Received: 28 March 2019 | Revised: 1 July 2019 | Accepted: 29 July 2019

DOI: 10.4218/etrij.2019-0163

ORIGINAL ARTICLE

ETRI Journal WILEY

Recent advances in deep learning-based side-channel analysis

Sunghyun Jin[1,2] | **Suhri Kim**[1,2] | **HeeSeok Kim**[3] | **Seokhie Hong**[1,2]

[1]School of Cyber Security, Korea University, Seoul, Rep. of Korea

[2]Center for Information Security Technologies, Institute of Cyber Security and Privacy, Korea University, Seoul, Rep. of Korea

[3]Department of Information Security, College of Science and Technology, Korea University, Sejong, Rep. of Korea

Correspondence
Seokhie Hong, School of Cyber Security, Korea University, Seoul, Rep. of Korea.
Email: shhong@korea.ac.kr

Funding information
This research was supported by the part of Military Crypto Research Center (UD170109ED) funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

As side-channel analysis and machine learning algorithms share the same objective of classifying data, numerous studies have been proposed for adapting machine learning to side-channel analysis. However, a drawback of machine learning algorithms is that their performance depends on human engineering. Therefore, recent studies in the field focus on exploiting deep learning algorithms, which can extract features automatically from data. In this study, we survey recent advances in deep learning-based side-channel analysis. In particular, we outline how deep learning is applied to side-channel analysis, based on deep learning architectures and application methods. Furthermore, we describe its properties when using different architectures and application methods. Finally, we discuss our perspective on future research directions in this field.

KEYWORDS
deep learning, machine learning, non-profiling attack, profiling attack, side-channel analysis

1 | INTRODUCTION

Experts estimate that the explosive growth of the Internet of Things (IoT) introduces not only a new dimension to information and communication technology but also difficult problems regarding security. As IoT devices tend to handle personal data, potential attacks must be considered when securing these devices. Side-channel analysis (SCA) is the most representative attack among potential attacks. SCA is a technique for unveiling secret information based on analyzing the data obtained from the execution of an algorithm, rather than the algorithm itself [1–3]. We shall refer to such data as leakages, and examples of leakages include time, power consumption, and electromagnetic emission.

The ultimate objective of SCA is to reveal the secret key of a device. This analysis is performed by exploiting the relations between leakages obtained during the execution of an algorithm and a secret key. An attacker collects leakages and analyzes them using specific models and appropriate metrics to determine the secret key. In this regard, SCA could be transformed into a classification problem. Machine learning models are mathematical functions that find patterns in data. As classifying the given data is an essential application of machine learning, machine learning techniques can be adapted to SCA. A drawback is that machine learning requires human engineering in some manner for better performance.

Deep learning is a subset of machine learning that has been studied since the mid-1990s. It can extract features automatically from input data. Deep learning did not gain considerable attention when it was first proposed, owing to insufficient performance caused by a lack of computing powers and training data. In the last decade, the gradual enhancement of computing power and the advent of big data have led to vast improvements in deep learning algorithms. Recent

works in the field have shown that deep learning algorithms achieve good performance in various applications, such as image recognition, speech recognition, and natural language processing. The performance of deep learning in these areas motivates its application to SCA.

Deep learning automatically learns features from data and generalizes the representation of data; this aspect will be of particular interest to a side-channel analyst. Accordingly, improvements in deep learning-based SCA (DLSCA) are expected. Indeed, recent studies have shown the feasibility of adapting deep learning properties to SCA for key retrieval and classification accuracy. These studies show that deep learning offers improved performance compared with classical side-channel attacks (SCAs). Hence, deep learning algorithms should be exploited in SCA. However, the performance improvement achieved by DLSCA compared with that of classical SCA is yet to be understood. Accordingly, a thorough survey of DLSCA is required.

In this article, we categorize and outline recent progress in DLSCA. First, we present deep learning-based profiling SCA. Deep learning-based profiling attacks can be primarily divided into two categories, namely power modeling using regression and key guessing through trace classification. Recent work has shown that side-channel protected algorithms can be analyzed when deep learning is used to classify power traces for key guessing. We demonstrate the characteristics of DLSCA based on the architecture used for the attack. Second, we introduce new approaches to utilize deep learning for SCA. Studies have been conducted on applying deep learning to non-profiling attacks, detecting side-channel leakages, and encoding side-channel leakages for non-profiling SCA. Each method exploits the learning process of deep learning for SCA, rather than focusing solely on the input and output functions. We illustrate the elements of deep learning used in SCA for each method. Finally, we discuss our perspective on the research to be pursued in the future.

Numerous studies on DLSCA have been conducted recently. However, we selected the articles based on the following criteria. This study mainly focuses on how deep learning is applied to SCA. In this regard, the articles covering the new method of applying deep learning to SCA were chosen first. We did not attempt to include articles dealing with deep learning only as part of machine learning (eg, performance comparison and data-related issues such as class imbalance [4–6]). Despite the importance of these studies, owing to our specific focus and the large volume of related publications, these articles were omitted. Independently, an introductory survey article on machine learning-based SCA was published during the review period of this article [7]. As a survey on machine learning-based SCA was presented systematically in [7], we recommend it to compensate for insufficient content.

The rest of this article is organized as follows: Section 2 describes the deep learning algorithm, which is an essential prerequisite for the subsequent survey in this study. Section 3 introduces attacks based on SCA, such as non-profiling attacks and profiling attacks, along with their countermeasures. Section 4 introduces deep learning-based SCA techniques, which are the focus of this study. Finally, Section 5 provides our comments on the future research direction.

2 | DEEP LEARNING

In this section, we briefly introduce the basics of deep learning algorithms, multilayer perceptron (MLP), and convolutional neural networks (CNN; cf. [8]).

2.1 | Basic deep learning

Deep learning was developed using artificial neural networks, which have been studied since the middle of the last century. Currently, deep learning is an active area of research owing to the advent of the big data era, the enhancement of computing power, and the development of deep learning algorithms. Numerous recent studies on deep learning have shown that deep learning techniques outperform other machine learning algorithms in the fields of image recognition, speech recognition, bioinformatics, natural language processing, and so on [9,10].

Deep learning is a subset of machine learning and is based on computational models composed of multiple processing layers that learn the representations of data with sequential abstraction [9]. In other words, by abstracting the representation of data sequentially for each layer, multiple processing layers constituting the deep neural network are trained for recognition or classification. This can be performed because a neural network having one or more hidden layers and a nonlinear function can approximate arbitrary Borel measurable functions based on the universal approximation theorem [8,11]. Furthermore, in contrast to the human-dependent feature selection of machine learning, deep learning architectures can extract features automatically, resulting in more accurate machine learning models.

There are several approaches to deep learning: supervised learning that learns using data with labels, unsupervised learning that learns using data without labels, and semi-supervised learning that learns using data with and without labels. In this study, we focus on deep learning architectures trained exclusively using a supervised approach.

The datasets used for deep learning are classified into three types: training, validation, and test datasets. The training dataset is a set of data used in the training phase, and the test dataset is a set of data analyzed using the trained model. The validation dataset is a set of data with labels used during the training phase to determine whether the training has been

performed correctly as the test dataset is not used during training. Indeed, the datasets do not have common data.

