mathcal{X} \rightarrow \mathcal{Y}$, which takes a vector of input features $\mathrm{x} \in \mathcal{X}$ and predicts the output class (label) to which x belongs. In SCA, the label is derived from the key and input through a cryptographic function and a leakage model. To train the neural network representing f_θ, the learning algorithm Alg has access to the data set $\mathcal{D}$ drawn from the underlying distribution $\mathcal{X} \times \mathcal{Y}$. As we assume the supervised learning setting, $\mathcal{D}$ is partitioned into disjoint subsets commonly denoted as the training set $\mathcal{D}_{\mathrm{tr}}$, validation set $\mathcal{D}_{\mathrm{vl}}$, and test set $\mathcal{D}_{\mathrm{ts}}$. The size of the training data set equals N, the size of the validation set equals V, and the size of the test set equals Q.

There are three common DLSCA threat models. The first one, the most common, assumes that there is only one device, and the measurements for both profiling and attack are made on that device and artificially divided for profiling and attack steps. We note that most SCA works consider this setting since it gives the worst-case scenario for the target. The second scenario, commonly denoted by portability, assumes different devices and/or secret keys to be used in the analysis. Finally, the third scenario assumes nonprofiling DLSCA where the evaluator/attacker has access to only one device and no knowledge of keys. In this article, we will (as common in DLSCA) follow the first scenario.

During profiling (training), a neural network is tested based on how well it generalizes on the unseen examples from the validation set $\mathcal{D}_{\mathrm{vl}}$. Once it converges to an acceptable error rate, the training stops, and the neural network, along with its parameters, is stored as f_θ where f_θ represents the SCA profiling model. During the attack (test), the goal is to predict labels y based on previously unseen traces $\mathbf{x} \in \mathcal{D}_{\mathrm{ts}}$ and the trained model f_θ. We depict a common DLSCA process flow in Figure 1.

The raw data step considers the data acquisition where the evaluator/attacker gathers measurements, both labeled ones to be used during training and unlabeled ones to be used in the attack. In the case where no feature engineering is needed, raw data are input for a deep neural network.

The second step, data preprocessing, involves different actions that would allow better neural network training. For instance, common techniques are data normalization/standardization to reduce the effect of samples/features with different magnitudes. Moreover, it is common to use various data augmentation techniques that would reduce the effects of data imbalancedness but also of hiding countermeasures [7]. Data augmentation refers to various techniques to generate synthetic measurements that allow the machine learning algorithm to generalize better.

Kim et al. [1] follow a similar approach to reduce overfitting. More precisely, they applied a regularization technique by introducing noise to the traces in the training phase to reduce the overfitting of the model. Intuitively, we can see that by adding noise, we make the classification task more difficult, making it more challenging for a neural network to overfit (i.e., learn perfectly the function describing the training data only). More precisely, this simple approach works since it is known that adding noise to the input data is equivalent to adding a regularization term to the objective function, making training with noise equivalent to the Tikhonov regularization (which is a generalization of the L_2 regularization). Interestingly, a recent paper [8] investigated various regularization techniques and concluded that L_2 (and L_1) is the best-performing regularizer in DLSCA. In next section, we will discuss two use cases given in [1].

The third step, feature engineering, deals with techniques to select the most informative features or construct them based on the existing ones. This step was the central one for the success of "traditional" profiling attacks such as template attacks and MLSCA. Indeed, many results showed that if the evaluator/attacker selects/constructs good features, the

