Imbalanced Data Problems in Deep Learning-Based Side-Channel Attacks: Analysis and Solution

Akira Ito, Kotaro Saito, Rei Ueno, *Member, IEEE*, and Naofumi Homma, *Senior Member, IEEE*

***Abstract*—In recent years, the threat of profiling attacks using deep learning has emerged. Successful attacks have been demonstrated against various types of cryptographic modules. However, the application of deep learning to side-channel attacks (SCAs) is often not adequately assessed because the labels that are widely used in SCAs, such as the Hamming weight (HW) and Hamming distance (HD), follow an imbalanced distribution. This study analyzes and solves the problems caused by dataset imbalance during training and inference. First, we state the reasons for the negative effect of data imbalance in classification for deep-learning-based SCAs and introduce the Kullback–Leibler (KL) divergence as a metric to measure this effect. Using the KL divergence, we demonstrate through analysis how the recently reported cross-entropy ratio loss function can solve the problem of imbalanced data. We further propose a method to solve dataset imbalance at the inference phase, which utilizes a likelihood function based on the key value instead of the HW/HD. The proposed method can be easily applied in deep-learning-based SCAs because it only needs an extra multiplication of the inverted binomial coefficients and inference results (i.e., the output probabilities) from the conventionally trained model. The proposed solution corresponds to data-augmentation techniques at the training phase, and furthermore, it better estimates the keys because the probability distributions of the training and test data are preserved. We demonstrate the validity of our analysis and the effectiveness of our solution through extensive experiments on two public databases.**

***Index Terms*—Side-channel attacks, deep learning, imbalanced data.**

I. Introduction

A NUMBER of side-channel attacks (SCAs) and corresponding countermeasures have been presented since Kocher *et al.* discovered this type of attack [1]. Among the existing SCAs, the profiling attack is considered to be a powerful variation. A profiling attack consists of a profiling phase and an attack phase. In the profiling phase, an attacker extracts the characteristics of a target device. In the attack phase, the attacker retrieves secret information using the side-channel information from the actual target device in combination with the characteristics extracted in the profiling phase. Various studies have shown that profiling attacks can estimate secret information more readily than simple (i.e., non-profiling) SCAs, even on cryptographic hardware with countermeasures such as masking and random delays [2]. In the context of the Internet of Things, where an attacker can easily access the same type of device as the target, such profiling attacks are becoming more practical and feasible.

The template attack, which was the first profiling attack to be proposed, creates a leakage model based on the assumption that the side-channel information follows a multi-dimensional Gaussian distribution [3]. In the attack phase, the secret information is estimated using the likelihood calculated by the estimated model. Principal component analysis and linear discriminant analysis are sometimes used as dimensionality reduction methods to improve the efficiency of this phase [4]–[7]. One major issue with template attacks is that their assumptions are sometimes unrealistic; that is, the side-channel information of many cryptographic devices is not necessarily represented as a multi-dimensional Gaussian distribution, which makes it difficult to evaluate the potential threats of a profiling attack to such cryptographic devices.

Recently, a new profiling attack based on deep learning (DL) was presented as a more efficient alternative [8]–[11]. In the profiling phase, a deep neural network (DNN) is trained such that the input is the side channel information from the profiled device and the output is the probability of an intermediate value in the cryptographic computation (e.g., the output of S-box in the AES first round). Next, in the attack phase, the log likelihood calculated with the trained DNN is used to estimate the secret information. To reduce the complexity during training, the output of the DNN (i.e., label) is sometimes given by the Hamming weight (HW) or Hamming distance (HD) of the intermediate value [12]. If the intermediate value is directly estimated, the dimension of the model output exponentially increases as the bit-length of the intermediate value increases. Hence, the HW/HD model is more feasible and scalable for various cryptographic implementations and leakage models. Unlike the earlier described template attacks, such DL-based profiling attacks have the advantage that no special (and sometimes unrealistic) assumptions about side-channel information are required, and their effectiveness has been demonstrated experimentally [13].

However, there is a notable problem with DL-based profiling attacks called the imbalanced data problem. This problem causes difficulty during learning due to the imbalance in the

Manuscript received February 11, 2021; revised May 16, 2021; accepted June 13, 2021. Date of publication June 24, 2021; date of current version August 3, 2021. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 20K19765, in part by the JSPS Research Fellow under Grant 20J12887, and in part by the Japan Science and Technology Agency (JST) Core Research for Evolutional Science and Technology (CREST), Japan, under Grant JPMJCR19K5. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Debdeep Mukhopadhyay. *(Corresponding author: Akira Ito.)*

Akira Ito and Kotaro Saito are with the Graduate School of Engineering, Tohoku University, Sendai 980-8577, Japan (e-mail: ito@riec.tohoku.ac.jp; skot@riec.tohoku.ac.jp).

Rei Ueno and Naofumi Homma are with the Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan (e-mail: rei.ueno.a8@tohoku.ac.jp; homma@riec.tohoku.ac.jp).

Digital Object Identifier 10.1109/TIFS.2021.3092050

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

occurrence probabilities of the HW/HD used for training. Moreover, when data are imbalanced, conventional machine learning metrics (e.g., accuracy and precision) become poor indicators of performance [12]. In [12], the ineffectiveness of these metrics was demonstrated experimentally and the Synthetic Minority Oversampling Technique (SMOTE) was proposed to solve the difficulty in learning. SMOTE is one of the most well-known augmentation techniques. However, the effectiveness of this data augmentation technique in the context of profiling SCAs has not yet been analyzed. In particular, though it is known that equalizing the occurrence probabilities of the class labels leads to changes in the probability distributions of the training and test data, the effect of the changes caused by SMOTE on the success rate of profiling SCAs is not mentioned in the existing literature.

Another solution to the problem, cross entropy ratio (CER) was reported in [14]. CER is the ratio of the averaged negative log likelihood (NLL) of all wrong keys to the NLL of the correct one. It was found in [14] that profiling attacks always succeed if the CER is less than 1 and the number of available traces is sufficient during the attack phase. In addition, [14] experimentally demonstrated that CER could solve the ineffectiveness of the performance metrics. However, it is still unclear why CER is effective for imbalanced data because the attack assumption used in [14], namely, independence between the intermediate values of the correct key and those of other key candidates, does not hold in general [15].

Thus, while the imbalanced data problem has been solved experimentally in [12] and [14], no analytical explanation of why the conventional solutions are effective has yet been provided, which makes the accurate assessment of DL-based SCA threats and development of effective countermeasures difficult.

A. Our Contributions

To address the problems caused by imbalanced data, this study first clarifies the negative effects of imbalanced data on DL-based SCAs, using a quantitative evaluation metric. In addition, we present a new solution to eliminate the negative effects of imbalanced data at the inference phase. The contributions of this study are twofold and can be summarized as follows:

1) **Analysis of the negative effects of imbalanced data using a quantitative evaluation metric**
   We first show that the degree of imbalance in the output distribution of neural networks (NNs) can be quantitatively assessed by the Kullback–Leibler (KL) divergence between the output distribution and binomial distribution. The motivation for the use of the KL divergence is to quantitatively evaluate the difficulty of DL-based SCAs directly from the viewpoint of the model's output distribution shape. In the proposed metrics, a small KL divergence means that the model's output distribution is close to the binomial distribution, which indicates a strong influence of the class imbalance. By contrast, a large KL divergence means that the model's output distribution has a steep shape like a one-hot vector. In this case, the model would be less influenced by the imbalanced data and would perform better for the key recovery in the attack phase. Next, we analyze the negative effects of imbalanced data using the KL divergence. In particular, we explain why CER can mitigate the negative effects of imbalanced data.
2) **A solution for the imbalanced data problem in the inference phase**
   We propose a new key estimation method based on the key-based likelihoods obtained during inference. The aim is to eliminate the negative effects of imbalanced data and yield a more efficient key estimation approach than the conventional HW/HD-based ones.[1] We then clarify the differences between the proposed method (i.e., the use of key-based likelihoods) and conventional data augmentation methods, such as SMOTE, which equalize the occurrence probabilities of the labels. Through experiments, we demonstrate that the proposed solution performs better than the conventional solutions such as data augmentation and the CER loss function because it does not shift the model's output away from the true distribution. In particular, we show that the problem caused by imbalanced data can be substantially reduced by the proposed method even when the KL divergence is small.

B. Paper Organization

The remainder of this paper is organized as follows. Section II describes profiling attacks using DNNs and the imbalanced data problem. In Section III, we state the reasons for the negative effects caused by imbalanced data on SCAs and use the KL divergence to analyze these effects in a quantitative manner. In Section IV, we explain the reason why the CER loss can mitigate the negative effects of imbalanced data from the viewpoint of KL divergence. Section V presents the proposed inference phase-based solution to the imbalanced data problem and describes the differences between the proposed solution and the conventional data augmentation. Section VI presents a set of experimental results to evaluate the claims of this paper, and finally, Section VII concludes the paper.

II. Preliminaries

This section presents a brief overview of profiling attacks using DNNs. We then describe the problems caused by imbalanced data in both the training and inference phases.