Deep learning architectures are expressed in (1). A deep neural network consists of an input layer, a series of hidden layers, and an output layer. The input layer is an identity function I in (1). The hidden layer is a composite function of a linear function and a nonlinear function. A hidden layer can be expressed as $A_i \circ \lambda_i$, where A_i is called an activation function. The deep learning architecture can contain one or more hidden layers as in (1). The sigmoid and rectified linear unit (ReLU) functions are representative examples of the activation function of the hidden layer. The neural network abstracts the representations of data through each hidden layer sequentially. The output layer is the function S in (1), and an identity function or softmax function is commonly used as an output layer. The softmax function is defined in (2). The softmax function (or normalized exponential function) has the effect of normalizing the output nodes so that the sum of all output nodes is equal to one while maintaining the magnitude of each node. That is, the softmax function creates properties similar to a probability density function.

$$\hat{f} = S \circ A_n \circ \lambda_n \circ A_{n-1} \circ \lambda_{n-1} \circ A_{n-2} \circ \cdots \circ A_1 \circ \lambda_1 \circ I \tag{1}$$

where λ_i represents linear functions and A_i represents nonlinear functions.

$$s(\vec{x})[i] = \frac{e^{\vec{x}[i]}}{\sum_j e^{\vec{x}[j]}}. \tag{2}$$

Learning or training in deep learning can be interpreted as a process of changing the weights of linear functions λ_i such that the neural network can abstract the representation of the data. The process of changing the weights is the same as solving an optimization problem using a gradient descent optimization algorithm and a backpropagation algorithm. The weights are changed to decrease an error, defined as the value of a loss function (or error function, cost function, or objective function).* The mean squared error (MSE) and cross-entropy are commonly used as cost functions [9]. Training proceeds as follows. First, the output values are calculated through the neural network in (1). Then, the loss function measures the difference between the output value and the label, where the output value refers to the value calculated from the actual input value, and the label refers to the expected output value. Subsequently, the gradient is obtained from an output of the loss function and is used to adjust each weight. This process is expressed in (3). Various optimizers, such as RMSprop and Adam algorithms, can be used to determine the optimal weights. They adjust the weights to the gradients with adaptive learning rates [9,12,13].

$$\hat{\theta} = \arg\min_{\theta} (\text{loss}(f(D_i; \theta), L_i)) \quad \text{for all } i \in I \tag{3}$$

where D_i represents data and L_i represents the label corresponding to D_i.

In the deep learning architecture, the number of hidden layers, the number of nodes in each hidden layer, the activation function, and the optimizer algorithm are called hyperparameters. Hyperparameters are non-trainable parameters that can be used to control the behavior of the deep learning algorithm. There are various methods for optimizing hyperparameters, such as a random search, grid search, and a search using an evolutionary algorithm [14,15].

There are two typical difficult cases in the training phase. First, if the training dataset and validation dataset are small, the loss function is likely to be biased. In this case, the validation of hyperparameters will be illegitimate. A k-fold cross-validation technique can be used to reduce the bias caused by small datasets. Second, when the deep learning architecture is trained, its performance can be improved only with regard to the training dataset and not the validation dataset. This phenomenon is called overfitting, which occurs when the weights of the neural network are overtrained for the training dataset. Techniques such as weight decay [16,17], dropout [18], batch normalization [19], and data augmentation [20] are sometimes applied to avoid overfitting. These techniques also improve the learning performance of a neural network.

2.2 | Multilayer perceptron

An MLP is one of the most common forms of a neural network. It is also called a feedforward neural network and has the form shown in Figure 1.

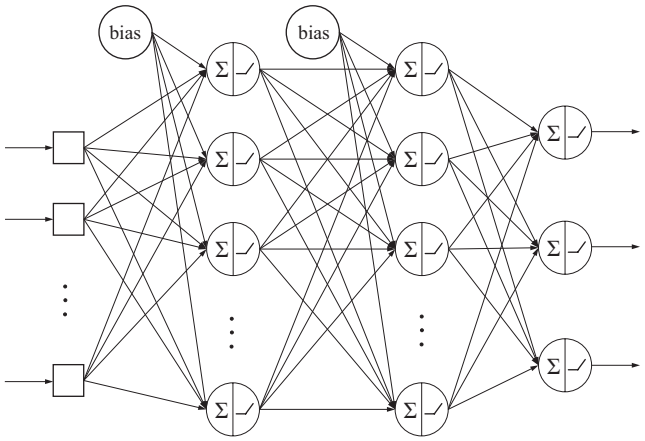

FIGURE 1 Multilayer perceptron architecture

*All the terms are not the same and have different meanings in some machine learning publications. In this article, we consider all the terms to have similar meaning.

In MLP, the linear functions λ_i in (1) take all the nodes in the previous layer as input, and the dot products over each weight of λ_i are then computed. Owing to this operation, the layers of an MLP are called fully connected layers. The output values of the dot products are the input to the activation function A_i. This is expressed in (4). The MLP is a neural network represented by the superposition of one or more operations with the form given in (4). The output of the MLP corresponds to the output of the last operation. Depending on the learning objectives, a softmax function can be applied:

$$X_n = A_n\left(W_n X_{n-1} + b_n\right). \tag{4}$$

Although the MLP is the most common deep learning architecture, its performance decreases with distorted data. The reason for this is that the output value is determined based on the numerical value of the input data, without considering the data topology.

2.3 | Convolutional neural network

The prototype for the current CNN was proposed in the form of a neocognitron by Kunihiko Fukushima, based on the work on the visual cortical structure of animals by David H. Hubel and Torsten Wiesel [21,22]. In 1998, Yann LeCun and others proposed LeNet for handwriting recognition. LeNet is now considered to represent the origin of modern CNN architectures. LeNet combines the concepts of shared weights, subsampling, and backpropagation algorithms [23,24]. AlexNet is another example of a CNN structure with good results [25]. When these CNN structures outperformed the traditional machine learning algorithms in visual recognition, the CNN was applied more broadly to various fields. Indeed, the CNN continues to gain popularity because of its high performance.

As stated previously, the performance of the MLP degrades when a distortion occurs, as it handles numerical values without considering the topology of the input data. As the CNN uses properties such as shared weights and subsampling, it maintains the invariance of the representations of data even when distortion occurs.

The CNN structure can be depicted as in Figure 2. A CNN is an architecture formed by superimposing convolutional layers and pooling layers in front of fully connected layers. The convolutional layer is a specialized kind of linear operation, and it operates as shown in Figure 3. All the values for a feature map [9] are obtained through a convolutional filter. That is, all the values of the feature map are calculated using the shared weight. Then, all the values of the feature maps enter the activation function directly. The pooling layer is usually placed after the convolutional layer and activation function. The primary function of the pooling layer is to reduce the size of the feature map extracted from the convolutional layer. The pooling layer can be viewed as a resampling method, and Max pooling and average pooling are mainly used in the pooling layer. Max pooling reduces the size of the feature map by selecting the maximum value of the target range, whereas average pooling selects the average value of the target range. In the CNN architecture, the superposition of convolutional layers and pooling layers plays the role of extracting features.