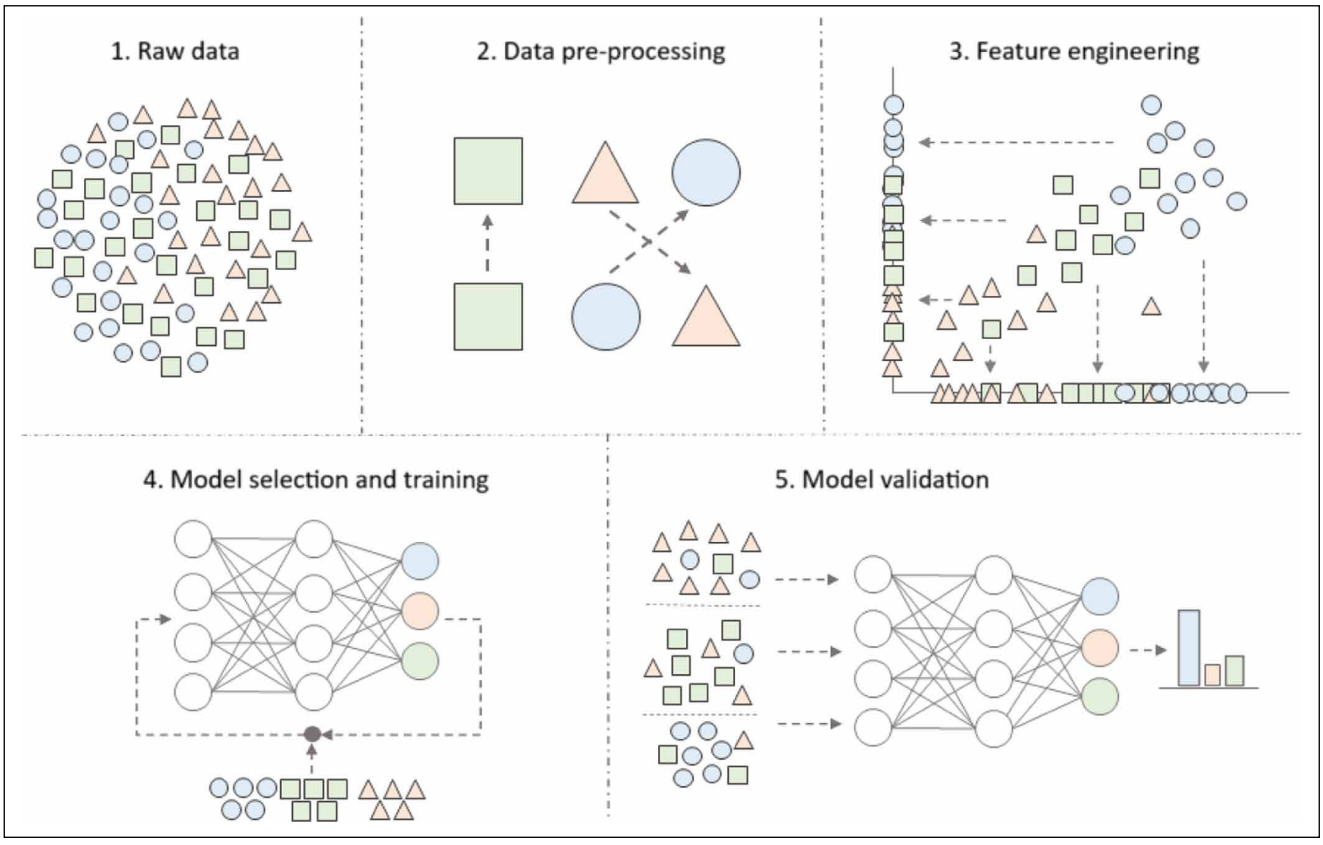

Figure 1. DLSCA process flow.

attack will work well. In DLSCA, feature engineering is less important since it is assumed that deep neural networks can select the most important features and use them for the attack. Still, it is not uncommon to at least select an interval of features[2] to make it a simpler analysis (see [1]). Still, we note that the most powerful attacks are currently made when no feature engineering is done, i.e., when feeding neural networks with raw data [5].

The next step is model selection and training. In this step, one first needs to select what machine (deep) learning algorithm to use. While DLSCA investigates diverse neural network types, two common options are to use either multilayer perceptron (MLP) or convolutional neural network (CNN) and to use them in a supervised manner. Once the architecture is selected based on the hyperparameters of the neural network type, it is necessary to tune it. Hyperparameter tuning represents one of the central points to achieving powerful deep learning performance, and SCA is no exception. Indeed, numerous SCA works explored designing a neural network that performs well and is (ideally) easy to tune. Each approach has certain advantages and drawbacks.

- Random/grid search. While random/grid search is easy to mount and can give excellent results (like in the current state-of-the-art), one still needs to define appropriate hyperparameter ranges that should be sufficiently small. In addition, one commonly needs to evaluate a large number of random models to improve the chances of obtaining good models.
- Advanced tuning techniques. Researchers in DLSCA used techniques such as genetic algorithms [3] and reinforcement learning [6] with very good success. Still, such techniques can be computationally expensive and have additional parameters to tune (shifting the problem from tuning the neural network hyperparameters to tuning the search technique parameters).
- Methodologies. Methodologies can provide a systematic way to build neural networks that

[2]By interval of features, we mean the part of the side-channel measurements containing leakage but also parts around it.

perform well in DLSCA. Unfortunately, it is difficult to design a methodology that is easy to follow and works for diverse targets/leakage models/neural network architectures.