A. DL-Based Profiling Attacks

This study mainly focuses on DL-based SCAs against AESs. A typical DL-based attack exploits the output of the S-box in the first round of AES and estimates a secret key in a similar manner to common non-profiling SCAs.

First, we describe the profiling (or training) phase. An attacker acquires training data from a device for profiling

[1] Note that this does not mean that the key values are the outputs of the NNs; HW/HD values are output by the NN, and the probability of each key is estimated using the outputs.

in advance. Training data $\mathcal{S}_P$ can be defined as

$$\mathcal{S}_P = \{(k_i, m_i, \boldsymbol{x}_i) \mid 1 \leq i \leq N_P\}, \tag{1}$$

where k_i denotes the one-byte partial key of a full key at ith observation, m_i denotes the one-byte plain text corresponding to k_i, $\boldsymbol{x}_i$ is the trace obtained from the device for profiling, and N_P denotes the number of traces in the profiling phase. In DL-based profiling attacks, the value (label) predicted by the DNN is the HW (or HD) computed using the secret key and plaintext. Therefore, the label is often set to $l_i = \mathrm{HW}(\mathrm{Sbox}(k_i \oplus m_i))$; that is, in the profiling phase, we train an NN to predict the probability of the HW/HD determined by the intermediate value from the corresponding trace $\boldsymbol{x}$. If byte-wise prediction is used, the output of the NN is a nine-class probability obtained from the softmax layer because the HW/HD takes values from zero to eight. Note that the value of k_i is not important for the training because the NN estimates the HW/HD of the intermediate value.

The purpose of the profiling phase is to train an NN to estimate the correct label with high probability from the input traces in the following attack phase. Let θ represent the model parameters of the NN (e.g., the weights of a multilayer perceptron or CNN) and let $q(l \mid \boldsymbol{x}; \theta)$ be the output probability represented by the NN. In the profiling phase, θ is determined such that the probability $q(l \mid \boldsymbol{x}; \theta)$ matches the true probability $p(l \mid \boldsymbol{x})$. Now, let $\mathcal{X}$ denote the set of possible values of trace $\boldsymbol{x}$ and $\mathcal{L}$ denote the set of possible values of label l; then, we can formulate the determination of θ as the minimization problem of the cross-entropy function

$$\begin{aligned} H(p, q) &= \mathop{\mathbb{E}}_{(l,\boldsymbol{x})\sim p} - \log(q(l \mid \boldsymbol{x}; \theta)) \\ &= -\int_{\mathcal{X}} \sum_{l\in\mathcal{L}} p(l, \boldsymbol{x}) \log\left(q(l \mid \boldsymbol{x}; \theta)\right) d\boldsymbol{x}, \end{aligned} \tag{2}$$

where $(l, \boldsymbol{x}) \sim p$ means the (random) variables l and $\boldsymbol{x}$ follow the true distribution p. Because the cross-entropy function takes its minimum value only when $q = p$, we should determine θ such that the function is minimized. However, (2) cannot be evaluated directly because it involves the integral and summation of an unknown distribution $p(l, \boldsymbol{x})$. Therefore, we approximate it with a finite number of sample points (i.e., training data) as follows:

$$\begin{aligned} H(p, q) &\approx \mathrm{NLL}(q) \\ &= \frac{1}{N_p} \sum_{i=1}^{N_p} - \log(q(l_i \mid \boldsymbol{x}_i; \theta)). \end{aligned} \tag{3}$$

The right-hand side of (3) is called NLL, and it is commonly used in machine learning [16], [17]. It is well known that (3) converges in probability towards (2).

The intuitive meaning of (3) becomes clear by transforming the NLL function as follows:

$$\mathrm{NLL}(q) = -\frac{1}{N_p} \log\left(\prod_{i=1}^{N_p} q(l_i \mid \boldsymbol{x}_i; \theta)\right), \tag{4}$$

where the joint probability with the argument of the log function is called the likelihood function. Here, let us consider the case in which trace $\boldsymbol{x}_i$ is observed. If the intermediate value is calculated from the corresponding key value k_i and plaintext m_i, then we obtain the likelihood as a HW/HD by l_i, which can be estimated from the trace. Moreover, this likelihood suggests that the estimated probability $q(l_i \mid \boldsymbol{x}_i; \theta)$ should be large. This also suggests that the true probability distribution $p(l \mid \boldsymbol{x})$ has a high probability at $p(l_i \mid \boldsymbol{x}_i)$.

In the attack phase, an attacker uses the NLL value derived from the likelihood function to estimate a key. That is, the sequence of labels $l_1, l_2, \ldots, l_{N_A}$ calculated for a key candidate k with different plaintexts must be plausibly estimated from actually observed traces $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_A}$, where N_A indicates the number of traces observed during the attack phase. The NLL for key candidate k can be written as

$$\mathrm{NLL}_k(q, \hat{\theta}) = -\frac{1}{N_A} \sum_{j=1}^{N_A} \log q(l = g(k, m_j) \mid \boldsymbol{x}_j; \hat{\theta}), \tag{5}$$

where the estimated parameter $\hat{\theta}$ is given from the preceding profiling phase and function g outputs the HW/HD of the intermediate value from the key and plaintext. The likelihood function in (5) is expected to have a maximum value when the correct key k^* is used. Thus, the attacker computes the likelihoods for all the key candidates in advance and estimates that the key with the largest likelihood value is the correct key.

B. Imbalanced Data Problem

In this section, we briefly describe the imbalanced data problem of DL-based SCAs and explain the previous methods for solving it. The occurrence probability of labels such as HW/HD that are used in profiling attacks, is obviously imbalanced. Data in which the occurrence probability of the labels deviates from the uniform distribution are called imbalanced data. It has been noted that profiling attacks are often difficult to learn and infer because of the effects of imbalanced data [12]. In fact, it was shown in [12] that most metrics used in conventional machine learning, such as accuracy and precision, are not useful in SCAs because of the imbalanced data. The imbalanced data problem has often been solved in machine learning by augmenting the minority class data, removal of majority class data, or the introduction of a penalty term to a loss function [18].

In SCAs, the effectiveness of SMOTE, which is a general-purpose data augmentation technique, has also been experimentally demonstrated in [12]. In contrast, the low-quality samples added during data augmentation can lead to low accuracy because the actual probability distribution of the observed data is generally not known during augmentation. In addition, because such data augmentation may artificially increase the number of minority data, there is a possibility that it could result in different data distributions during training and inference and substantially affect the success rate (SR) of SCA. For these reasons, a quantitative analysis on the effect of data augmentation in profiling SCAs is required.

Another approach to solving the imbalanced data problem uses a recent indicator called the cross-entropy ratio (CER),[2]

[2] A formal definition of CER is given in Section IV.

which was reported and evaluated in [14]. CER is defined as the ratio of the expected value of the log-likelihoods for all wrong keys to the log-likelihood of the correct key. Therefore, it is expected that minimizing the CER is equivalent to making the NLLs of other key candidates larger than the NLL of the correct key, which can help estimate the correct key. In [14], the SR converged to 1 in probability when the CER was less than 1.[3] However, the proof validating CER in the literature is based on an assumption that does not hold in practice [15]. As a result, the rationale for the CER loss is still unclear.

III. Evaluation of the Imbalanced Data Problem

In this section, we first state the reasons for the negative effects of imbalanced data for DL-SCAs in a qualitative manner, and then use the KL divergence to quantify the degree of these effects.

A. Negative Effects of Imbalanced Data

The negative effects of imbalanced data in DL-SCAs originate from the following two factors specific to SCAs.

1) There is not always a strong relationship between the side-channel information (e.g., the EM radiation and power consumption) and the intermediate value of interest.
2) The occurrence probability of labels (i.e., the HW/HD) used for SCAs commonly follows a binomial distribution.

First, classification tasks in conventional machine learning (e.g., image and speech recognition) usually assume that a given input (e.g., image and speech waveforms) generally contains sufficient information for classification. The class labels for these tasks are clearly interpretable and explainable by humans, and in this sense, most data have clear criteria for classification. However, in the case of SCAs, it is not always possible to uniquely identify the intermediate value from the side-channel information for some specific reasons (e.g., the algorithmic noise, the presence of countermeasures, and/or a lower signal-to-noise ratio (SNR) at the measurement). As a result, the side-channel information of a single trace may be insufficient for estimating the values of the HW/HD. If the side-channel information and the corresponding intermediate value are close to independent, the probability distribution to be estimated by the NN (i.e., $p(l \mid \boldsymbol{x})$) is close to $p(l)$. Second, the HW/HD of uniformly distributed random values follows a binomial distribution; hence, $p(l) = \mathrm{Bin}(l)$, where $\mathrm{Bin}(l)$ is a binomial distribution. Therefore, the true probability distribution is sometimes very close to $p(l \mid \boldsymbol{x}) \approx \mathrm{Bin}(l)$. Because of these two factors, the output distribution of an NN is often heavily distorted towards the binomial distribution, which is a major issue when using machine learning techniques in SCAs.