3 | SIDE-CHANNEL ANALYSIS

Although the cryptographic algorithm has been proven to be secure, naive implementation can leak the secret information through side-channel leakages such as time, power consumption, and electromagnetic emissions. These unintended leakages are due to the physical properties of an electronic device. Such physically observable phenomena are determined by the operation and the data being processed. In this article, we refer to the "sampled leakage" as a trace or waveform. A captured trace contains features that reflect the operation of a device and the processed data. SCA is any attack using such phenomena from implementation. Since its introduction in 1996 by Kocher [1], SCA has been considered a practical and powerful attack. Although many factors support this consideration, a particular factor is that it is possible to adopt the divide-and-conquer approach. This approach reduces computation complexity

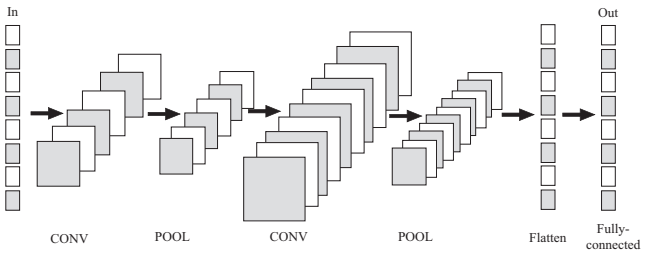

FIGURE 2 Convolutional neural networks architecture

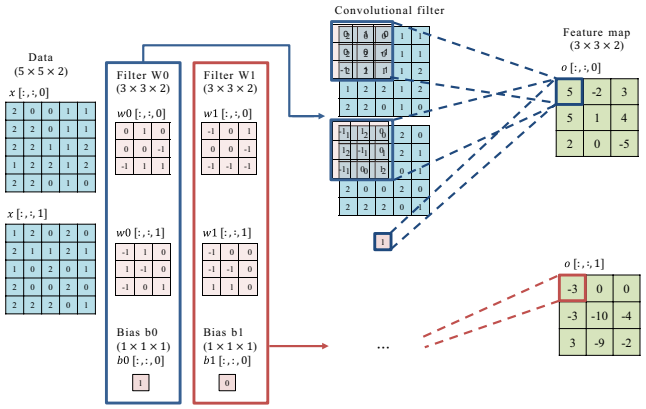

FIGURE 3 Example of a convolutional filter operation

remarkably compared with theoretical attacks against cryptographic algorithms.

SCA is categorized into non-profiling and profiling attacks, according to the environment provided to an attacker. In this section, we explain the non-profiling attack, profiling attack, SCA countermeasures, and issues related to advanced attacks briefly.

3.1 | Non-profiling attack

Non-profiling attacks assume that an attacker can only acquire traces from the target equipment. Depending on the number of traces required for the attack, non-profiling attacks are divided into two types. The first type guesses the secret information with one or only a few traces. Simple power analysis (SPA) [1,2] is a representative attack of the first type. SPA can deduce secret information from one or only a few traces obtained during the execution of an algorithm.

The second type guesses secret information through statistical analysis. Differential power analysis (DPA) [2], correlation power analysis (CPA) [26], and mutual information analysis [27] are examples of the second type. These attacks calculate hypothetical leakages for each possible key using power models such as Hamming weight and Hamming distance. Then, the key is inferred through a comparison between actual leakages and hypothetical leakages. For this comparison, statistical methods are used, such as a calculation of the difference of the mean between groups, the correlation coefficient, and cross-entropy. As statistical methods are used for these analyses, this type of attack exploits many traces. Especially in these kinds of attacks, trace alignment is a necessary step prior to attack [3,28,29].

3.2 | Profiling attack

Profiling attacks assume that an attacker has a programmable device identical to the target device. This programmable device is called a profiling device. An attacker can use this device to learn and exploit the physical properties of the profiling device to extract the secret information of the target device. Profiling attacks include template attack [30] and stochastic model [31].

Template attacks extract the information corresponding to the attack point with the profiling equipment. The subtraces of traces corresponding to the attack point form a certain distribution. Template attack assumes that such subtraces for a datum follow a multivariate Gaussian distribution. The attacker estimates the mean and covariance matrix of the subtraces for each possible intermediate data group. This process is called the profiling phase. After the profiling phase, an attacker infers data from the trace obtained from the target device through the maximum likelihood approach. This step is called the attack phase. The attacker can infer the key from the data obtained in the attack phase. Similar to non-profiling attack, it is necessary to align traces. In addition, as the number of attack points used increases, the complexity increases exponentially. To mitigate such problems, methods applying dimension reduction such as principal component analysis (PCA) and linear discriminant analysis (LDA) have been proposed [32,33]. Furthermore, to avoid numerical obstacles such as the inverse of the covariance matrix, the pooled covariance matrix has been proposed to be used for all intermediate values [34].

3.3 | Countermeasures

Side-channel analysis countermeasures are categorized into masking and hiding [3]. First, masking countermeasures render statistical analysis impossible by randomizing the intermediate values [35,36]. To make intermediate values appear random, actual random values are selected from a uniform distribution and are then combined with the intermediate values. Random values conceal all the intermediate values during cryptographic operations. The random values are then removed during the last step to produce the intended output. Second, hiding countermeasures reduce the signal-to-noise ratio (SNR) to make leakages unrelated to data or operations. They are realized through misalignment by injecting random jitters and a change in internal operation order [37–39]. An alternative method is producing constant waveforms regardless of the processed data using hardware dual-rail or software encoding techniques [40–43]. Just as naive implementations can be vulnerable to SCA, side-channel vulnerabilities can occur in the implementation of SCA countermeasures. Hence, realizations of SCA countermeasures must be implemented and evaluated through methods such as test vector leakage assessment (TLVA) [44].

Despite these countermeasures, SCA attackers can attack using advanced methods with more complexity. Second-order and higher order differential power analyses have been proposed to defeat masking countermeasures. Second-order and higher-order power analyses are methods based on the property that the actual leakages corresponding to the intermediate value can be extracted by combining the samples of the trace relating to the intermediate value [45,46]. However, these second-order and higher order attacks require finding points to be combined, which involves human engineering, that is the performance of the attack depends on the maturity of the attacker. Moreover, as noise is included in the traces, an attacker cannot obtain 100% of the actual leakage corresponding to the intermediate value. Furthermore, noise exponentially increases the number of traces required for an attack. To perform SCA to defeat hiding countermeasures, increasing the SNR is

necessary. The SNR can be improved through pre-processing steps such as noise reduction, alignment, and superimposition of the waveforms [3,28,29,47–49].

4 | DEEP LEARNING-BASED SIDE-CHANNEL ANALYSIS

Leakage pre-processing is essential in SCA, as sampled leakage is noisy and statistical methods must be applied to multiple traces. Misaligned traces or noise in the collected traces decrease the correlation between the traces and the processed data of the device. Therefore, noise reduction, alignment, and dimension reduction must be performed prior to SCA. Moreover, points of interest (POI)—significant features in the trace—must be selected. In short, pre-processing and POI selection are the most critical steps that affect the performance of SCA. Similarly, SCA using machine learning techniques, such as *k*-means clustering, support vector machine, and random forest, also requires pre-processing and POI selection. As pre-processing and POI selection require human engineering, they significantly impact the results [50–56].

By contrast, deep learning techniques select features automatically from data. Therefore, when deep learning is applied to SCA, it is less affected by pre-processing and POI selection. Indeed, studies have shown that, when deep learning is adopted, it is possible to analyze data without the steps required by classical SCA and machine learning-based SCA.

The general learning strategy for DLSCA is shown in Figure 4. The traces are used as input data, and the intended output values that an attacker wishes to extract from the trace are used as the labels. The labels can be viewed as output values of the Sbox or the Hamming weight of the Sbox output value. The form of the label changes depending on the learning objective. In the case of classification, as the Sbox output values or their Hamming weights are categorical data, they can be converted by one-hot encoding (one-hot code) and used as labels [9]. In the case of regression, the values themselves can be used. In Figure 4, $S(P_i + K)$ is an output of SubByte of exclusive-or between input and key such as advanced encryption standard (AES) SubBytes for classification.

In this section, we describe the application methods and characteristics of DLSCA based on deep learning architectures, namely, the MLP, CNN, and recent approaches. The DLSCA introduced in this section used the classification property of deep learning, unless explicitly stated otherwise.

4.1 | MLP-based profiling side-channel analysis

The MLP was the first neural network architecture applied to SCA owing to its structural simplicity. Studies have shown that the MLP can potentially enhance the performance of SCA. The first MLP-based SCA was performed by using regression to characterize leakage. More recently, MLP has been mostly used to classify the intermediate value of the trace.

