Received 17 May 2023, accepted 29 July 2023, date of publication 2 August 2023, date of current version 10 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3301178

METHODS

Non-Profiled Deep Learning-Based Side-Channel Analysis With Only One Network Training

KENTARO IMAFUKU, SHINICHI KAWAMURA, (Fellow, IEEE), HANAE NOZAKI, JUNICHI SAKAMOTO, AND SAKI OSUKA
National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan
Corresponding author: Kentaro Imafuku (imafuku-kentaro@aist.go.jp)

ABSTRACT We propose efficient protocols for non-profiled deep learning-based side-channel analysis (DL-SCA). While the existing protocol, proposed by Timon in 2019, requires computational resources for training as many neural networks as the number of key candidates, our protocol requires training only one network, which can be transformed into a network associated with each key candidate. For instance, in the case of analysis for the AES, the network training complexity is 1/256 of that for the existing protocol. In this study, we describe our idea and formulate it as two protocols depending on the metrics used. We numerically examine them by implementing each protocol with two network architectures, multilayer perceptron and convolutional neural network. Using publicly available open data (ASCAD), we show that both protocols efficiently work as expected. We also clarify that our trained network, as in Timon's original case, can be recycled for an attack against the same device with different key materials. Non-profiled DL-SCAs are superior to profiled ones in that they require no reference device for profiling before analyzing the target device. This property holds for our proposal as well.

INDEX TERMS Deep learning-based side-channel analysis, non profiled analysis.

I. INTRODUCTION

The emergence of new information processing technologies sometimes poses new threats to the IT community. Although new technologies enable a variety of applications that enrich our lives, they may also offer malicious attackers new tools that were previously unavailable. Thus, to safely enjoy new technologies, we must carefully evaluate their safety.

Machine learning is a prime example of such technology today. Since computational resources, such as GPUs, have become relatively inexpensive, deep learning (DL) has attracted considerable attention among several machine learning technologies. Its applications in many fields, such as image recognition and automated driving, demonstrate the potential of DL. At the same time, malicious uses of DL have already been recorded. Thus, we need to further investigate the benefits and threats of DL.

In the field of hardware security, the use of DL in side-channel attacks (SCA) was recognized relatively early. SCA is an attack to reveal secret parameters in cryptographic systems, typically secret keys, by analyzing side-channel information. Execution time, power consumption, and electromagnetic waves are typical examples of side-channel information. They are obtained as time series data by measurements during the execution of the cryptographic algorithm. Successful use of DL with physically observed data naturally entails its use for SCA. Such attacks are called DL-based SCAs (DL-SCAs).

SCA is roughly categorized into two types depending on the data available during the attack. Referring to the target device as the device under test (DUT), non-profiled SCA [1], [2], [3], [4], [5] has access only to the data from DUT. At the same time, profiled SCA [6], [7], [8], [9] is an attack when additional data from reference devices are available. Both SCA types are practically used, depending on the attacking scenarios.

Following the above two categories, DL-SCAs may be categorised into non-profiled DL-SCA and profiled DL-SCA types. The profiled DL-SCA, in which data from the reference devices are available as training data, can be regarded as

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

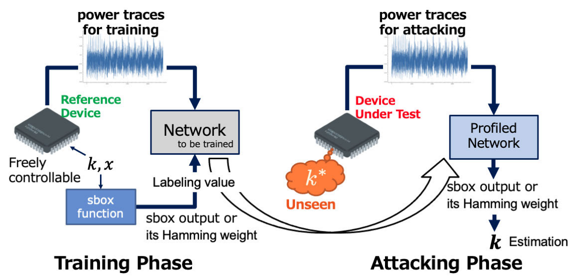


FIGURE 1. Sketch of profiled DL-SCA.