To explain how the bias induced by the binomial distribution (i.e., the imbalanced data problem) affects DL-based SCAs, consider a case in which the S-box output of the AES is used as the intermediate value. In this case, the NN output distribution (i.e., $q(l = g(k, m_i) \mid \boldsymbol{x}_i; \theta)$) represents the conditional probability of the HW of the intermediate value given trace $\boldsymbol{x}_i$. Here, the HW value is given as an integer from 0 to 8 according to the output of the S-box. To estimate the correct key k^*, the output distribution of the NN should assign a high probability to the correct label $g(k^*, m_i)$. However, when the output distribution is close to a binomial distribution, the NN output distribution is strongly biased towards the binomial distribution, especially when the relationship between the trace and the HW value is weak. In this case, independent of the correct label (i.e., the HW of the S-box output with a correct key guess) for a given trace, the NN output probability is biased such that the probability of $\mathrm{HW} = 4$ is the highest and that of $\mathrm{HW} = 0$ or 8 is the lowest.

Moreover, when we estimate the S-box output for all key candidates given a specific trace, the hypothetical HW for each key candidate is determined by a binomial distribution depending on the plaintext. If an infrequent label (e.g., $\mathrm{HW} = 0$ or 8) is guessed with the correct key $g(k^*, m_i)$, the estimated likelihood of the correct key is lower than that of the wrong key guesses that include frequent labels (e.g., $\mathrm{HW} = 4$). Such effects are not a problem in the ideal case, where an infinite number of traces are available for the attack phase, because all the key candidates will have the same amount of bias.[4] However, this negative effect is non-trivial in practice because the number of available traces is finite. In particular, the effect increases as the number of traces available for an attack decreases.

B. Quantitative Evaluation Metric

For quantitative evaluation of the effect of imbalanced data, we employ the KL divergence as an evaluation metric. The basic idea of our metric is to address the problems mentioned in the previous section by evaluating the distance between the conditional probability $p(l \mid \boldsymbol{x})$ and occurrence probability of labels $\mathrm{Bin}(l)$ (i.e., the binomial distribution). The KL divergence is given as a function that takes two probability distributions and returns a real number greater than or equal to zero (the KL divergence equals zero if and only if the two distributions are identical) [16].

The KL divergence between $p(l \mid \boldsymbol{x})$ and $\mathrm{Bin}(l)$ is defined as follows:

$$D_{\mathrm{KL}}(\mathrm{Bin} \mid\mid p) = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p} \mathop{\mathbb{E}}_{l \sim \mathrm{Bin}} \log\left(\frac{\mathrm{Bin}(l)}{p(l \mid \boldsymbol{x})}\right)$$
$$= \int_{\mathcal{X}} p(\boldsymbol{x}) \sum_{l \in \mathcal{L}} \mathrm{Bin}(l) \log\left(\frac{\mathrm{Bin}(l)}{p(l \mid \boldsymbol{x})}\right) d\boldsymbol{x}. \quad (6)$$

This equation cannot be computed directly because probability distribution $p(l \mid \boldsymbol{x})$ is generally unknown. Instead, probability $p(l \mid \boldsymbol{x})$ is replaced by estimated probability $q(l \mid \boldsymbol{x}; \theta)$ of NNs. In addition, the expected value for the side-channel information $\boldsymbol{x}$ is approximated from the finite number of

[3] This means that given an infinite number of traces at the time of key estimation, SR will be 1 with infinitely high probability.

[4] For any key candidate, the number of traces when $\mathrm{HW} = 0$ converges to $\frac{1}{256}$ of all the traces used in the attack phase. Similarly, when $\mathrm{HW} = 1$, the number of traces asymptotically converges to $\frac{8}{256} = \frac{1}{8}$ of all traces. These ratios are determined according to the binomial distribution.

samples obtained during the profiling phase. Using these approximations, (6) can be simplified to

$$\hat{D}_{\mathrm{KL}}(\mathrm{Bin} \parallel q) = \frac{1}{N_P} \sum_{i=1}^{N_P} \sum_{l \in \mathcal{L}} \mathrm{Bin}(l) \log \left(\frac{\mathrm{Bin}(l)}{q(l \mid \boldsymbol{x}_i; \theta)} \right). \quad (7)$$

Here, binomial distribution $\mathrm{Bin}(l)$ is given by

$$\mathrm{Bin}(l) = \frac{\binom{r}{l}}{2^r}, \quad (8)$$

where r denotes the bit length of the target intermediate value ($r = 8$ in standard SCAs on AES) and $\binom{r}{l}$ is the binomial coefficient.

A smaller KL divergence indicates that the NN is more affected by imbalanced data because $p(l \mid \boldsymbol{x})$ (i.e., the distribution of the NN output) is close to the binomial distribution. Here, an SCA is impossible in principle when the KL divergence is zero because $p(l \mid \boldsymbol{x}) = \mathrm{Bin}(l) \Leftrightarrow L \perp\!\!\!\perp \boldsymbol{X}$, where L and $\boldsymbol{X}$ denote the random variables of the label and trace, respectively. This indicates that the attacker can obtain no information about the HW/HD of intermediate values from side-channel traces. In contrast, a one-hot vector-like distribution has a high KL divergence,[5] which is an ideal NN output (if the NN is well-trained and sufficiently accurate). In fact, as the KL divergence increases, the NN output becomes closer to a one-hot vector-like distribution, where the likelihood of a correct label is far greater than those of the wrong labels. Thus, the evaluation of KL divergence values can estimate the difficulty of DL-based SCAs. In other words, we can efficiently perform a DL-based SCA by training an NN such that the KL divergence increases. Moreover, the use of the CER loss corresponds to such a training approach, as noted in the next section.

IV. Analysis of the CER Loss Using the KL Divergence

In this section, we explain why the CER loss mitigates the imbalanced data problem in DL-SCAs using the KL divergence introduced above. It is mentioned in [14] that the reason for effectiveness of the CER loss for imbalanced data problems is the success of profiling SCA for CER < 1. However, the attack assumption, which is that the intermediate values of the correct and wrong key candidates are independent, does not always hold [15]. In other words, there must be another reason for the CER loss to be able to mitigate the effects of the imbalanced data problem in DL-based SCAs. To explain this reason, we first show that the increase in the KL divergence between the binomial distribution and the model's output during the training phase helps to reduce the negative effects of imbalanced data. We then show that the CER loss is an example of this case, that is, training a model using the CER loss substantially increases the KL divergence.

A. Effect of Increasing KL Divergence

As described in Section III, the imbalanced data problem in DL-based SCAs comes from the bias (which follows the binomial distribution) in the probabilities of label occurrence, and results in an over-estimation of an incorrect label's probability. This suggests that the imbalanced data problem can be solved if a model is trained to eliminate this bias. Such training can be achieved by maximizing the KL divergence in addition to the usual minimization of the NLL loss function.

To explain the effect of maximizing the KL divergence precisely, consider the following deformation of (7).

$$\hat{D}_{\mathrm{KL}}(\mathrm{Bin} \parallel q) = \sum_{l \in \mathcal{L}} \mathrm{Bin}(l) \log(\mathrm{Bin}(l)) - \frac{1}{N_P} \sum_{i=1}^{N_P} \sum_{l \in \mathcal{L}} \mathrm{Bin}(l) \log(q(l \mid \boldsymbol{x}_i; \theta)). \quad (9)$$

In this equation, we focus on the second term because the first one is a constant value (i.e., the entropy of binomial distribution). The term allow us to confirm that maximizing the KL divergence makes output probability $q(l \mid \boldsymbol{x}_i; \theta)$ close to zero; maximization of the KL divergence decreases the probabilities of frequent labels (e.g., HW = 4) more heavily than those of infrequent labels because (9) contains the inner product of the binomial distribution and $q(l \mid \boldsymbol{x}_i; \theta)$. This means that training a model to increase the KL divergence can reduce the negative effects of class imbalance. Note that $\forall l \in \mathcal{L}, q(l \mid \boldsymbol{x}_i; \theta) \neq 0$ holds because the sum of $q(l \mid \boldsymbol{x}_i; \theta)$ for all l must be one.

B. Relationship Between the CER Loss and KL Divergence

We first describe the formulation of the CER loss. As noted in Section II-A, the cross-entropy function shown in (2) is an objective function for rendering an NN output probability $q(l \mid \boldsymbol{x}; \theta)$ close to true probability $p(l \mid \boldsymbol{x})$. In addition, the NLL in (3) is alternatively used in practice for evaluating the cross-entropy function. Here, the NLLs calculated with a wrong key guess do not necessarily converge to (2) in probability because the NLL in (3) is defined as an approximation of the cross-entropy function where the labels are calculated with the correct key k^*. If we compute the NLLs from labels with a wrong key guess, we should take into account that these labels have been sampled from a distribution that is different from the distribution of true probability $p(l \mid \boldsymbol{x})$.