Yang and others were the first to use the MLP as regression to characterize the power model of the Sbox output of the AES [57]. The choice of power models has a significant impact on the performance of attacks such as CPA. The Hamming weight and Hamming distance models are frequently used as a power model. For a leakage model, leakage characterization and average traces are occasionally used, because they are more similar to the actual leakage model than the Hamming weight or Hamming distance [3,58,59]. In [57], an MLP was trained using the Sbox output value and real power trace as the data and label, respectively. Then, the trained MLP was used as a power model function to improve the performance of CPA.

Martinasek and others were the first to use an MLP to categorize the Sbox output value from a trace [60]. They trained an MLP against an unprotected AES with its power traces and Sbox outputs. Since then, earlier works of MLP-based SCA have not used raw traces but have used pre-processed traces through average trace reduction, a wavelet transform, and PCA to improve the performance of the attack [61–63]. However, later works have confirmed that a DLSCA can achieve good performance even when raw traces are used as input data. For example, Maghrebi and others presented the results of experiments suggesting that an MLP without PCA outperforms an MLP with PCA [64]. Consequently, raw traces were used as inputs in subsequent studies.

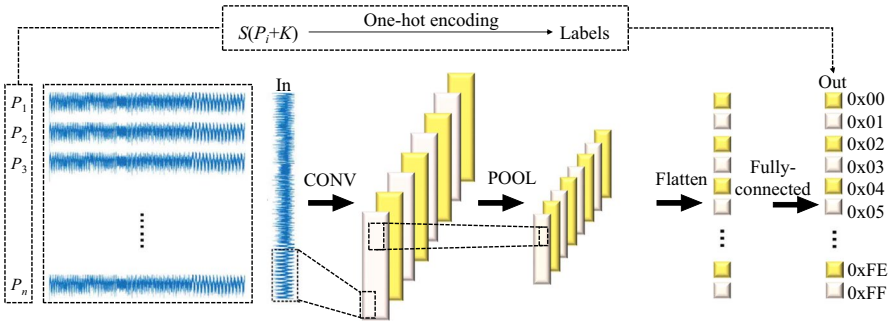

FIGURE 4 Example of a structure from deep learning-based side-channel attack. Traces and intermediate values are used as data and labels, respectively

There are ongoing studies that compare the performance of classical profiling attacks with that of deep learning-based profiling attacks. In [64,65], the performances of a classical template attack and an MLP-based attack were compared. In 2016, Maghrebi and others conducted experiments on both AES hardware and software implementations, analyzing their performance when implementing a template attack, random forest, MLP, CNN, autoencoder, and long short-term memory (LSTM) [64]. They demonstrated that unprotected AES hardware and software implementations can be analyzed with an MLP.

It has also been shown that first-order masked AES can be analyzed using an MLP [62,66,67]. For instance, Martinasek and other sanalyzed a masked AES software implementation using an MLP. They proposed two methods for analyzing an AES-rotating Sbox masking (RSM) implementation using an MLP on a publicly available dataset—the DPA contest v4 and v4.2 [68]. The first method used the MLP to find the masking information used in the RSM. Then, CPA was performed on the masked Sbox output value using the masking information obtained through the MLP. The second method used an MLP to identify the output value of the RSM from the trace, given the masking information obtained earlier through template attacks. Maghrebi and others were the first to confirm that first-order masked AES software can be analyzed by using only an MLP [64]. In other words, the MLP can identify the original Sbox output without any knowledge of the masking information.

4.2 | CNN-based profiling side-channel analysis

Unlike an MLP, which uses only numerical values without considering the data topology, a CNN has a structural property that is more robust to data distortions. For example, it is widely known that a CNN offers good performance for image recognition, despite considerable data distortions. In the case of SCA, traces are distorted owing to the noise resulting from the measurement environment and side-channel countermeasures. Therefore, it is natural to apply a CNN to SCA.

As mentioned earlier, Maghrebi and others applied the CNN architecture to SCA [64]. Cagli and others then proposed a CNN-based SCA on a protected AES with jitter-based hiding methods [69]. They were the first to show that a CNN could be used to neutralize jitter-based hiding countermeasures without any other pre-processing. In [69], through experimentation with learning the Sbox output of AES protected by random delay insertion and clock jitter, the robustness of a CNN to data distortions was demonstrated. Although CNN has fewer weights to train than MLP, it requires ample learning data to learn the general invariant features of the traces from a device protected by a hiding method. To deal with insufficient training data and to prevent overfitting, they also proposed a data augmentation technique to defeat hiding countermeasures. Specifically, jitter-based hiding methods were simulated by randomly shifting real traces and by inserting/removing a certain number of random points on real traces. These simulated traces were used as additional training data. Consequently, they confirmed that the training data were sufficiently increased for learning. Through their analysis, it was established that CNN-based SCA does not require pre-processing steps such as trace alignment. Their result indicates that CNN-based SCA can also be used to evaluate side-channel resistance objectively.

A study has shown that CNN with additional input neurons enhances the performance of DLSCA. Note that CNN is composed of two basic parts: feature extraction and feature classification/regression based on the extracted feature. Hettwer and others proposed a CNN architecture with domain knowledge (DK) neurons [70]. DK neurons are used as an additional input to the fully connected layer—as additional inputs for feature classification/regression. Their experiments were based on the assumption that this additional information can be used to improve the learning performance. The results in [70] showed that the adoption of the DK neurons improved the performance. In addition, they observed that learning the round key outperforms the case of learning the output of the Sbox. However, further investigation is required as they did not provide comprehensive reasoning regarding why learning the round key as a label shows a better performance than learning the Sbox output.

Instead of using raw power trace as input data, a technique involving transforming the power trace and using it as input data have also been proposed. On the one hand, Yang and others proposed a technique that uses the short-time Fourier transform to transform the power trace in one-dimensional data into a spectrogram expressed in time-frequency representation and used the spectrogram as input data [71]. They claimed that the spectrogram is more suitable for CNN because it contains the features of time and frequency information simultaneously. Experiments show that DLSCA using spectrograms can be similar to or better than DLSCA using power traces in the time domain. On the other hand, Kim and others proposed a method that adds artificial noise to the input trace for the robustness of DLSCA, similar to denoising autoencoder [72].

Carbone and others proposed CNN-based profiled SCA against a secure RSA implementation with message, exponent, and modulus blinding [73] as side-channel countermeasures [74]. Their work was the first DLSCA against public key cryptosystems. They used a CNN architecture to classify the address or value of the register. Their experiments showed that public key cryptosystems can also be analyzed using DLSCA.

After the CNN-based DLSCA technique was proposed, studies were conducted to compare the performance of

CNN-based SCA and other profiling SCA [75,76]. The ASCAD public dataset was proposed to compare the performances of DLSCA and other profiling SCA methods objectively [77]. In addition, reproducibility has begun to be emphasized, for example, by disclosing specific hyperparameters to make the results of previous studies available.

4.3 | Latest research direction on side-channel analysis using deep learning

To the best of our knowledge, three new directions for DLSCA have been proposed recently. These new approaches exploited the learning process of deep learning for SCA, rather than focusing solely on the input and output functions of the classification or regression processes.

The first direction is side-channel leakage detection using deep learning, independently proposed by Masure and others, Hettwer and others, and Timon. Whereas Masure and others and Hettwer and others proposed deep learning-based leakage detection methods for a profiled scenario, Timon proposed a technique for a non-profiled situation. In [78], by accumulating the absolute values of the weights of the first layer of the MLP, an attacker can determine on which points the leakage occurred. A disadvantage is that it is difficult to apply this method in a CNN architecture. Sensitivity analysis can be used for leakage detection using a CNN [78,79]. Generally, sensitivity analysis is used to understand the operation of a mathematical model. In DLSCA, sensitivity analysis can be used for leakage detection and as a new indicator of whether a neural network has learned.