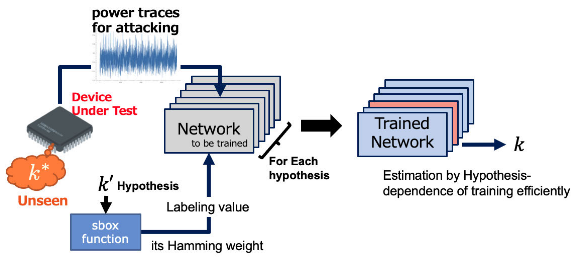


FIGURE 2. Sketch of non-profiled DL-SCA proposed in [19].

a straightforward application of supervised learning. Because of this affinity, profiled DL-SCAs have been studied more extensively [10], [11], [12], [13], [14], [15], [16] than non-profiled DL-SCAs [17], [18], [19]. Similar to ordinary SCAs, both types of DL-SCAs have practical significance.

As shown in Fig.1, profiled DL-SCA consists of two phases. The first phase is the training phase, in which data from the reference devices are used to train a network. The network is taught the relationship between traces as side-channel information and labelling values. Notably, the output of the sbox function or its Hamming weight are typically examined as the labelling values. The network has parameters that control the output for a given input. The training is a process of tuning the parameters so that the network statistically reproduces the input–output relation. The second phase is the attacking phase: the profiled network is fed traces from the DUT to estimate the corresponding labelling values. Using the estimated values with the corresponding plaintexts, the attacker can estimate the secret key used in the DUT.

The basic protocol for non-profiled DL-SCAs has been proposed by Timon in [19], which is, to our best knowledge, the only such protocol. The basic idea of the protocol is that training the network on correctly labelled data yields a better-trained network than training on wrongly labelled data. In the original paper, the Hamming weight and the least significant bit of the output of the sbox function were used as the labelling values. The protocol may be outlined as follows (see Fig.2 and Sec.II-B for details):

1) A key is hypothetically assumed. The labelling value is computed with the plaintext and the assumed key.
2) By pairing the labelling value with the corresponding trace, a data set consisting of the pairs is prepared.
3) Using the data set, a network is trained. When the training is accomplished, the obtained value of the loss function, or most typically cross-entropy, is recorded as a metric of the degree of network training.
4) The key hypothesis is varied and the obtained values of the loss function are compared; the key yielding the best training is taken as an estimation of the true key.

The protocol works based on the difference in "learnability" of the correctly and wrongly labelled data. Learnability is one of the fundamental concepts clarified in machine learning. In this sense, the protocol brilliantly exploits an essential aspect of machine learning. This was a significant step in understanding non-profiled DL-SCAs.

Compared to profiled DL-SCAs, the non-profiled DL-SCA proposed by Timon requires more computational resources. In both DL-SCA types, network training, an iterative search over parameter space, is used. In general, such a parameter search requires considerable computational resources. In Timon's protocol, such training must be repeated for each key candidate, resulting in substantial computational complexity. Therefore, the non-profiled DL-SCA appears less practical than the profiled DL-SCA, which requires only one network training for the training data.

To evaluate the true significance or impact of non-profiled DL-SCA, more efficient protocols must be thoroughly explored. The present study proposes efficient protocols for non-profiled DL-SCAs. In our approach, a network is trained in such a way that the network can be transformed a posteriori into all networks that are to be obtained with key-dependent training. The universality of such a network is comparable to that of a stem cell which can be transformed into a variety of cells. This stem cell-like network plays a central role in our approach.

Our Contributions:

Our contributions in this study are threefold:

1) We propose protocols to realise non-profiled DL-SCA with only one training process. Consequently, the required training complexity for AES [20] is approximately 1/256 of that in Timon's protocol [19]. Our approach is general and can be applied regardless of the network details or the cryptographic algorithm used in DUT. Note that, in general, the more complex and functional the network, the greater the complexity of the training. The more such networks are used, the greater the superiority of our ideas to Timon's originals.
2) We numerically show that the proposed protocols work using a publicly available dataset ASCAD [21].
3) We clarify that our trained network, as is Timon's original case, can be recycled for an attack against the same device with different key materials.

The remainder of this paper is organised as follows: Section II outlines the problem setting for our discussion and Timon's protocol. In Sec.III, we describe our idea and formalise it as two protocols: S-protocol and T-protocol. Then, in Sec.IV, we numerically examine our protocols and study their characteristics, followed by discussion in Sec.V.