Let $p_k(l, \boldsymbol{x})$ be the probability distribution of labels for a key candidate k. We define the cross entropy between $p_k(l, \boldsymbol{x})$ and the output probability of NN as

$$H(p_k, q) = \mathop{\mathbb{E}}_{(l,\boldsymbol{x}) \sim p_k} - \log(q(l \mid \boldsymbol{x}; \theta)) = \int_{\mathcal{X}} \sum_{l \in \mathcal{L}} -p_k(l, \boldsymbol{x}) \log(q(l \mid \boldsymbol{x}; \theta)) d\boldsymbol{x}, \quad (10)$$

where we assume that $\mathrm{NLL}_k(q)$ converges in probability towards (10) when an infinite number of samples are given. In addition, p_k is equal to p if $k = k^*$. With p_k, the CER can

[5] If the shape of $p(l \mid \boldsymbol{x})$ is like a one-hot vector, $p(l \mid \boldsymbol{x})$ takes a very small value when $l \neq l^*$, where l^* is the correct label. The KL divergence becomes large in this case because the argument of the log function in (6) is increased.

be defined as

$$\mathrm{CER}(q) = \frac{H(p_{k^*}, q)}{\mathop{\mathbb{E}}_{k \neq k^*} H(p_k, q)}. \tag{11}$$

Using the above definition, we analyze the CER loss minimization from the viewpoint of KL divergence. We now focus on the denominator of the CER because the numerator is equivalent to the common cross entropy. The expected value is equal to the average, and therefore the denominator is rewritten as

$$\begin{aligned}\mathop{\mathbb{E}}_{k \neq k^*} H(p_k, q) &= \frac{1}{|\mathcal{K}| - 1} \sum_{k \neq k^*} H(p_k, q)\\ &= \frac{|\mathcal{K}|}{|\mathcal{K}| - 1} \mathop{\mathbb{E}}_{k} [H(p_k, q)] - \frac{1}{|\mathcal{K}| - 1} H(p_{k^*}, q),\end{aligned} \tag{12}$$

where $\mathcal{K}$ is the set of all key candidates (i.e., $\mathcal{K} = \{k \mid 0 \leq k \leq 255, k \in \mathbb{N}\}$ for typical SCAs on AES), and $|\mathcal{K}|$ denotes the number of key candidates (i.e., 256). The coefficient of the first term $|\mathcal{K}|/(|\mathcal{K}|-1)$ is close to one. In addition, the second term is far smaller than the first term because the NN is trained such that $H(p_{k^*}, q)$ of the second term (i.e., the cross entropy for the correct key) is minimized. The value of $H(p_{k^*}, q)$ is also divided by $|\mathcal{K}| - 1 (i.e., 255)$. Thus, (12) can be approximated as

$$\mathop{\mathbb{E}}_{k \neq k^*} H(p_k, q) \approx \mathop{\mathbb{E}}_{k} H(p_k, q). \tag{13}$$

As a result, the cross entropy of the right-hand side of (13) can be expanded with respect to plaintext m as follows:

$$\begin{aligned}H(p_k, q) &= \mathop{\mathbb{E}}_{(l, \boldsymbol{x}) \sim p_k} - \log(q(l \mid \boldsymbol{x}; \theta))\\ &= -\int_{\mathcal{X}} \sum_{l \in \mathcal{L}} p_k(l, \boldsymbol{x}) \log(q(l \mid \boldsymbol{x}; \theta)) d\boldsymbol{x}\\ &= -\int_{\mathcal{X}} \sum_{m \in \mathcal{M}} p_k(m, \boldsymbol{x}) \log(q(l = g(k, m) \mid \boldsymbol{x}; \theta)) d\boldsymbol{x}\\ &= -\int_{\mathcal{X}} \sum_{m \in \mathcal{M}} p(m, \boldsymbol{x}) \log(q(l = g(k, m) \mid \boldsymbol{x}; \theta)) d\boldsymbol{x}\\ &= \mathop{\mathbb{E}}_{(m, \boldsymbol{x}) \sim p} - \log(q(l = g(k, m) \mid \boldsymbol{x}, \theta)).\end{aligned} \tag{14}$$

Here, $\mathcal{M}$ is the set of all plaintexts, and $p_k(m, \boldsymbol{x}) = p(m, \boldsymbol{x})$ because the relationship between the plaintext and side-channel information sampled from the device does not depend on the hypothetical key value. Therefore, the cross-entropy function can be rewritten as

$$\begin{aligned}\mathop{\mathbb{E}}_{k} H(p_k, q) &= - \mathop{\mathbb{E}}_{(m, \boldsymbol{x}) \sim p} \mathop{\mathbb{E}}_{k} \log(q(l = g(k, m) \mid \boldsymbol{x}, \theta))\\ &= - \mathop{\mathbb{E}}_{(m, \boldsymbol{x}) \sim p} \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \log(q(l = g(k, m) \mid \boldsymbol{x}, \theta)).\end{aligned} \tag{15}$$

We then separate the set of key candidates into subsets as follows:

$$\mathcal{K} = \bigcup_{l \in \mathcal{L}} \mathcal{K}_l, \tag{16}$$

where $\mathcal{K}_l = \{k \mid l = g(k, m)\}$. Because $|\mathcal{K}_l| = \binom{r}{l}$ and $g(k, m)$ is constant against l, (15) can be represented as follows:

$$\mathop{\mathbb{E}}_{k} H(p_k, q) = D_{\mathrm{KL}}(\mathrm{Bin} \mid\mid q) + H(\mathrm{Bin}), \tag{17}$$

where $H(\mathrm{Bin})$ denotes the entropy of the binomial distribution. Consequently, the CER is approximated as

$$\mathrm{CER}(q) \approx \frac{H(p, q)}{D_{\mathrm{KL}}(\mathrm{Bin} \mid\mid q) + H(\mathrm{Bin})}. \tag{18}$$

Thus, the minimization of the CER loss is equivalent to maximizing the KL divergence between the binomial distribution and NN output distribution, and this explains the reason for the effectiveness of CER loss for training with imbalanced data.

V. Solving the Imbalanced Data Problem During Inference

In this section, we propose an effective solution for the imbalance data problem that is implemented during the inference phase (i.e., during the secret key estimation in the attack phase). The basic idea of our method is to estimate the secret key using the likelihood function based on the key value probability (and inferred HW/HD) instead of the conventional HW/HD-based likelihood function. The use of the key-based likelihood removes the bias of binomial distribution in the output distribution of the NNs during inference. Note that we still use the NN for inferring the HW/HD; the NN does *not* directly infer the 256 class labels that represent key candidates as its output. In the proposed method, the network first classifies a trace into $r+1$ labels representing the HW/HD (e.g., $r = 8$ in the typical case of DL-SCAs on AES) and then estimates the correct key using the key-based likelihood.

The existing solutions at the training phase described in the previous sections, which forcibly increase the KL divergence between the binomial distribution and the model output, may move the output distribution of the model further away from the true distribution p. In contrast, the proposed solution makes the occurrence probabilities of class labels uniform. This is similar to the approach in data augmentation; however, our solution preserves dataset quality and cause no difference in the distributions of the training and test data. This section first presents the formulation and algorithm of the proposed method, and then describes the relationship between conventional data augmentation and the proposed method.

A. Inference Using Key-Based Likelihoods

The proposed key estimation method employs the NLL function of key value probability instead of the NLL of the HW/HD. As mentioned in Section III, when the network output distribution is almost equal (or quite similar) to a binomial distribution (i.e., when the KLD is close to zero), it has a strong negative effect on conventional key estimation using (5) because the conventional method does not compensate for any bias. In other words, we cannot avoid the imbalanced data problem as long as (5) (i.e., the HW/HD-based likelihood of NLL) is used to estimate the key. To address this problem, we focus on the key value probability, which should always be uniformly distributed in the context of SCAs.

The NLL of key value probability (key-based NLL or KNLL) is defined as follows:

$$\mathrm{KNLL}_k(p) = -\frac{1}{N_A}\sum_{j=1}^{N_A} \log p(k \mid m_j, \boldsymbol{x}_j), \tag{19}$$

where k denotes the key candidate and N_A represents the number of traces used in the attack phase. Note here that the key value probability $p(k \mid m_j, \boldsymbol{x}_j)$ has no inherent bias, unlike the HW/HD probability.