In [79], Masure and others proposed selecting POI using sensitivity analysis. Whereas classical POI selection uses the SNR, the POI selection method proposed by these authors is based on a leakage detection method called gradient visualization. They confirmed that there are differences between the SNR and gradient visualization in terms of POI selection. To compare the classical POI selection method with their proposed gradient visualization technique, template attacks were performed with different POI selection methods. The results from [79] illustrated that template attacks perform better when using the gradient visualization technique to select POI.

A leakage analysis technique using attribution methods was proposed in a profiled scenario [80]. Attribution methods were used to analyze how each component of input datum influences the output. Hettwer and others used such attribution methods for leakage analysis, and presented three attribution methods, which are based on the saliency map [81], layer-wise relevance propagation [82], and occlusion sensitivity analysis [83]. These methods detect leakage and are used to select POI in a similar manner as the aforementioned gradient visualization technique. Additionally, Hettwer and others proposed the usage of the attribution method as an SCA distinguisher. According to the value of the output node, different

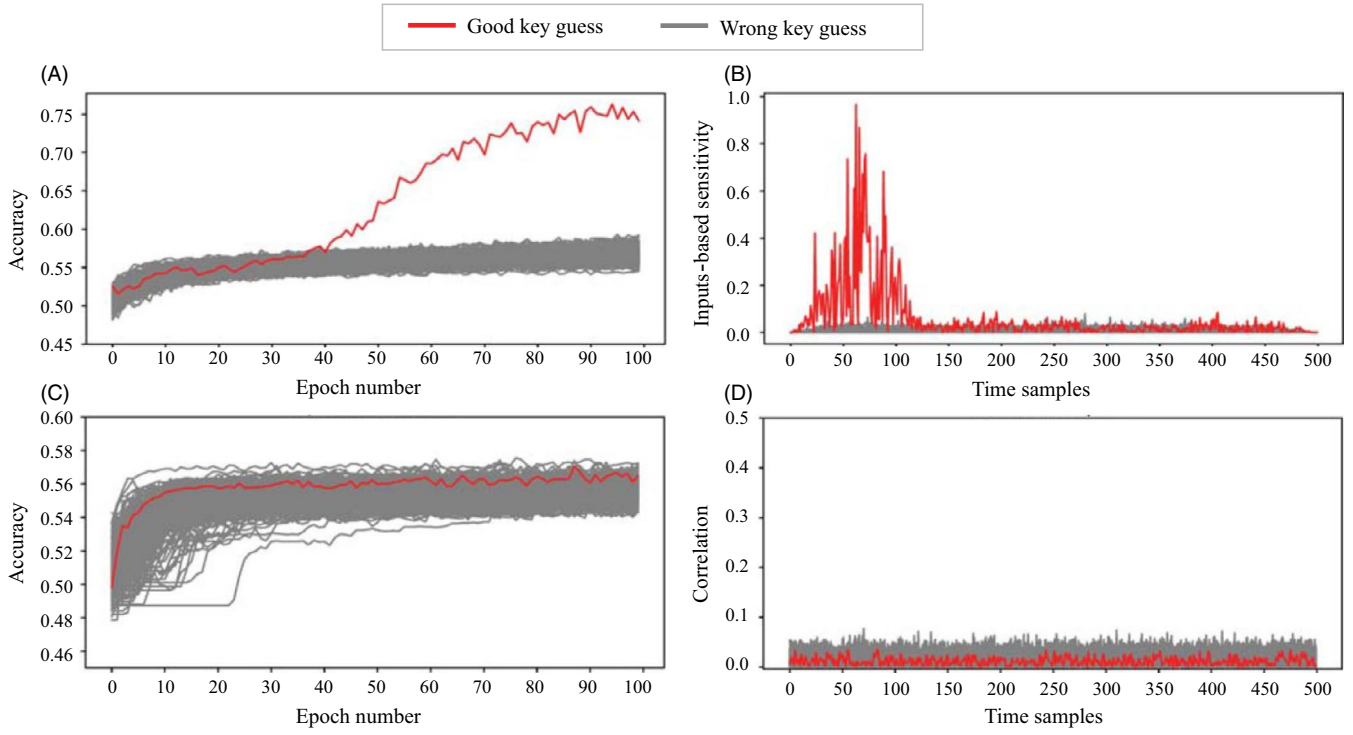

FIGURE 5 Results of attack on unprotected implementation with de-synchronization [78]. (A) CNN-DDLA accuracy. (B) CNN-DDLA input-based sensitivity. (C) MLP-DDLA accuracy. (D) CPA

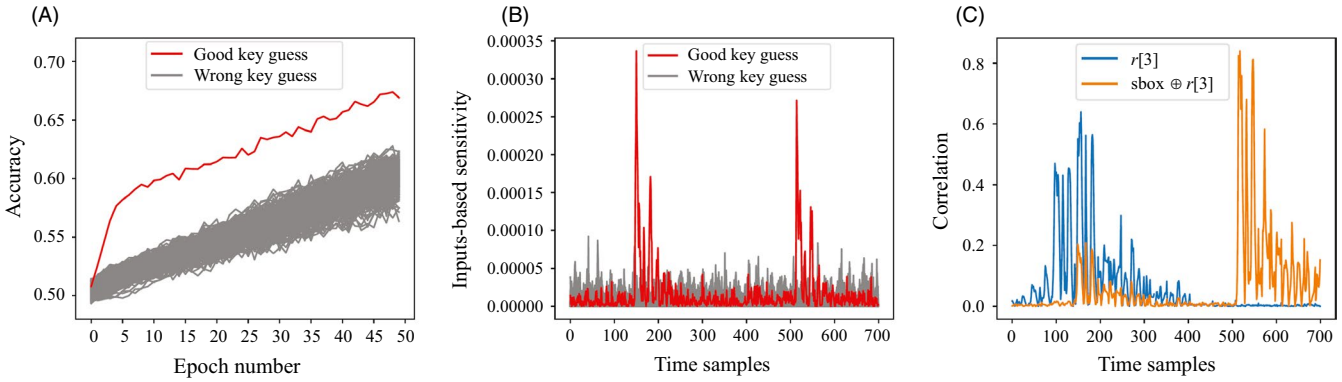

FIGURE 6 Results of MLP-DDLA attack on ASCAD [78]. (A) Accuracy, (B) Inputs-based sensitivity, and (C) CPA reverse engineering

properties of attribution are observed. Based on these properties, it is possible to find the key using the attribution method.

The second direction is a deep learning-based non-profiling SCA technique. Timon was the first to propose a non-profiling method called differential deep learning analysis (DDLA), which exploits the fact that whether a neural network is trained depends on whether the key is correctly guessed [78]. DDLA uses the trace and intermediate value as the input and label, respectively. As DDLA is a partition-based SCA, intermediate values that are the output of injective functions such as MSB and LSB, or the Hamming weight of intermediate values must be used as a label. The output of the injective function reflects the partition class well according to the key guess. Similar to DPA and CPA, in DDLA, the intermediate value is computed through the guessed round key and is used as a label. Note that an intermediate value calculated with the wrong key is unrelated to the trace such that the neural network will not be trained on the traces. However, when the intermediate value computed through the right key is related to the trace, the neural network can be trained. Based on this examination, after training the neural network for each key guess, an attacker can determine the correct key by observing the variation of a metric such as the accuracy or output of a loss function.