Finally, we summarise the main findings of this study in Sec. VI.

II. PROBLEM SETTING AND TIMON's IDEA

This Section introduces the setting and notations, followed by a review of Timon's original work in [19].

A. SETTING

The following is a summary of the settings used in the present study:

- We consider a function that maps the bitwise XOR of an input ($x \in \mathcal{X}$) and a parameter ($k \in \mathcal{K}$) to an output ($y \in \mathcal{Y}$), similar to the sbox function in AES. Subsequently, this relation is denoted as $y = \mathrm{sbox}(x \oplus k)$, generally called the sbox function, without further specifying the function's form. The input and the parameter are called *plaintext* and *secret key*, respectively.
- *A DUT* is a device that executes an algorithm consisting of the sbox function with a particular secret key denoted by k^*. This study uses the secret key used in DUT as the *true key*.
- A set of plaintexts is introduced as $\{x_1, \cdots, x_N\}$ with index $j \in \{1 \cdots, N\}$. $w_j \in \mathcal{W}$ denotes the trace obtained by measuring power consumption, electromagnetic leakage, etc. when j-th plaintext x_j is input.
- *Non-profiled DL-SCAs* are herein considered the attacks to estimate the true key solely from a given data set
$$D_{\mathrm{DUT}} := \left\{ \langle x_j, w_j \rangle \mid j \in \{1, \cdots, N\} \right\} \tag{1}$$
using DL techniques.
- A labelling function is introduced. The domain and range of the function are $\mathcal{X} \times \mathcal{K}$ and a set of discrete values, say $\Lambda := \{\ell_1, \cdots, \ell_d\}$, respectively. A specific choice of Λ is considered later.
- As a network to be trained, we consider an input–output system with a tuning parameter θ determined through the so-called network training process, where the system's domain is $\mathcal{W}$. The output is the probability distribution of the discrete set Λ obtained assuming that the last layer of the system is implemented by the so-called softmax function [22]. Let $\mu_\theta^{(w)}(\ell_\alpha)$ be the probability distribution obtained by inputting the trace $w \in \mathcal{W}$ into the network. It should be noted that, because of the properties of the softmax function,
$$\mu_\theta^{(w)}(\ell_\alpha) \geqslant 0, \quad \text{and} \quad \sum_{\alpha=1}^{d} \mu_\theta^{(w_j)}(\ell_\alpha) = 1 \tag{2}$$
always hold.

B. TIMON's PROTOCOL

Based on the above setting, below we formulate Timon's protocol for our discussion:

1) For each key candidate $\kappa \in \mathcal{K}$ and plaintext $x_j \in \mathcal{X}$, labelling value is defined as
$$\ell_j(\kappa) := \lambda(x_j, \kappa) \tag{3}$$
where $\lambda(x, k)$ is the labelling function introduced in the previous section and specified below.
2) Using the data set of (1), we prepare a κ-depending data set consisting of pairs of the labelling value and traces associated with the corresponding plaintext as
$$D(\kappa) := \left\{ \langle \ell_j(\kappa), w_j \rangle \mid j \in \{1, \cdots, N\} \right\}. \tag{4}$$
Note that the number of such data sets is $|\mathcal{K}|$, the number of κ' variations.
3) We execute the network training using the data set in (4). The training is an iterative search for θ to obtain the network output $\mu_\theta^{(w_j)}(\ell_\alpha)$ minimising cross-entropy [22], [23]
$$\mathcal{S}_\theta(\kappa) := -\sum_{j=1}^{N} \log \mu_\theta^{(w_j)}(\ell_j(\kappa)). \tag{5}$$
The search can be performed using a standard method of training for the general DL framework. We do not specify the training algorithm unless it is relevant to our discussion. Let θ_κ^* be the value of the parameter determined by the training.
4) By comparing $\mathcal{S}_{\theta_\kappa^*}(\kappa)$ over $\kappa \in \mathcal{K}$, we estimates the true key. More specifically, we can consider
$$\arg\min_{\kappa \in \mathcal{K}} \mathcal{S}_{\theta_\kappa^*}(\kappa) \tag{6}$$
as the most likely candidate for the true key. We can also determine the estimation ranking of each candidate key by arranging $\mathcal{S}_{\theta_\kappa^*}(\kappa)$ in the order of decreasing size, if necessary.