Kim et al. [1] followed the third option and provided design principles for CNN-based DLSCA. In more detail, the authors used the strategy of a 1-D analog to the VGG architecture, characterized by the use of the minimal filter size and highly stacked convolution layers followed by several fully connected layers. Moreover, the authors provided the following key principles to follow:

- a minimal filter size (3);
- the convolution block continues until the spatial dimension is reduced to 1;
- the number of channels starts from a small number and keeps increasing (similar to VGG);
- using batch normalization and dropout (both used to prevent overfitting).

There are multiple approaches one can follow in this step to fight overfitting. For instance, it is possible to change the objective function to include regularization factors, and common examples are L_1 and L_2 penalties. Another option is to use a dropout that will temporarily remove random neurons along with their connections from the network during the training. Next, one can implement an early stopping strategy that will stop the training process once a number of epochs without improvement happen. Finally, batch normalization will also reduce overfitting by normalizing the inputs of each neural network layer.

Once the model is selected, i.e., the performance on the training set is satisfactory, the last step is to evaluate the model behavior on the previously unseen data set, a task done in the model validation step. As commonly done in MLSCA, to evaluate the attack performance, it is necessary to use SCA metrics such as key rank, success rate, and guessing entropy, since machine learning metrics, such as accuracy, can be misleading [9]. Still, even the model validation step can give different results from one run to another due to data selection (i.e., what SCA measurements are used in training/attack) and algorithmic randomness (due to weight initialization and optimization algorithm randomness). Kim et al. recognized the influence of various data splits and presented the results for each fold from the cross-validation and the average results [1]. Their results showed that certain folds can perform radically differently.

Experimental observations

Use cases

AES RD (Figure 2). This data set is a software AES implementation protected with the random delay countermeasure. Each measurement consists of 3,500 features. The data set has 50,000 traces. We take 20,000 traces for the training and 5,000 for validation. For this data set, we observe that the original and measurements with added noise indicate that the target is easy to break. Moreover, the results with the artificial noise are much better than without noise and require only around three traces to break the target. This is interesting as it was the first result showcasing that deep learning could come close to an optimal attack (i.e., requiring one attack trace) even if the target is protected.

ASCAD fixed key (Figure 3). This data set contains side-channel measurements consisting of 700 features representing side-channel leakages from processing the third S-Box output byte in the first AES

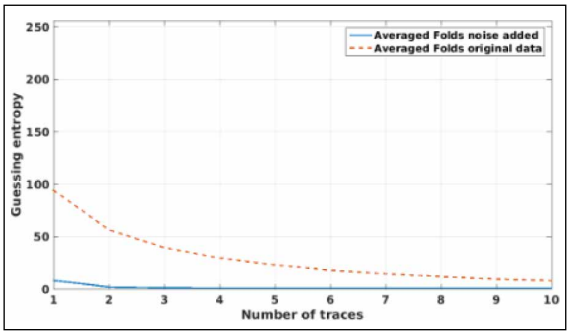

Figure 2. AES RD results.

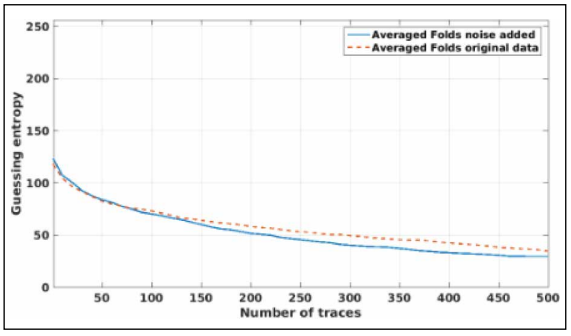

Figure 3. ASCAD fixed key results.

encryption round (the third S-box is the first masked one). The data set consists of 60,000 traces with the fixed key. We take 28,000 traces for the training and 7,000 for validation. We see that the neural network struggles for this data set regardless of whether we added noise. More precisely, the attack is unsuccessful as the final guessing entropy is high. Still, we can see a clear convergence with the increase in the number of attack traces, indicating that one can break the target, but we require more attack traces. Finally, with more attack traces, we observe a slight benefit of added noise, again indicating that regularization helps.