Timon experimentally verified that CNN-based DDLA can be used against both masking and hiding countermeasures. Figure 5 shows the performance of DDLA and CPA when an MLP and CNN are used for DDLA to analyze AES with hiding countermeasures. As has been repeatedly emphasized previously, CNN-based SCA does not require pre-processing in the non-profiled scenario. Moreover, as depicted in Figures 5 and 6, DDLA can detect leakage with sensitivity analysis in a non-profiled scenario. Although DDLA has the advantage that it can be implemented in a non-profiled scenario, suitable hyperparameters must be determined and set. Furthermore, an attacker must train the neural network in proportion to the number of keys.

The third direction is the usage of a neural network as a trace encoder for non-profiling SCA. Robyns and others proposed a correlation optimization technique for improving correlation electromagnetic analysis (CEMA) [84]. For key recovery through CEMA, only one time sample for each trace is important. Accordingly, they trained a neural network that uses trace as the input, a sample as the output for each byte for regression, and the constraint to maximize the correlation between the output and intermediate value as a loss function. Through such training, the neural network becomes the encoding function that outputs a single sample to maximize the performance of CEMA. They conducted experiments on CEMA using correlation optimization. Their experiments indicated that CEMA can outperform CNN-based classification, using correlation optimization even with shallow MLP and transformed traces, which are desynchronized traces transformed into the frequency domain.

5 | CONCLUSION AND FUTURE WORKS

In this study, we surveyed recent, state-of-the-art advances in DLSCA. We confirmed that an MLP and CNN are effective for SCA even without pre-processing. We also surveyed new approaches for applying deep learning algorithms to non-profiling attacks, leakage detection, and leakage encoder. To provide a comprehensive overview, we summarized our findings in Table 1.

Owing to the characteristics of deep learning algorithms, DLSCA does not require pre-processing. This makes it suitable as a tool for evaluating side-channel resistance objectively. However, as most of the studies focus on profiling attacks, additional studies dealing with the following aspects are required.

First, a comprehensive study of the application of deep learning in SCA is required to interpret why the neural network is effective for SCA. Previous studies only confirmed that deep learning algorithms enhance SCA performance

TABLE 1 Summary of deep learning-based side-channel attack studies

Ref.	Architecture	Type	Target	Countermeasure	Usage	Remark
[57]	MLP	Profiling	AES	None	Regression	Leakage characterization
[60]	MLP	Profiling	AES	None	Classification	Average traces for each plaintext are used
[61]	MLP	Profiling	AES	None	Classification	Above average traces for each plaintext was reduced by average trace for all traces.
[63]	MLP	Profiling	AES	None	Classification	Discrete wavelet transform is used as pre-processing
[62]	MLP	Profiling	AES	Masking	Classification	Mask and value of output of masked Sbox are identified separately, and PCA is used as pre-processing
[66]	MLP	Profiling	AES	Masking	Classification	Masking information is identified prior to DPA
[67]	MLP	Profiling	AES	Masking	Classification	Value of output of masked Sbox is identified after masking information is identified using template attack
[64]	MLP/CNN/ Autoencoder/ LSTM	Profiling	AES	Masking	Classification	Unknowing masking information, value of output of Sbox is identified
[69]	CNN	Profiling	AES	Masking/Hiding	Classification	To defeat jitter-based hiding countermeasure, CNN with data augmentation are used
[70]	CNN	Profiling	AES	Masking/Hiding	Classification	Domain knowledge neurons are introduced
[71]	CNN	Profiling	AES	Masking	Classification	Spectrogram of trace is used as input
[72]	CNN	Profiling	AES	Masking/Hiding	Classification	Adding artificial noise to the input
[74]	CNN	Profiling	RSA	Message blinding/ Exponent blinding/ Modulus blinding	Classification	Value manipulation and Register manipulation are analyzed
[75]	MLP/CNN	Profiling	AES	Masking/Hiding	Classification	Analyzing on performance of CNN-based SCA
[76]	CNN	Profiling	AES	Masking/Hiding	Classification	Comparison between CNN-based SCA and Template attack
[77]	CNN	Profiling	AES	Masking/Hiding	Classification	Public dataset ASCAD is introduced
[78]	MLP/CNN	Non-profiling	AES	Masking/Hiding	Classification	Non-profiling attack and leakage detection
[79]	CNN	Profiling	AES	Masking/Hiding	Classification	POIs selection and leakage detection based on gradient visualization
[80]	CNN	Profiling	AES	Masking/Hiding	Classification	POIs selection and leakage detection based on attribution method
[84]	MLP	Profiling	AES	Masking/Hiding	Regression	Trace encoder for non-profiling SCA

through various experiments. Understanding the exact reason is crucial to remove leakage issues that currently lead to performance issues.

Second, an efficient hyperparameter search method is required for SCA. The performance of SCA cannot benefit from using a deep learning algorithm if the hyperparameter is not selected properly, and searching for a suitable hyperparameter is expensive. Hence, studies should examine an efficient hyperparameter search method for SCA.

Third, other deep learning architectures must be explored. Many other deep learning architectures exist, and each architecture offers different properties. DLSCA can benefit significantly from an intensive survey of existing architectures.

As a final note, we thank the authors of prior studies on DLSCA.

ORCID

Sunghyun Jin https://orcid.org/0000-0002-9521-0937
Seokhie Hong https://orcid.org/0000-0001-7506-4023

REFERENCES

1. P.C. Kocher, *Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems*, in Proc. Annu. Int. Cryptology Conf., Santa Barbara, CA, USA, 1996, pp.104–113.
2. P. Kocher, J. Jaffe, and B. Jun, *Differential power analysis*, in Ann. Int. Cryptol. Conf., Santa Barbara, CA, USA, Aug. 1999, pp. 388–397.
3. S. Mangard, E. Oswald, and T. Popp, Power analysis attacks: Revealing the secrets of smart cards, Springer Science & Business Media, Heidelberg, 2008.
4. L. Lerman, Z. Martinasek, and O. Markowitch, *Robust profiled attacks: should the adversary trust the dataset?* IET Inf. Secur. **11** (2016), no. 4, 188–194.