To calculate the key value probability from the model's output, we rewrite (19) with the joint probability $p(k, l \mid m_j, \boldsymbol{x}_j)$. Let l_j be the label determined by m_j and k. Equation (19) is then given as follows:

$$\begin{aligned}\mathrm{KNLL}_k(p) &= -\frac{1}{N_A}\sum_{j=1}^{N_A} \log p(k, l_j \mid m_j, \boldsymbol{x}_j)\\ &= -\frac{1}{N_A}\sum_{j=1}^{N_A} \log p(k \mid m_j, l_j, \boldsymbol{x}_j) p(l_j \mid \boldsymbol{x}_j)\\ &= -\frac{1}{N_A}\sum_{j=1}^{N_A} \log p(k \mid m_j, l_j) p(l_j \mid \boldsymbol{x}_j), \end{aligned}\tag{20}$$

where we utilize the fact that a key value and a trace have conditional independence given a label and a plaintext (i.e., $p(k \mid m_j, l_j, \boldsymbol{x}) = p(k \mid m_j, l_j)$). Here, $p(k \mid m, l)$ is the probability that a value of k is selected from the uniform distribution given an HW/HD, and therefore is given by the inverse of the binomial coefficient, i.e.,

$$p(k \mid m, l) = \begin{cases} \dfrac{1}{\binom{r}{g(k,m)}} & (l = g(k, m)) \\ 0 & (\text{otherwise}) \end{cases}, \tag{21}$$

where r is the bit length of the intermediate value. The posterior probability of the class label given a trace (i.e., $p(l_j \mid \boldsymbol{x}_j)$) can be approximated by the output distribution of the NN (i.e., $q(l_j \mid \boldsymbol{x}_j; \hat{\theta})$) as described in Section III, and therefore (20) is estimated as follows:

$$\begin{aligned}\mathrm{KNLL}_k(p) &\approx \mathrm{KNLL}_k(q, \hat{\theta})\\ &= -\frac{1}{N_A}\sum_{j=1}^{N_A} \log \frac{q(l = g(k, m_j) \mid \boldsymbol{x}_j; \hat{\theta})}{\binom{r}{g(k,m_j)}}. \end{aligned}\tag{22}$$

Consequently, in contrast to the conventional HW/HD-based NLL of (5), (22) includes the inverse of the binomial coefficients, which indicates that the KNLL cancels out the bias of the binomial distribution. Therefore, (22) can efficiently estimate the correct key even when the probability estimated by the NN $q(l \mid \boldsymbol{x}; \hat{\theta})$ is close to the binomial distribution. In other words, the use of KNLL essentially solves the imbalanced data problem even when the traces do not contain sufficient information about the intermediate value of interest.

Fig. 1 illustrates the process of our method when the correct label is zero for a trace and the correct key value is 47. Ideally, the probability of the correct HW/HD label in the NN output should be much higher than that of the others, as in a one-hot vector. However, in practice, the network output will be close to a binomial distribution, as Fig. 1(a) shows, in which the probability of the correct label (red) is not necessarily high. To address this problem, we first divide the NN output probabilities by the binomial coefficients, as shown in Fig. 1(b). Note that the sum of the divided outputs is not equal to one at this stage. Next, we distribute these divided outputs to all the key candidates, as shown in Fig. 1(c); the probability of each key candidate is given accordingly. As a result, the sum of probabilities for all key candidates becomes one, and we obtain the key value probabilities for the trace. Finally, we calculate KNLL_k using the key value probabilities given by the traces of the attack phase.

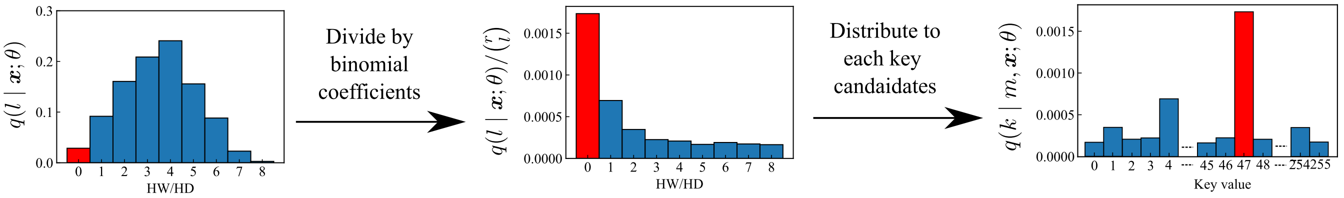
Algorithm 1 Secret key estimation using KNLL

```
Require: Data for profiling: $\mathcal{S}_P$, Data for attack: $\mathcal{S}_A$, Set of key candidates: $\mathcal{K}$, Bit-length of intermediate value: $r$
Ensure: Estimated secret key: $k'$
 1: $\theta \leftarrow \mathrm{Train}(\mathcal{S}_P)$
 2: for $k \in \mathcal{K}$ do
 3:     $\mathrm{KNLL}_k \leftarrow 0$
 4: end for
 5: $\mathrm{invcoef} = [1/\binom{r}{0}, 1/\binom{r}{1}, \ldots, 1/\binom{r}{r}]$
 6: for $k \in \mathcal{K}$ do
 7:     for $(m, \boldsymbol{x}) \in \mathcal{S}_A$ do
 8:         $l \leftarrow g(k, m)$
 9:         $\mathrm{KNLL}_k \leftarrow \mathrm{KNLL}_k - (\log(q(l \mid \boldsymbol{x}; \theta)) + \log(\mathrm{invcoef}[l]))$
10:     end for
11:     $\mathrm{KNLL}_k \leftarrow \mathrm{KNLL}_k / |\mathcal{S}_A|$
12: end for
13: $k' \leftarrow \arg\min_k \mathrm{KNLL}_k$
14: return $k'$
```

Algorithm 1 shows the pseudo code of the proposed method, where the inputs are the data for profiling and attack phases, and the output is the estimated secret key k' obtained by the model. In Line 1, we train the model using the profiling data with the conventional HW/HD-based NLL. In Lines 2–5, we initialize two variables KNLL_k and invcoef. Here $\mathcal{K}$ is the key space, KNLL_k is a variable to store the key-based NLL of a key candidate k, and invcoef is the array of inverted binomial coefficients. In Lines 6–12, we calculate the key-based NLL of each key candidate using the trained model. Note that this algorithm is the same as conventional HW/HD-based secret key estimation except for the subtraction of the logarithm in invcoef (i.e., $-\log(\mathrm{invcoef}[l])$). Finally, in Lines 13–14, we return the key candidate with the smallest key-based NLL as the estimated key k'. Hence, our method is easily implemented using only small modifications to the conventional inference phase.

B. Relationship to Data Augmentation

We explain the relationship between data augmentation techniques and the proposed method based on the key-based likelihood. As an example, we consider SMOTE, which is one of the most well-known data augmentation algorithms for increasing minority data when training an NN [19]. In machine



(a) NN output

(b) Divided NN output

(c) Key value probability

Fig. 1. Illustration of the proposed method: (a) NN output, (b) NN output probabilities divided by the binomial coefficients, and (c) key value probability (The values in this figure are not real values by experiment, but are derived from hypothetical models).

learning, various data augmentation algorithms have been presented based on SMOTE. SMOTE first randomly selects a sample from the minority label data and then selects a other samples in the vicinity of the sample, where a is a predetermined parameter given as an integer. Next, a point between these a randomly selected neighborhood points and the first sample point is added to the data as a new point. This procedure is repeated until the occurrence probabilities of the labels are equalized.

As shown in [15], a SMOTE-based data augmentation could also improve DL-based SCAs if the side-channel information does not contain much information about the HW/HD. One of the major reasons this would work is that SMOTE-based data augmentation would help move the network output distribution further from a binomial one and closer to a uniform distribution. This can be described by Bayes' theorem as follows:

$$p(l \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid l)}{p(\boldsymbol{x})} p(l). \tag{23}$$

When the relationship between the side-channel information and the label is weak, the right-hand side of (23) is equivalent to label occurrence probability $p(l)$ under the approximation of $\frac{p(\boldsymbol{x}|l)}{p(\boldsymbol{x})} \approx 1$. This suggests that SMOTE can mitigate the negative effects of imbalanced data by making the occurrence probability $p(l)$ uniform. We note that the use of the proposed key probability corresponds to dividing both sides of (23) by $p(l)$, and therefore it can reduce the negative effect of imbalanced data. Thus, the two methods would have a similar effect on the imbalanced data problem.

However, data augmentation methods such as SMOTE never consider the specific features of side-channel information, such as the jitter included in traces [9], and do not guarantee the quality of the added data because the true distribution of the data is generally unknown. As a result, added samples may be far from actual traces and may not necessarily be effective for DL-SCAs. In fact, it was reported that a SMOTE-like data-augmentation technique sometimes decreased the performance of a model because it often added unnatural samples [20], [21]. In addition, samples added by the interpolation of minority class samples could make a model overfit the data. In contrast, the proposed method does not require any additional samples; therefore, the trained NN is not affected by the dataset quality. The advantages of the proposed method over SMOTE-based data augmentation are experimentally demonstrated in Section VI.

VI. Experimental Evaluation

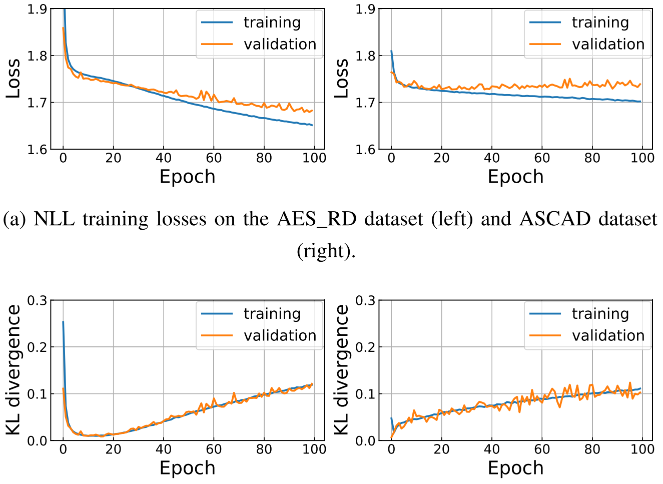
A. Experimental Setup

In this section, we present the results of our evaluation of the proposed method and the arguments presented in this paper through a set of experiments. We employed the following two datasets for the evaluation of SCAs.