General findings

Kim et al. [1] also provided general observations, as listed in the following. We also provide a short discussion based on the state-of-the-art for each observation.

- It seems impossible to develop a single neural network with superior performance over all SCA scenarios. While this statement is rather generic, it makes sense to be true, especially when considering theoretical results such as the No Free Lunch theorem. While more recent results indicate that one architecture can be used for various targets, it is still far from having a single architecture for all scenarios. We believe the community should invest more effort into a middle path: the architectures that work well for different targets and multiple architectures to cover the variety of cases observed in modern SCA.
- It is possible to enhance the robustness of the model against the noise by adding Gaussian mean-free noise. Already, Kim et al. [1] clearly showed this to be true. Subsequent works in DLSCA and regularization further demonstrated that regularization is important to prevent overfitting and that L_2 regularization performs well.
- When conducting experiments with cross-validation, i.e., using a number of folds, it is possible that the performance over folds will vary extremely. While there are not many follow-up works considering cross-validation in DLSCA, it seems reasonable to expect that based on the selected training measurements, the obtained model will perform differently. Since the SCA community is well accustomed to reducing the attack phase randomness by using guessing entropy instead of key rank, it seems reasonable to adopt a set of procedures to reduce training data and algorithm randomness.
- Attacking a data set with a countermeasure can be easier than attacking an unprotected data set. In the end, it all depends on the countermeasure implemented and the quality of the data set [for instance, what is the signal-to-noise ratio (SNR)]. If the neural network architecture is well-performing, it is indeed possible to break both unprotected and protected data sets with a small number of attack traces. Still, for comparable SNR, we expect the protected data set to be somewhat more difficult than an unprotected one, but the difference can be small.

Perspectives and long-term impact

The work from Kim et al. [1] is one of the first works examining how to design neural networks for SCA (more precisely, how to select the neural network hyperparameters) that perform well. Moreover, it examines the influence of regularization techniques and discusses overfitting. This work has motivated several directions for further research.

While the first work on DLSCA used rather an elaborate strategy to design neural networks (more precisely, to decide on their hyperparameters) [3], it was still very much unclear can we do better (and potentially simpler) hyperparameter tuning. Hyperparameter tuning presents the core aspect of most DLSCA works in the next years. Zaid et al. [10] considered the problem from a different perspective and aimed to propose a design methodology. Nevertheless, while the results were good, following it for different data sets/targets is not easy. Rijsdijk et al. [6] proposed to use of reinforcement learning to find well-performing neural networks. While such an approach resulted in an excellent performance, more recent results indicated that even a random search on well-selected hyperparameter ranges could find excellent neural networks [5]. Finally, recent results indicate that one neural network architecture can provide good results for multiple implementations and algorithms [4]. This, in a way, confirms and supports the choice in [1] to have a single architecture for multiple data sets.

While many works discuss how data augmentation can help DLSCA, those works commonly

construct new, synthetic measurements by adding noise to the original measurements. This represents a fundamental difference from [1] since noise is added to the original measurements. As such, the approach from [1] should be considered preprocessing through regularization to prevent overfitting. Multiple works consider regularization, either explicitly or implicitly, by providing neural network architectures with certain regularization. Perin et al. [11] proposed an information bottleneck approach to monitor the training evolution. Robissout et al. [12] introduced a metric to evaluate the performance of a deep neural network in SCA during the profiling phase. This metric was used as a monitoring metric for early stopping. Rezaeezade and Batina [8] compared four different regularizer techniques and concluded that L_1 and L_2 are the most effective.

Finally, the new standard draft for minimal requirements for evaluating machine learning-based SCA resistance includes the work from Kim et al. [1] in the list of important literature on MLSCA.[3]

Today, DLSCA represents one of the most powerful options for evaluating the security of cryptographic algorithms. The results achieved in the last few years are remarkable and mainly stem from improved data preprocessing, feature engineering, and hyperparameter tuning. Each of those steps allows us to obtain well-performing deep neural networks capable of defeating diverse countermeasures while, at the same time, being relatively small in terms of the trainable parameters. However, despite that progress, there are still many questions that remain. One of the central ones is how to develop and tune neural network architecture that can be used either directly or after some fine-tuning (to, for instance, account for different input dimensions—side-channel trace lengths) for different targets. ■

Acknowledgments

This work was supported in part by the NWA Cybersecurity Call with project name PROACT under Grant NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO); and in part by the Netherlands Organization for Scientific Research NWO Project DISTANT under Grant CS.019.