Timon also discusses the estimation using sensitivity analysis. However, we omit it because it is less relevant to the purpose of the present study. In Timon's protocol, the total training complexity is proportional to $|\mathcal{K}|$. Therefore, Timon's protocol requires considerable computational resources, particularly in cases where the size of the trained network is reasonably large.

Considering the choice of the labelling function,
$$\lambda(x, k) = \mathrm{Hw}(\mathrm{sbox}(x \oplus k)) \tag{7}$$
and
$$\lambda(x, k) = \mathrm{LSB}(\mathrm{sbox}(x \oplus k)) \tag{8}$$
are suggested in [19]. As carefully pointed out in section III-C in the paper, the labelling function must not be bijective for the protocol. If the labelling function is bijective, consisting of a group of traces with the same label value $\ell_j(\kappa)$ from $D(\kappa)$ in (4) as $Q_\ell := \left\{ w_j \mid \ell_j(\kappa) = \ell \right\}$, variations in κ are reflected only in the labelling value attached to each group but not in the membership of Q_ℓ because the value of $\mathcal{S}_{\theta_\kappa^*}(\kappa)$ does not depend on κ. In the case of AES, the sbox function is bijective; thus, it cannot be employed as the labelling function. For this reason, an alternative labelling function that depends on a power consumption model like the above examples needs to be chosen. Timon's protocol is

summarised as Algorithm 1 below. Let us underline that the network training part in line 3 is in the loop on the key variable of lines 1 to 9.

Algorithm 1 Timon's Protocol

Inputs: $\{\langle x_j, w_j\rangle \,|\, j \in \{1, \cdots, N\}\}$, Net_θ, n_{epochs}, sbox
/ D_{DUT} in (1), a network parametrized by θ, the number of epochs for training, and sbox function*/*

Output: $\hat{\kappa}$ */* The estimation for the true key.*/*

```
1:  for $\kappa \in \mathcal{K}$ do
2:      $\ell_j(\kappa) = \mathrm{sbox}(x_j \oplus \kappa) \rightarrow D(\kappa) = \{\langle \ell_j(\kappa), w_j\rangle \,|\, j \in \{1, \cdots, N\}\}$
3:      $\mathbf{DL}(\mathrm{Net}_\theta, D(\kappa), n_{epochs}) \rightarrow \mathrm{Net}_{\theta^*}$ /* Network Training with Deep Learning.*/
4:      for $j \in \{1, \cdots, N\}$ do
5:          $\langle \mathrm{Net}_{\theta^*_\kappa}, w_j\rangle \rightarrow \mu^{(w_j)}_{\theta^*_\kappa}(\ell)$
6:          $\langle \mathrm{sbox}, x_j, \kappa, \mu^{(w_j)}_{\theta^*_\kappa}(\ell)\rangle \rightarrow \mu^{(w_j)}_{\theta^*_\kappa}(\mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa)))$
7:          $\mathcal{S}_{\theta^*_\kappa}(\kappa) = \mathcal{S}_{\theta^*_\kappa}(\kappa) - \log \mu^{(w_j)}_{\theta^*_\kappa}(\mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa)))$
8:      end for
9:  end for
10: $\hat{\kappa} = \arg\min_{\kappa \in \mathcal{K}} \mathcal{S}_{\theta^*_\kappa}(\kappa)$
11: return $\hat{\kappa}$
```

III. OUR PROPOSAL

In the previous section, we showed that the training complexity is proportional to $|\mathcal{K}|$ in Timon's protocol. However, the training complexity can be reduced to 1 based on a simple idea explained in this section. As shown below, our proposal is based on the additivity rule, a fundamental law in probability theory. The additivity rule states that when events A and B are mutually exclusive, the probability of A or B is the sum of the probabilities of A and B, described as

$$p(A \cup B) = p(A) + p(B), \quad \text{for} \quad A \cap B = \emptyset. \tag{9}$$

Equation (9) holds in general and does not depend on the probability distributions of A and B.