5. Z. Martinasek et al., *k-nearest neighbors algorithm in profiling power analysis attack*, Radioeng. **25** (2016), no. 2, 365–382.
6. S. Picek et al., *The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations*, IACR Trans. Cryptogr. Hardw. Embed. Syst. **1** (2018), no. 8, 209–237.
7. B. Hettwer, S. Gehrer, and T. Güneysu, *Applications of machine learning techniques in side-channel attacks: a survey*, J. Cryptogr. Eng. (2019), https://doi.org/10.1007/s13389-019-00212-8.
8. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016. https://www.deeplearningbook.org/.
9. Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature **521** (2015), no. 7553, 436–444.
10. S. Mahdavifar and A. A. Ghorbani, *Application of deep learning to cybersecurity: a survey*, Neurocomput. **347** (2019), 149–176.
11. K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Netw. **4** (1991), no. 2, 251–257.
12. G. Hinton, N. Srivastava, and K. Swersky, *Neural networks for machine learning lecture 6a overview of mini-batch gradient descent*, Neural Netw. Machine Learn., Coursera MOOC, 2012. https://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf.
13. D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization*, arXiv preprint arXiv:1412.6980, 2014. https://arxiv.org/abs/1412.6980.
14. J. Bergstra and Y. Bengio, *Random search for hyper-parameter optimization*, J. Machine Learn. Res. **13** (2012), 281–305.
15. S. R. Young et al. *Optimizing deep learning hyper-parameters through an evolutionary algorithm*, in Proc. Workshop Mach. Learn. High-Perform. Comput. Environ., New York, NY, USA, Nov. 2015, pp. 4:1–5.
16. N. Tikhonov, *On the stability of inverse problems*, Dokl. Akad. Nauk SSSR **39** (1943), 195–198.
17. R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Roy. Stat. Soc.: Ser. B (Methodol.) **58** (1996), no. 1, 267–288.
18. N. Srivastava et al., *Dropout: a simple way to prevent neural networks from overfitting*, J. Mach. Learn. Res. **15** (2014), no. 1, 1929–1958.
19. S. Ioffe and C, Szegedy. *Batch normalization: accelerating deep network training by reducing internal covariate shift*, arXiv preprint arXiv:1502.03167, 2015. https://arxiv.org/abs/1502.03167v2.
20. N. Jaitly and G. E. Hinton, *Vocal tract length perturbation (VTLP) improves speech recognition*, in Proc. Int. Conf. Machine. Learning, Atlanta, GA, USA, 2013, pp. 1–5.
21. D. H. Hubel and T. N. Wiesel, *Receptive fields and functional architecture of monkey striate cortex*, J. Phys. **195** (1968), no. 1, 215–243.
22. K. Fukushima, *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*, Biol. Cybern. **36** (1980), no. 4, 193–202.
23. Y. LeCun et al., *Handwritten digit recognition with a back-propagation network*, in Proc. Adv. Neural Inf. Process. Syst, Denver, CO, USA, Nov. 1989, pp. 396–404.
24. Y. LeCun et al., *Gradient-based learning applied to document recognition*, Proc. IEEE **86** (1998), no. 8, 2278–2324.
25. A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Proc. Adv. Neural Inf. Process. Syst., Stateline, NV, USA, 2012, pp. 1097–1105.
26. E. Brier, C. Clavier, and F. Olivier, *Correlation power analysis with a leakage model*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Cambridge, MA, USA, Aug. 2004, pp. 16–29.
27. B. Gierlichs et al., *Mutual information analysis*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Washington, D.C., USA, Aug. 2008, pp. 426–442.
28. J. G. van Woudenberg, M. F. Witteman, and B. Bakker, *Improving differential power analysis by elastic alignment*, Cryptogr. Track RSA Conf., San Francisco, CA, USA, Feb. 2011, pp. 104–119.
29. R. A. Muijrers, J. G. van Woudenberg, and L. Batina, *Ram: rapid alignment method*, in Proc. Int. Conf. Smart Card Res. Adv. Applicat., Leuven, Belgium, Sept. 2011, pp. 266–282.
30. S. Chari, J. R. Rao, and P. Rohatgi, *Template attacks*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Redwood Shores, CA, USA, Aug, 2002, pp. 13–28.
31. W. Schindler, K. Lemke, and C. Paar, *A stochastic model for differential side channel cryptanalysis*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Edinburgh, UK, 2005, pp. 30–46.
32. C. Archambeau et al., *Template attacks in principal subspaces*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Yokohama, Japan, Oct. 2006, pp. 1–14.
33. F.-X. Standaert and C. Archambeau, *Using subspace-based template attacks to compare and combine power and electromagnetic information leakages*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Washington, D.C., USA, Aug. 2008, pp. 411–425.
34. O. Choudary and M. G. Kuhn, *Efficient template attacks*, in Proc. Int. Conf. Smart Card Res. Adv. Applicat., Berlin, Germany, Nov. 2013, pp. 253–270.
35. S. Chari et al. *Towards sound approaches to counteract power-analysis attacks*, in Proc. Annu. Int. Cryptol. Conf. (CRYPTO), Santa Barbara, CA, USA, Aug. 1999, pp. 398–412.
36. L. Goubin and J. Patarin, *Des and differential power analysis the "duplication" method*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Worcester, MA, USA, Aug. 1999, pp. 158–172.
37. J.-S. Coron and I. Kizhvatov, *An efficient method for random delay generation in embedded software*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst., Lausanne, Switzerland, Sept. 2009, pp. 156–170.
38. J-S. Coron and I. Kizhvatov, *Analysis and improvement of the random delay countermeasure of CHES 2009*, in Int. Workshop Cryptogr. Hardw. Embed. Syst., Santa Barbara, USA, Aug. 2010, pp. 95–109.
39. N. Veyrat-Charvillon et al., *Shuffling against side-channel attacks: A comprehensive study with cautionary note*, in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Sec. (ASIACRYPT), Beijing, China, Dec. 2012, pp. 740–757.
40. K. Tiri, M. Akmal, and I. Verbauwhede, *A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards*, in Proc. Eur. Solid-State Circ. Conf., Florence, Italy, Sept. 2002, pp. 403–406.
41. T. Popp and S. Mangard., *Masked dual-rail pre-charge logic: DPA-resistance without routing constraints*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Edinburgh, UK, Aug. 2005, pp. 172–186.
42. C. Chen et al. *Balanced encoding to mitigate power analysis: a case study*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Paris, France, Nov. 2014, pp. 49–63.
43. H. Maghrebi, V. Servant, and J. Bringer, *There is wisdom in harnessing the strengths of your enemy: customized encoding to thwart side-channel attacks*, in Proc. Int. Conf. Fast Softw. Encrypt. (FSE), Bochum, Germany, Mar. 2016, pp. 223–243.
44. G. Becker et al. *Test vector leakage assessment (TVLA) methodology in practice*, in Proc. Int. Cryptogr. Module Conf. 1001 (2013).