1) AES_RD Dataset: The AES_RD dataset contains power traces obtained from a software 128-bit AES implemented on an 8-bit AVR microcontroller with a random-delay-based countermeasure [22]. The countermeasure consists of an insertion of instructions to randomly generate delay, which dynamically changes the point of interest and reduces the signal-to-noise ratio to an extremely low value. In this experiment, the 50,000 traces in the AES_RD dataset are divided into two sets of 25,000 traces. One set is used for training, and another is used for testing.

2) ASCAD: The ASCAD dataset [23] is a public dataset for assessing SCAs using machine learning and has been used in many previous studies on DL-based SCAs [14], [24]. The ASCAD dataset contains power traces of AES software implementation running on ATmega8515 with a Boolean-masking to counter first-order attacks. The dataset consists of two subsets: variable-key and fixed-key subsets. Similar to many previous studies, we used the fixed-key set in this study. The number of traces in the fixed-key set was 50,000 for training and 10,000 for testing. In addition, 10% of the traces used for training were used for validation, and the remainder were used for practical training (i.e., updating the model parameters).

The proposed and conventional DL-based SCAs were applied to the datasets under the following conditions. We employed a model recommended in [24] because the network architecture is dependent on the dataset. The Adam optimizer [25] was used with a learning rate of 0.0001. The batch size and the number of training epochs were basically set to 50 and 100, respectively. The trained model was used at each epoch for key estimation because the purpose of this experiment is to show the relationships among the following metrics in SCAs: the NLL loss, KL divergence, and SR. The labels are given as the HWs of the first-round AES S-box output computed from the plaintext and secret key. To calculate SRs, we estimated the sub-key value 1,000 times and counted the number of times when the first rank is that of the correct key. At each evaluation, we choose 500 traces from the test data randomly and uniformly. We used Intel Xeon W-2145 CPU with 128 GB memory, GeForce GTX 2080 Ti, and TensorFlow 2.4.1 for the experiments.



(a) NLL training losses on the AES_RD dataset (left) and ASCAD dataset (right).

(b) KL training divergences on the AES_RD dataset (left) and ASCAD dataset (right).

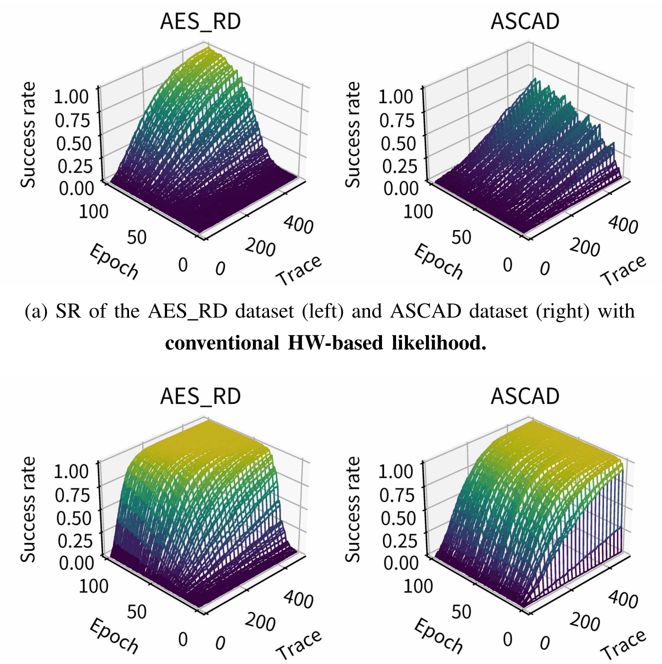
Fig. 2. NLL loss and KL divergence.

B. Comparison of Likelihoods

We first evaluated the proposed method and arguments in Sections III and V through an experimental attack using the key- and HW-based likelihoods. The conventional (HW-based) NLL loss was used for training the NN. Note here that the learning rate for ASCAD was set to 0.005 only in this experiment in order to avoid underfitting the training set. The training times for AES_RD and ASCAD datasets were 278 and 300 seconds, respectively.

Fig. 2 shows the NLL loss and KL divergence in training the models, where the horizontal axes represent the number of epochs, and the vertical axes represent the values of loss and KL divergence in Figs. 2(a) and (b), respectively. Fig. 3 shows the SRs of the trained models on AES_RD and ASCAD datasets with (a) the HW-based likelihood and (b) the proposed key-based likelihood for various numbers of epochs and traces. Fig. 4 shows an example of the cross-section view of Fig. 3 for 500 traces.

From Fig. 2(b), we confirm that the KL divergence is very small at the beginning of training and gradually increases as the training progresses for both the AES_RD and ASCAD datasets. This indicates that the model first fits the binomial distribution at the early stage of training, and then trains such that the probability of the correct label increases. The results validate our arguments 1) and 2) in Section III; for these reasons, the KL divergence is initially very close to the binomial distribution. In addition, Fig. 2(a) shows that the validation loss of the trained model for ASCAD is the lowest at approximately 20 epochs and then increases gradually. This increase shows the model is overfitting. However, the SR on ASCAD dataset with the HW-based NLL continues to improve even after 20 epochs, as shown in Fig. 3(a). This indicates that overfitting cannot always be negative in the case of DL-based SCAs. This further validates our argument; as the KL divergence increases even after 20 epochs, as shown in Fig. 2(b), the model overfitting reduces the negative effect of imbalanced data by moving the model output far away from a binomial distribution. More precisely, for the HW-based NLL, the negative effect of the imbalanced data may be higher than that of overfitting; and therefore, the increase in KL divergence can improve the attack performance even under overfitting.



(a) SR of the AES_RD dataset (left) and ASCAD dataset (right) with **conventional HW-based likelihood.**

(b) SR of the AES_RD dataset (left) and ASCAD dataset (right) with the **proposed key-based likelihood.**

Fig. 3. Success rate of the conventional and proposed methods.

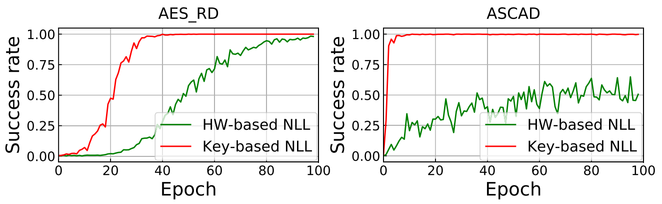
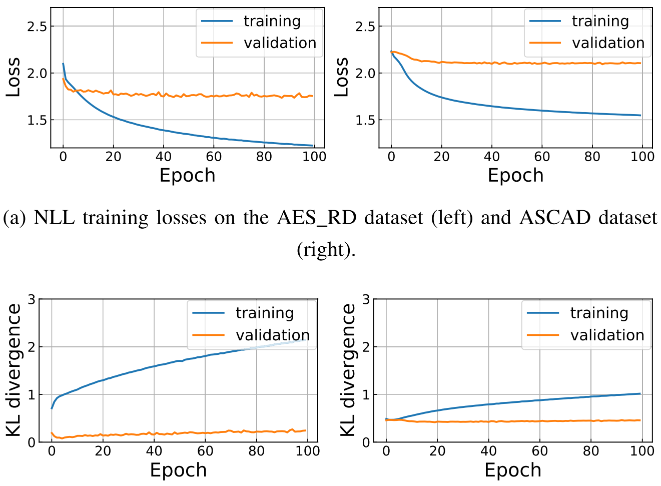


Fig. 4. Success rate of the conventional and proposed methods for the AES_RD dataset (left) and ASCAD dataset (right) for 500 traces.

Next, we confirm from Figs. 3 and 4 that the SR result of the proposed key-based likelihood is clearly superior to that of the HW-based likelihood. The results show that the key estimation using KNLL is better than HW-based NLL for any value of KL divergence. In addition, Fig. 4 shows that smaller values of KL divergence lead to larger differences in the SR values of the HW-based likelihood and key-based likelihood. This suggests that the proposed KNLL has a larger advantage over the conventional HW-based NLL on DL-based SCAs when the output distribution of the model is closer to the binomial distribution.

C. Analyzing Data Augmentation Method

We then analyze the effect of data augmentation from the viewpoints of KL divergence and the NN output probability. In this experiment, we used SMOTE as a typical data augmentation method and used the implementation provided in an open-source library "imbalanced-learn" as a conventional



(a) NLL training losses on the AES_RD dataset (left) and ASCAD dataset (right).

(b) KL training divergences on the AES_RD dataset (left) and ASCAD dataset (right).

Fig. 5. NLL loss and KL divergence with SMOTE.