A. PRINCIPLE OF OUR APPROACH

Here we explain how the additivity rule plays a role in our approach. Let us hypothesise three probability distributions, (10)–(12), to be obtained as outputs of three networks using the following three labelling functions:

- $\lambda(x, k) = \mathrm{Hw}(\mathrm{sbox}(x \oplus k))$: We perform the network training described in the previous section by temporarily assuming the value of the key to κ. Then we obtained a network parameterised by θ^*_κ. Let
$$\mu^{(w)}_{\theta^*_\kappa}(\ell) \tag{10}$$
be the output probability distribution from the trained network when w is input. It should be noted that the range of ℓ must coincide with the range of the labelling function, that is $\ell \in \{0, \cdots, 8\}$ in this case.
- $\lambda(x, k) = \mathrm{sbox}(x \oplus k)$: We perform the network training by assuming the value of the key as κ. We obtain a network parameterised by σ^*_κ. Let
$$\nu^{(w)}_{\sigma^*_\kappa}(r) \tag{11}$$
be the output probability distribution from the trained network when w is input. It should be noted that the range of r is the range of the sbox function itself.
- $\lambda(x, k) = x$: We perform the network training. Let
$$\rho^{(w)}_{\eta^*}(\xi) \tag{12}$$
be the output probability distribution from the trained network when w is input, where η^* is the value of the network parameter obtained through the training. Notice that the range of ξ is the range of x, *i.e.*, $\mathcal{X}$. The training process to obtain η^* is independent of κ and so is η^*.

Now let us hypothetically consider a joint probability distribution

$$p_\kappa(w, \ell, r, \xi) \tag{13}$$

describing "the true probability distribution" where w, ℓ, r, and ξ are the random variables mentioned above. If the above three trainings are properly performed with sufficient data, the obtained probability distributions are good approximations of the conditional probability distributions derived from the joint probability as

$$\mu^{(w)}_{\theta^*_\kappa}(\ell) \simeq p_\kappa(\ell|w), \ \nu^{(w)}_{\sigma^*_\kappa}(r) \simeq p_\kappa(r|w), \ \text{and} \ \rho^{(w)}_{\eta^*}(\xi) \simeq p(\xi|w). \tag{14}$$

It should be noted that these approximations are a consequence of the choice of the cross-entropy as a loss function for the network training, just as in (5). We can easily prove that the loss function is minimised only when the output distribution agrees with the corresponding conditional probability [22]. Because additivity naturally holds between the conditional probability distributions on each right-hand side, we can safely conclude that additivity approximately holds for the three distributions on the left-hand side. Thus we obtain

$$\mu^{(w)}_{\theta^*_\kappa}(\ell) \simeq \sum_{r \in \Omega_\ell} \nu^{(w)}_{\sigma^*_\kappa}(r), \quad \Omega_\ell := \{r \mid \mathrm{Hw}(r) = \ell\} \tag{15}$$

and

$$\nu^{(w)}_{\sigma^*_\kappa}(r) \simeq \sum_{\xi \in \Xi_{r,\kappa}} \rho^{(w)}_{\eta^*}(\xi), \ \Xi_{r,\kappa} := \{\xi \mid \mathrm{sbox}(\xi \oplus \kappa) = r\}. \tag{16}$$

Notably, Ω_ℓ and $\Xi_{r,\kappa}$ can be prepared in advance independently from the training process and their preparation may be performed utilizing significantly less computational resources than the network training. By combining (15)-(16), we obtain

$$\mu^{(w)}_{\theta^*_\kappa}(\ell) \simeq \sum_{r \in \Omega_\ell} \sum_{\xi \in \Xi_{r,\kappa}} \rho^{(w)}_{\eta^*}(\xi) \tag{17}$$

as our central finding. In Timon's protocol, $\mu^{(w)}_{\theta^*_\kappa}(\ell)$ on the left-hand side was supposed to be obtained via multiple κ-dependent network training processes. Thus, obtaining $\mu^{(w)}_{\theta^*_\kappa}(\ell)$ for all $\kappa \in \mathcal{K}$ would require $|\mathcal{K}|$ network training processes. At the same time, according to (17), once $\rho^{(w)}_{\eta^*}(\xi)$ is obtained, $\mu^{(w)}_{\theta^*_\kappa}(\ell)$ can be constructed using Ω_ℓ and $\Xi_{r,\kappa}$. In short, the presented finding reduces the number of training processes from $|\mathcal{K}|$ times to 1.