45. T. S. Messerges. *Using second-order power analysis to attack DPA resistant software*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Worcester, MA, USA, Aug. pp. 238–251.
46. E. Prouff, M. Rivain, and R. Bevan, *Statistical analysis of second order differential power analysis*, IEEE Trans. Comput. **58** (2009), no. 6, 799–811.
47. C. Clavier, J. S. Coron, and N. Dabbous, *Differential power analysis in the presence of hardware countermeasures*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Worcester, MA, USA, Aug. 2000, pp. 252–263.
48. S. Nagashima et al., *DPA using phase-based waveform matching against random-delay countermeasure*, in Proc. IEEE Int. Symp. Cir. Syst., New Orleans, LA, USA, May 2007, pp. 1807–1810.
49. F. Durvaux et al., *Efficient removal of random delays from embedded software implementations using hidden markov models*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Graz, Austria, Nov. 2012, pp. 123–140.
50. L. Lerman, G. Bontempi, and O. Markowitch, *Side channel attack: an approach based on machine learning*, in Proc. Int. Workshop Construct. Side-Channel Anal. Secure Design (COSADE), Darmstadt, Germany, 2011, pp. 29–41.
51. G. Hospodar et al., *Machine learning in side-channel analysis: a first study*, J. Cryptogr. Eng. **1** (2011), no. 4, 293–302.
52. T. Bartkewitz and K. Lemke-Rust, *Efficient template attacks based on probabilistic multi-class support vector machines*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Graz, Austria, Nov. 2012, pp. 263–276.
53. A. Heuser and M. Zohner, *Intelligent machine homicide*, in Proc. Int. Workshop Construct. Side-Channel Anal. Secure Design (COSADE), Darmstadt, Germany, May 2012, pp. 249–264.
54. J. Heyszl et al., *Clustering algorithms for non-profiled single-execution attacs on exponentiations*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Berlin, Germany, Nov. 2013, pp. 79–93.
55. L. Lerman, G. Bontempi, and O. Markowitch, *A machine learning approach against a masked AES*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Berlin, Germany, Nov. 2013, pp. 61–75.
56. R. Specht et al., *Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution em measurements*, in Proc. Int. Workshop Construct. Side-Channel Anal. Secure Design (COSADE), Berlin, Germany, Apr. 2015, pp. 3–19.
57. S. Yang et al., *Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations*, in Proc. Int. Conf. Inf. Secur. Cryptol. (ICISC), Seoul, Rep. of Korea, Nov. 2011, pp. 169–185.
58. C. Whitnall and E. Oswald, *Profiling DPA: efficacy and efficiency trade-offs*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Santa Barbara, CA, USA, Aug. 2013, pp. 37–54.
59. A. Moradi and F. X. Standaert, *Moments-correlating DPA*, in Proc. ACM Workshop Theory Implement. Secur. (ACM), Vienna, Austria, Oct. 2016, pp. 5–15.
60. Z. Martinasek and V. Zeman, *Innovative method of the power analysis*, Radioengineering **2** (2013), no. 2, 586–594.
61. Z. Martinasek, J. Hajny, and L. Malina, *Optimization of power analysis using neural network*, in Proc. Int. Conf. Smart Card Res. Adv. Applicat. (CARDIS), Berlin, Germany, Nov. 2014, pp. 94–107.
62. R. Gilmore, N. Hanley, and M. O'Neill, *Neural network based attack on a masked implementation of AES*, in Proc. IEEE Int. Symp. Hardw. Orient. Secur. Trust (HOST), Washington, DC, USA, May 2015, pp. 106–111.
63. P. Saravanan et al., *Power analysis attack using neural networks with wavelet transform as pre-processor*, in Proc. Int. Symp. VLSI Design Test, Coimbatore, India, July 2014, pp. 1–6.
64. H. Maghrebi, T. Portigliatti, and E. Prouff, *Breaking cryptographic implementations using deep learning techniques*, in Proc. Int. Conf. Secur. Privacy Appl. Cryptogr. Eng. (SPACE), Hyderabad, India, Dec. 2016, pp. 3–26.
65. Z. Martinasek and L. Malina, *Comparison of profiling power analysis attacks using templates and multi-layer perceptron network*, 1st Int. Conf. Math. Method Sci. Eng. (MMCTSE), 2014, pp. 134–139.
66. Z. Martinasek et al., *Power analysis attack based on the MLP in DPA contest v4*, in Proc. Int. Conf. Telecommun. Signal Process., Prague, Czech Republic, July 2015, pp. 154–158.
67. Z. Martinasek, P. Dzurenda, and L. Malina, *Profiling power analysis attack based on mlp in DPA contest v4.2*, in Proc. Int. Conf. Telecommun. Signal Process. (TSP), Vienna, Austria, June 2016, pp. 223–226.
68. TELECOM ParisTech~SEN Research Group, DPA contest (4th ed.), 2013–2014, http://www.DPAcontest.org/v4/.
69. E. Cagli, C. Dumas, and E. Prouff, *Convolutional neural networks with data augmentation against jitter-based countermeasures*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Taipei, Taiwan, Sept. 2017, pp. 45–68.
70. B. Hettwer, S. Gehrer, and T. Güneysu, *Profiled power analysis attacks using convolutional neural networks with domain knowledge*, in Proc. Int. Conf. Select. Areas Cryptogr. (SAC). Calgary, Canada, 2018, pp. 479–498.
71. G. Yang et al., *Convolutional neural network based side-channel attacks in time-frequency representations*, in Proc. Int. Conf. Smart Card Res. Adv. Appl. (CARDIS), Montpellier, France, Nov. 2018, pp. 1–17.
72. J. Kim et al., *Make some noise: Unleashing the power of convolutional neural networks for profiled side-channel analysis*, Cryptology ePrint-Archive Report 2018/1023, 2018, https://eprint.iacr.org/2018/1023.
73. J.-S. Coron, *Resistance against differential power analysis for elliptic curve cryptosystems*, in Proc. Int. Workshop Cryptogr. Hardw. Embed. Syst. (CHES), Worcester, MA, USA, Aug. 1999, pp. 292–302.
74. M. Carbone et al., *Deep learning to evaluate secure rsa implementations*, IACR Trans. Cryptogr. Hardw. Embed. Syst. **2** (2019), no. 5, 132–161.
75. S. Picek et al., *On the performance of convolutional neural networks for side-channel analysis*, in Proc. Int. Conf. Secur. Privacy Appl. Cryptogr. Eng., Kanpur, India, Dec. 2018, pp. 157–176.
76. Y. Zotkin, F. Olivier, and E. Bourbao (eds.), *Deep learning vs. template attacks in front of fundamental targets: experimental study*, Cryptology ePrint Archive Report 2018/1213, 2018, https://eprint.iacr.org/2018/1213.
77. E. Prouff et al., *Study of deep learning techniques for side-channel analysis and introduction to ascad database*, Cryptology ePrint, Archive, Report 2018 (2018/53), https://eprint.iacr.org/2018/053.
78. B. Timon, *Non-profiled deep learning-based side-channel attacks with sensitivity analysis*, IACR Trans. Cryptogr. Hardw. Embed. Syst. **2** (2019), no. 4, 107–131.
79. L. Masure, C. Dumas, and E. Prouff, *Gradient visualization for general characterization in profiling attacks*, Cryptology ePrint Archive, Report 2018/1196, 2018, https://eprint.iacr.org/2018/1196.
80. B. Hettwer, S. Gehrer, and T. Güneysu, *Deep neural network attribution methods for leakage analysis and symmetric key recovery*,

Cryptology ePrint Archive, Report 2019/143, 2019, https://eprint.iacr.org/2019/143

81. A. Vedaldi, A. Zisserman, and K. Simonyan, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, arXiv preprint arXiv:1312.6034, 2013. https://arxiv.org/abs/1312.6034.
82. S. Bach et al., *On pixel-wise explanations for non-linear classier decisions by layer-wise relevance propagation*, PLoS ONE **10** (2015), no. 7, e0130140. https://doi.org/10.1371/journal.pone.0130140.
83. M. D. Zeiler and R. Fergus, *Deep inside convolutional networks: Visualising image classification models and saliency maps*, arXiv preprint arXiv:1311.2901, 2013. https://arxiv.org/abs/1311.2901.
84. P. Robyns, P. Quax, and W. Lamotte, *Improving cema using correlation optimization*, IACR Trans. Cryptogr. Hardw. Embed. Syst. (TCHES) **1** (2019), no. 1, 1–24.

AUTHOR BIOGRAPHIES

Sunghyun Jin received his BS degree in mathematics and computer science from the University of Seoul, Seoul, Rep. of Korea, in 2015. He received his MS degree in information security from Korea University, Seoul, Rep. of Korea, in 2017, where he is currently pursuing his PhD degree. His main research interests include side-channel analysis and its countermeasures, and machine learning-based cryptanalysis.

Suhri Kim received her BS degree in mathematics from Korea University, Seoul, Rep. of Korea, in 2014. She received her MS degree in information security from Korea University, Seoul, Rep. of Korea, in 2016, where she is currently pursuing her PhD degree. Her main research interests include public key cryptosystems, optimizing implementations, and side-channel analysis and its countermeasures.

HeeSeok Kim received his BS degree in mathematics from Yonsei University, Seoul, Rep. of Korea, in 2006, and his MS and PhD degrees in engineering and information security from Korea University, Seoul, Rep. of Korea, in 2008 and 2011, respectively. He worked as a postdoctoral researcher at the University of Bristol, UK, from 2011 to 2012. From 2013 to 2016, he worked as a senior researcher at the Korea Institute of Science and Technology Information. Since 2016, he has been with Korea University. His research interests include side-channel attacks, cryptography, and network security.

Seokhie Hong received his MS and PhD degrees in mathematics from Korea University, Seoul, Rep. of Korea, in 1997 and 2001, respectively. He worked for SECURITY Technologies Inc. from 2000 to 2004. From 2004 to 2005, he worked as a postdoctoral researcher with COSIC at KU Leuven, Belgium. Since 2005, he has been with Korea University, where he is now working in the School of Cyber Security. His specialty lies in the area of information security, and his research interests include the design and analysis of symmetric-key cryptosystems, public-key cryptosystems, and forensic systems.