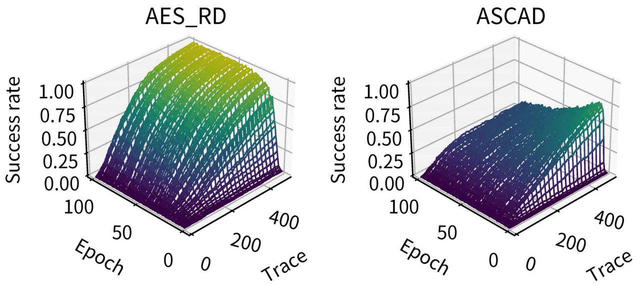
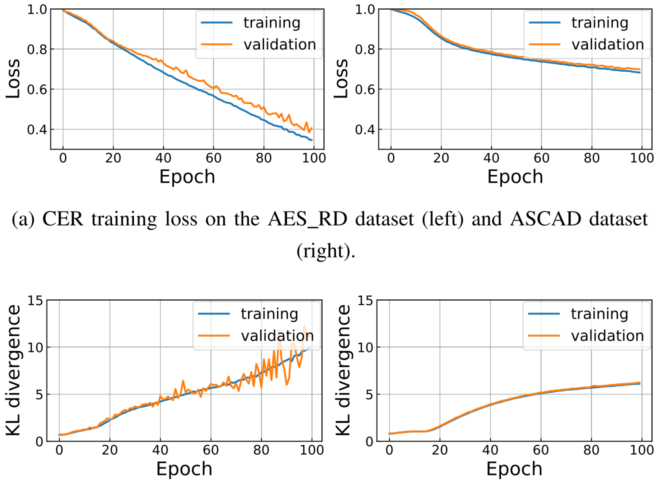


Fig. 6. Success rate of **SMOTE** for the AES_RD dataset (left) and ASCAD dataset (right).

method in [12]. The number of nearest neighbors used to augment the data points was set to five, and the number of minority labels was increased until the occurrence frequencies for all labels became the same. In this paper, we reproduced the settings of [12] as much as possible, although its parameters are not available due to the lack of description in the paper and the public source code. Note that we calculated the validation loss during training using the original samples, and therefore the validation loss and KL divergence show how well the trained model fits the test data. At the inference phase, we used the conventional HW-based likelihood to estimate the secret key. The training times of AES_RD and ASCAD with SMOTE are 661 and 680 seconds, respectively.

Fig. 5 shows the NLL loss and KL divergence of the trained model using datasets augmented using SMOTE. The results confirm that the KL divergences of these models generally become larger than those trained with the original datasets (Fig. 2(b)). This is because increasing the number of minority class data using SMOTE makes the probability distribution for label occurrence uniform, as described in Section V-B. In addition, the results show that the validation loss also increases because the probability distribution of training is different from that of validation because of the data augmentation.

Fig. 6 shows the SR of key estimation using the trained models. We confirm here that SMOTE improves the SR of the



(a) CER training loss on the AES_RD dataset (left) and ASCAD dataset (right).

(b) KL training divergence on the AES_RD dataset (left) and ASCAD dataset (right) with the CER loss.

Fig. 7. CER loss and KL divergence.

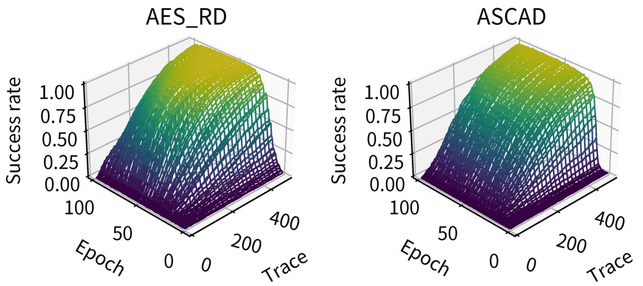


Fig. 8. Success rate of models trained using the **CER loss** for the AES_RD dataset (left) and ASCAD dataset (right).

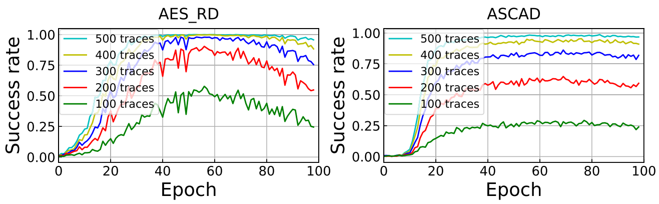


Fig. 9. Success rate of models trained using the **CER loss** for the AES_RD dataset (left) and ASCAD dataset (right) when the number of traces varies from 100 to 500.

HW-based NLL when compared with the results in Fig. 3(a). In contrast, the results are not better than those obtained by the proposed key-based likelihood, as shown in Fig. 3(b). This is because, whereas SMOTE is able to mitigate the negative effects of imbalanced data, the reduction in dataset quality caused by the artificial examples added by SMOTE negatively affects the SR results. Thus, the proposed method is more likely to obtain better results than data augmentation techniques because the key estimation using KNLL does not cause a deterioration in dataset quality.

D. Comparison of Loss Functions

We analyze the results of the key estimation using the model trained with the CER loss. We computed the CER loss based on the approximation presented in [14]. Here, we set the

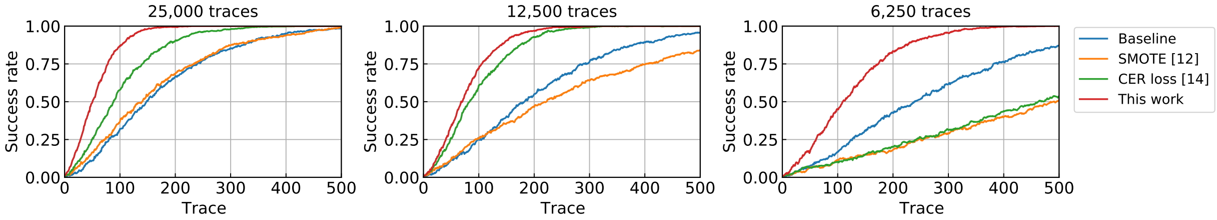


Fig. 10. Success rate of models trained on AES_RD dataset with trained traces varying from quarter to whole.

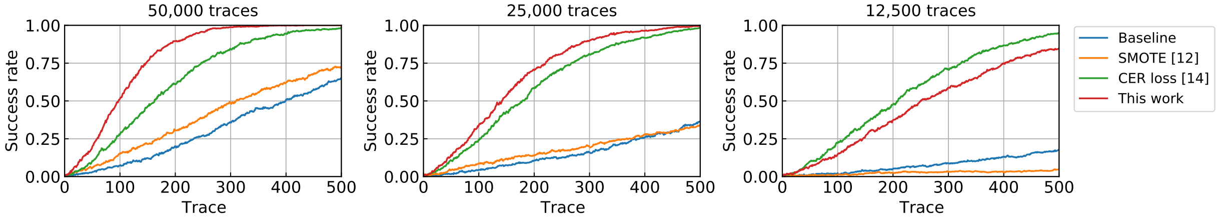


Fig. 11. Success rate of models trained on ASCAD dataset with trained traces varying from quarter to whole.

number of repetitions for averaging NLLs to 100 throughout our experiments because detail on how to set this number was not given in the literature. The training times of AES_RD and ASCAD datasets with CER loss were 723 and 1034 seconds, respectively.

Fig. 7 shows the CER loss and KL divergence of the model training on the datasets. We used the HW-based NLL for key estimation, similarly to [14]. The results confirm that the KL divergence of the models increases rapidly and the CER loss decreases rapidly as the training progresses. This is consistent with the analysis of the CER loss in Section IV. Fig. 8 shows the SR using the models trained with the CER loss. The result shows that the SR results are clearly better than their counterparts trained with the NLL loss in Fig. 3(a). This is because the increase of KL divergence between the model output and the binomial distribution mitigates the negative effect of imbalanced data.

In contrast, as shown in Figs. 7(a) and 8, although the validation loss decreases monotonically, the SR does not necessarily increase. Fig. 9 shows an example of a cross-section view of Fig. 8 for 100, 200, 300, 400, and 500 traces. In particular, the figure clearly shows that the SR decreases at around 100 epochs in the AES_RD dataset. This is inconsistent with the statement in [14] that the CER loss is a valid metric for DL-based SCAs. As mentioned in Section IV, one of the reasons for this is the assumption that the intermediate values computed from the correct and incorrect keys are independent [15].

Finally, the SR results are worse than those obtained using the proposed method, as shown in Fig.3(b). This indicates that the use of the proposed key-based likelihood is more effective than that of CER loss for imbalanced data problems under the conditions of this experiment. Note here that the result, in which a monotonically increasing KL divergence does not necessarily improve the SR, is not inconsistent with the claim in this study. This is because the KL divergence is only a metric of the impact of imbalanced data (i.e., how much the model output is biased toward the binomial distribution), and not a metric for estimating the SR of DL-based SCAs.

E. Effect of Decreasing the Number of Trained Traces

We finally analyze the effect of decreasing the number of trained traces on the SR. To investigate this effect, we performed key estimations using models trained on half and quarter the numbers of traces of AES_RD and ASCAD datasets. We used the same hyper-parameters as Section VI-B–VI-D in this experiment. Fig. 10 and 11 show the SRs using models training on original, half, and quarter of the training data. Because the number of iterations per epoch (i.e., the number of parameter updates) decreases when the training data is reduced, we increased the number of epochs according to the amount of reduced training data. For example, we doubled the number of epochs when the training data are half. Fig. 10 and 11 show the SRs of the best performing trained model for all epochs. "Baseline" in the figures is the case where conventional HW-based NLL is used for both training and inference.