Let us further stress that the distribution $\rho^{(w)}_{\eta^*}(\xi)$ is the output from the network trained by data labelled by plaintext. Equation (17) shows that the network can be transformed a posteriori into all networks that are to be obtained with key-dependent training; thus, such a network is referred to as the stem cell-like network in Sec.I.

B. IMPLICATIONS OF $\rho^{(w)}_{\eta^*}(\xi)$

Leaving aside the mathematical structure of our approach, let us consider the implications of the proposed network training for the estimation of the known plaintext, which may seem slightly odd at first glance.

The application of Bayes' theorem to $p(\xi|w_j)$ in (14) yields

$$\rho^{(w_j)}_{\eta^*}(\xi) \simeq p(\xi|w_j) = \frac{p(\xi)}{p(w_j)}p(w_j|\xi) \tag{18}$$

As ξ is a plaintext variable, let us assume $p(\xi) = |\mathcal{X}|^{-1}$ for clarity. In addition, let us consider a simple case where ξ-dependence in w_j is primarily given through $\mathrm{Hw}(\mathrm{sbox}(\xi \oplus k^*))$ with true key k^*. In this case, $p(w_j|\xi) \simeq p(w_j \mid \mathrm{Hw}(\mathrm{sbox}(\xi \oplus k^*)))$, and the above expression can be rewritten as

$$\rho^{(w_j)}_{\eta^*}(\xi) \simeq \frac{|\mathcal{X}|^{-1}}{p(w_j)}\, p(w_j \mid \mathrm{Hw}(\mathrm{sbox}(\xi \oplus k^*))). \tag{19}$$

Equation (19) indicates that the probability is approximately equal for all ξ yielding the same value of $\mathrm{Hw}(\mathrm{sbox}(\xi \oplus k^*))$ and that the distribution depends on the true key. The essence of the proposed training is to obtain the true key dependence in $\rho^{(w)}_{\eta^*}(\xi)$ rather than correctly estimate the known plaintext.

C. PROTOCOLS

In this section, we formalise the above idea as two concrete protocols. We call them S-protocol and T-protocol, depending on the metrics used.

1) S-PROTOCOL

The first protocol is a direct extension of Timon's protocol and is based on the same metric, namely

$$\mathcal{S}_{\theta^*_\kappa}(\kappa) := -\sum_{j=1}^{N} \log \mu^{(w_j)}_{\theta^*_\kappa}(\ell_j(\kappa)). \tag{20}$$

However, when computing $\mu^{(w_j)}_{\theta^*_\kappa}(\ell_j(\kappa))$ on the right-hand side, our finding in (17) is used instead of repeating the network training for each $k \in \mathcal{K}$. The protocol is summarised in Algorithm 2, and Fig.3. As described above, the network