[3]Guidelines for Evaluating Machine-Learning-Based SCA Resistance.

References

[1] J. Kim et al., "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, 2019.

[2] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. CHES*, Lecture Notes in Computer Science, vol. 2523. Redwood City, CA, USA: Springer, Aug. 2002, pp. 13–28.

[3] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Proc. 6th Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng. (SPACE)*, Lecture Notes in Computer Science, vol. 10076, C. Carlet, M. A. Hasan, and V. Saraswat, Eds., Cham, Switzerland: Springer, 2016, pp. 3–26.

[4] S. Picek et al., "SoK: Deep learning-based physical side-channel analysis," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–35, Feb. 2023, doi: 10.1145/3569577.

[5] G. Perin, L. Wu, and S. Picek, "Exploring feature selection scenarios for deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2022, pp. 828–861, Aug. 2022. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/9842

[6] J. Rijsdijk et al., "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, pp. 677–707, Jul. 2021. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8989

[7] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures—Profiling attacks without pre-processing," in *Proc. 19th Int. Conf. Cryptograph. Hardw. Embedded Syst. (CHES)*, Lecture Notes in Computer Science, vol. 10529, Taipei, Taiwan, W. Fischer and N. Homma, Eds., Cham, Switzerland: Springer, Sep. 2017, pp. 45–68.

[8] A. Rezaeezade and L. Batina, "Regularizers to the rescue: Fighting overfitting in deep learning-based side-channel analysis," *J. Cryptograph. Eng.*, vol. 14, pp. 609–629, Nov. 2024, doi: 10.1007/s13389-024-00361-5.

[9] S. Picek et al., "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, pp. 209–237, Nov. 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7339

[10] G. Zaid et al., "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 1, pp. 1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391

[11] G. Perin, I. Buhan, and S. Picek, "Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis," in *Constructive Side-Channel Analysis and Secure Design*, S. Bhasin and F. De Santis, Eds., Cham, Switzerland: Springer, 2021, pp. 53–81.

[12] D. Robissout, "Online performance evaluation of deep learning networks for profiled side-channel analysis," in *Proc. 11th Int. Workshop Constructive Side-Channel Anal. Secure Design (COSADE)*, Lecture Notes in Computer Science, vol. 12244, Lugano, Switzerland, G. M. Bertoni and F. Regazzoni, Eds., Cham, Switzerland: Springer, Apr. 2020, pp. 200–218, doi: 10.1007/978-3-030-68773-1_10.

Jaehun Kim is at Pandora, Oakland, CA 94612 USA. Kim has a PhD from the Delft University of Technology, Delft, The Netherlands.

Stjepan Picek is an associate professor at Radboud University, 6500 GL Nijmegen, The Netherlands. His research interests are security/cryptography, machine learning, and evolutionary computation.

Annelie Heuser is a permanent researcher at CNRS, IRISA, Université de Rennes, 35000 Rennes, France. Her main interests include the security of embedded devices, especially combined with side-channel information/attacks, machine and deep learning algorithms, and malware.

Shivam Bhasin is a senior research scientist and a program manager of cryptographic engineering at Nanyang Technological University (NTU), Singapore 639798. His research interests include embedded security and trusted computing. Bhasin has a master's from MINES Saint-Étienne, Saint-Étienne, France, and a PhD from Telecom ParisTech, Paris, France.

Alan Hanjalic is a professor of computer science, the head of the Multimedia Computing Group, and the head of the Intelligent Systems Department, Delft University of Technology (TU Delft), 2628CD Delft, The Netherlands. His research interests include multimedia information retrieval and recommender systems.

■ Direct questions and comments about this article to Stjepan Picek, Faculty of Science, Radboud University, 6500 GL Nijmegen, The Netherlands; Delft University of Technology, 2628CD Delft, The Netherlands; stjepan.picek@ru.nl.