The results confirm that the performance of all methods is degraded as the number of traces decreases. In particular, the data-augmentation method SMOTE does not achieve successful key recovery when the training data is small. Because SMOTE is a linear interpolation-based data-augmentation method, the variation of the augmented data is reduced when the training data is small, which would yield a rapid overfitting of the augmented traces. The performance of models using CER loss is generally better than that of the baseline models and models with SMOTE. However, the result for the 6,250 traces in the AES_RD dataset shows that the performance with CER loss can be significantly degraded when the number of traces is small. Finally, the proposed method achieves the best performance except for 12,500 traces in the ASCAD dataset. In addition, the reduction in performance with respect to the reduced number of traces is almost constant. In this sense, the proposed method is robust to the decrease in training data. These results allow us to conclude that the proposed method has the best overall performance in this experiment. Although CER loss has a comparable performance to the proposed method in several settings, the performance of CER loss is not convincing for some settings. Thus, the proposed method can efficiently eliminate the negative effects of imbalanced data while being robust to varying experimental settings in comparison with the conventional methods.

VII. Conclusion

In this study, we presented an analysis of the imbalanced data problem, which is one of the main issues hampering the application of DL to SCAs. In particular, we argued that the imbalanced data problem is caused by two factors: 1) the weak relationship between side-channel information and the intermediate value to be estimated, and 2) the occurrence probability of labels biased by a binomial distribution. To evaluate the negative effects quantitatively, we employed an evaluation metric based on the KL divergence between the model's output distribution and the binomial distribution. We then used the KL divergence to describe why data augmentation (like SMOTE) and the CER loss at the training phase can effectively mitigate the negative effect of imbalanced data.

In addition, we proposed a new solution to mitigate the imbalanced data problem at the inference phase and explained the relationship between the proposed solution and the conventional ones. The proposed key estimation method is based on the key-based likelihood function instead of the conventional HW/HD-based one, and its aim is to efficiently estimate the correct key even when the KL divergence is small (i.e., when application of the conventional method is difficult).

Subsequently, we demonstrated the validity and effectiveness of our analysis and solution through experiments using two datasets under various conditions. From our experimental results, we obtained the following conclusions:

1) The outputs of trained models are sometimes strongly biased toward the binomial distribution in DL-based SCAs.
2) The KL divergence works as a metric to evaluate the negative effect of binomial distributions.
3) The reason for SMOTE and CER loss mitigation in the imbalanced data problems can be explained by KL divergence, that is, they move the model output distribution far away from a binomial one.
4) The proposed method is more effective than the conventional methods such as SMOTE and CER loss in reducing the negative effects of imbalanced data.

Subsequent applications of the proposed method to various types of cryptographic modules remain a future work. For example, this study mainly focused on the software implementation of AES, but the applicability of the proposed method to other hardware architectures and cryptosystems (i.e., symmetric and public key cryptography) should be investigated. In particular, the use of our method to effectively solve the imbalanced data problem would be more important in cases with more complex and larger architectures as well as cryptosystems that may have intermediate values with longer bit lengths. In addition, the task of assessing the vulnerability of AES to DL-based SCAs using the proposed method should be examined in detail. For example, our method can reduce the difference in the success rate of attacks on the labels of intermediate values and attacks on the HW/HD, which could lead to the identification of leakage sources.

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO* (Lecture Notes in Computer Science), vol. 1109. Barbara, CA, USA: Springer, 1996, pp. 104–113.

[2] B. Hettwer, S. Gehrer, and T. Güneysu, "Applications of machine learning techniques in side-channel attacks: A survey," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 135–162, Jun. 2020, doi: 10.1007/s13389-019-00212-8.

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

[4] C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater, "Template attacks in principal subspaces," in *Proc. CHES*, in Lecture Notes in Computer Science, vol. 4249. Berlin, Germany: Springer, 2006, pp. 1–14. [Online]. Available: https://iacr.org/archive/ches2006/01/01.pdf

[5] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Proc. CHES*, in Lecture Notes in Computer Science, vol. 5154. Washington, DC, USA: Springer, 2008, pp. 411–425. [Online]. Available: https://www.iacr.org/archive/ches2008/51540408/51540408.pdf

[6] O. Choudary and M. G. Kuhn, "Efficient template attacks," in *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds. Cham, Switzerland: Springer, 2014, pp. 253–270.

[7] E. Cagli, C. Dumas, and E. Prouff, "Enhancing dimensionality reduction methods for side-channel attacks," in *Smart Card Research and Advanced Applications*, N. Homma and M. Medwed, Eds. Cham, Switzerland: Springer, 2016, pp. 15–33.

[8] Z. Martinasek, J. Hajny, and L. Malina, "Optimization of power analysis using neural network," in *Smart Card Research and Advanced Applications*, A. Francillon and P. Rohatgi, Eds. Cham, Switzerland: Springer, 2014, pp. 94–107.

[9] E. Cagli, C. Dumans, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Proc. CHES*, in Lecture Notes in Computer Science, vol. 10529, W. Fischer and N. Homma, Eds. Berlin, Germany: Springer, 2017, pp. 45–68.

[10] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, pp. 1–36, Nov. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8391

[11] B. Hettwer, T. Horn, S. Gehrer, and T. Guneysu, "Encoding power traces as images for efficient side-channel analysis," in *Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST)*, Dec. 2020, pp. 46–56.

[12] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni, "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

[13] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. Unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 3, pp. 148–179, May 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8292

[14] J. Zhang, M. Zheng, J. Nan, H. Hu, and N. Yu, "A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 73–96, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8583

[15] E. De Chérisey, S. Guilley, O. Rioul, and P. Piantanida, "Best information is most successful," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, no. 2, pp. 49–79, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7385

[16] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Germany: Springer, 2006.

[17] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning—Data Mining, Inference, and Prediction*, 2nd ed. New York, NY, USA: Springer-Verlag, 2009.

[18] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *J. Big Data*, vol. 6, no. 1, pp. 1–54, Dec. 2019.

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[20] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "CutMix: Regularization strategy to train strong classifiers with localizable features," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6023–6032.

[21] T. Shimada, S. Yamaguchi, K. Hayashi, and S. Kobayashi, "Data interpolating prediction: Alternative interpretation of mixup," in *Proc. 2nd Learn. From Ltd. Labeled Data Workshop*, 2019, pp. 1–8.

[22] J.-S. Coron and I. Kizhvatov, "An efficient method for random delay generation in embedded software," in *Proc. 11th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, in Lecture Notes in Computer Science, vol. 5747, C. Clavier and K. Gaj, Eds. Lausanne, Switzerland: Springer, 2009, pp. 156–170. [Online]. Available: https://www.iacr.org/archive/ches2009/57470156/57470156.pdf

[23] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptograph. Eng.*, vol. 10, no. 2, pp. 163–188, Jun. 2020, doi: 10.1007/s13389-019-00220-8.

[24] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel, "Revisiting a methodology for efficient CNN architectures in profiling attacks," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 3, pp. 147–168, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8586

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2015, pp. 1–15.

Akira Ito received the B.E. degree in information engineering and the M.S. degree in information sciences from Tohoku University, Japan, in 2017 and 2019, respectively. He is currently pursuing the Ph.D. degree with Tohoku University. His research interests include arithmetic circuits, formal verification, and hardware security.

Kotaro Saito received the B.E. degree in information engineering from Tohoku University, Japan, in 2020, where he is currently pursuing the master's degree. His research interest includes hardware security.

Rei Ueno (Member, IEEE) received the B.E. degree in information engineering and the M.S. and Ph.D. degrees in information sciences from Tohoku University, Japan, in 2013, 2015, and 2018, respectively. He is currently an Assistant Professor with the Research Institute of Electrical Communication, Tohoku University. He is also joining the JST as a Researcher for a PRESTO project. His research interests include arithmetic circuits, cryptographic implementations, formal verification, and hardware security. He received the Kenneth C. Smith Early Career Award in microelectronics from ISMVL 2017.

Naofumi Homma (Senior Member, IEEE) received the B.E. degree in information engineering and the M.S. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 1997, 1999, and 2001, respectively. From 2001 to 2009, he was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, where he became an Associate Professor in 2009. Since 2016, he has been a Professor with the Research Institute of Electrical Communication, Tohoku University. From 2009 to 2010 and from 2016 to 2017, he was a Visiting Professor with Telecom ParisTech, Paris, France. His research interests include computer arithmetic, electronic design automation methodology, and hardware security. He received an IP Award from the LSI IP Design Award in 2005, the Best Paper Award from the Workshop on Synthesis and System Integration of Mixed Information Technologies in 2007, the RIEC Award in 2012, the Best Symposium Paper Award from the 2013 IEEE International Symposium on Electromagnetic Compatibility, the Best Paper Award from the 2014 IACR Conference on Cryptographic Hardware and Embedded Systems, the Japan Society for the Promotion of Society Prize in 2018, and the German Innovation Award in 2018.