Algorithm 2 S-Protocol

```
Inputs: $D_{\mathrm{DUT}} = \{\langle x_j, w_j\rangle \mid j \in \{1, \cdots, N\}\}$, $\mathrm{Net}_\eta$, $n_{epochs}$, sbox
    /* $D_{\mathrm{DUT}}$ in (1), a network parametrized by $\eta$, the number of epochs for training, and sbox function */
Output: $\hat{\kappa}$ /* The estimation for the true key. */
 1: $\mathrm{DL}(\mathrm{Net}_\eta, D_{\mathrm{DUT}}, n_{epochs}) \to \mathrm{Net}_{\eta^*}$  /* Network training with Deep Learning. */
 2: for $\kappa \in \mathcal{K}$ do
 3:     $\mathcal{S}_{\theta^*_\kappa}(\kappa) = 0$
 4:     for $j \in \{1, \cdots, N\}$ do
 5:         $\langle \mathrm{Net}_{\eta^*}, w_j\rangle \to \rho^{(w_j)}_{\eta^*}(\xi)$
 6:         $\langle \mathrm{sbox}, x_j, \kappa, \rho^{(w_j)}_{\eta^*}(\xi)\rangle \to \mu^{(w_j)}_{\theta^*_\kappa}(\mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa)))$
            /* According to (17). */
 7:         $\mathcal{S}_{\theta^*_\kappa}(\kappa) = \mathcal{S}_{\theta^*_\kappa}(\kappa) - \log \mu^{(w_j)}_{\theta^*_\kappa}(\mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa)))$
 8:     end for
 9: end for
10: $\hat{\kappa} = \arg\min_{\kappa \in \mathcal{K}} \mathcal{S}_{\theta^*_\kappa}(\kappa)$
11: return $\hat{\kappa}$
```

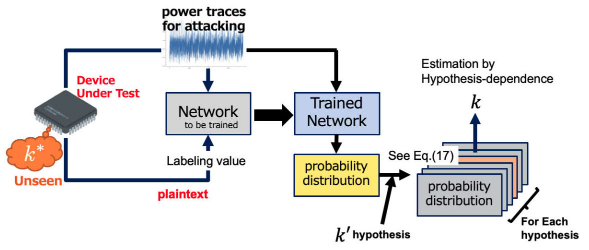


FIGURE 3. Sketch of the S-protocol.

training is done with data labelled by plaintext. Let us stress that the network training part in line 1 is separated from the loop on the key variable of lines 2-9 and is executed only once, unlike in Timon's original protocol shown in Algorithm 1.

2) T-PROTOCOL

The second protocol is based on a new metric that is uniquely derived from the essence of our approach. Referring to the distribution in (19), we introduce the following distribution:

$$\tau^{(w_j)}_\kappa(\xi) := \begin{cases} Z_j^{-1} \text{for } \xi \text{ s.t } \begin{pmatrix} \mathrm{Hw}(\mathrm{sbox}(\xi \oplus \kappa)) \\ = \mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa)) \end{pmatrix} \\ 0 \quad \text{otherwise} \end{cases} \tag{21}$$

where Z_i is the number of ξ that satisfies $\mathrm{Hw}(\mathrm{sbox}(\xi \oplus \kappa)) = \mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa))$. The distribution is obtained from (19) by replacing k^* by κ and by supposing

$$p(w_j|\mathrm{Hw}(\mathrm{sbox}(\xi \oplus \kappa)) = 0 \tag{22}$$

for ξ s.t. $\mathrm{Hw}(\mathrm{sbox}(\xi \oplus \kappa)) \neq \mathrm{Hw}(\mathrm{sbox}(x_j \oplus \kappa))$.

Using of (21), $\tau^{(w_j)}_{\kappa=k^*}(\xi)$ is expected to be one of the most similar distributions to $\rho^{(w_j)}_{\eta^*}(\xi)$ among $\{\tau^{(w_j)}_\kappa(\xi)\}_{\kappa\in\mathcal{K}}$.

Algorithm 3 T-Protocol

```
Inputs: $D_{\mathrm{DUT}} = \{\langle x_j, w_j \rangle \mid j \in \{1, \cdots, N\}\}$, $\mathrm{Net}_\eta$, $n_{epochs}$, sbox
    /* $D_{DUT}$ in (1), a network parametrized by $\eta$, the number of epochs for training, and sbox function*/
Output: $\hat{\kappa}$ /* The estimation for the true key.*/
 1: $\mathbf{DL}(\mathrm{Net}_\eta, D_{\mathrm{DUT}}, n_{epochs}) \rightarrow \mathrm{Net}_{\eta^*}$   /* Network training with Deep Learning.*/

 2: for $\kappa \in \mathcal{K}$ do
 3:     $\mathcal{T}(\kappa) = 0$
 4:     for $j \in \{1, \cdots, N\}$ do
 5:         $\langle \mathrm{Net}_{\eta^*}, w_j \rangle \rightarrow \rho^{(w_j)}_{\eta^*}(\xi)$
 6:         $\langle \mathrm{sbox}, x_j, \kappa \rangle \rightarrow \tau^{(w_j)}_{\kappa}(\xi)$ /* According to (21).*/
 7:         $\mathcal{T}(\kappa) = \mathcal{T}(\kappa) + \sum_\xi \tau^{(w_j)}_\kappa(\xi) \log \left[\tau^{(w_j)}_\kappa(\xi) / \rho^{(w_j)}_{\eta^*}(\xi)\right]$
 8:     end for
 9: end for
10: $\hat{\kappa} = \arg\min_{\kappa \in \mathcal{K}} \mathcal{T}(\kappa)$
11: return $\hat{\kappa}$
```

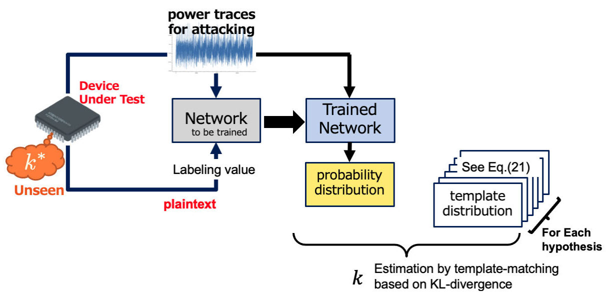


FIGURE 4. Sketch of the T-protocol.

It should be noted that the similarity for all j is maintained only for $\kappa = k^*$ and no other κ. Additionally, (21) can be prepared independently from the network training. Therefore, by preparing $\{\tau^{(w_j)}_\kappa(\xi)\}_{\kappa \in \mathcal{K}}$ in advance as a sort of template, and by introducing a metric to measure the similarity between $\rho^{(w_j)}_{\eta^*}(\xi)$ and $\tau^{(w_j)}_\kappa(\xi)$ as

$$\mathcal{T}(\kappa) := \sum_{j=1}^{N} \sum_{\xi} \tau^{(w_j)}_\kappa(\xi) \log \frac{\tau^{(w_j)}_\kappa(\xi)}{\rho^{(w_j)}_{\eta^*}(\xi)}, \tag{23}$$

we can estimate the true key through the κ-dependence of the metric. We assumed that the last layer of the network is a softmax function. Therefore, $\rho^{(w_j)}_{\eta^*}(\xi)$ can be guaranteed to be positive for all ξ. Equation (23) is the Kullback-Leibler divergence, which is often used as the standard information-theoretic metric of the similarity between two probability distributions [24]. The protocol is summarised in Algorithm 3 and Fig.4. Similar to the S-protocol, the network training is performed with data labelled by plaintext. The network training part in line 1 is separated from the loop on the key variable in lines 2-9 and is executed only once.

TABLE 1. Four data for examination.

		sbox0 (masked with 0)	sbox2 (randomly masked)
sample points		32400 ∼ 33099	46900 ∼ 47599
desync-param	0	(A)	(C)
	50	(B)	(D)

IV. NUMERICAL EXPERIMENTS

In this section, we numerically examine our protocols in practice. For this purpose, similarly to [19], we use a publicly available dataset introduced in [21]. The dataset named ANSSI SCA Database (ASCAD) aims to provide the research community with a benchmarking reference. Below, we first summarise the main characteristics of the data analysed in the present study and then present the results of applying our approach.

A. TARGET DATA

In the present study, "`ATM_AES_v1_fixed_key.h5`" included in the ASCAD v1 dataset was used. The data in the file is a set of traces by measurements of the electromagnetic waves of an assembly-developed AES cryptographic operation on the ATMega8515 8-bit AVR microcontroller. The measurements were taken for the first round of AES processing with a fixed key and various plaintext inputs. As illustrated in Fig. 4 in [21], the corresponding plaintexts and keys are stored in the file in addition to the traces. The 16 sbox in AES from 0 to 15 are with the so-called masked implementation as countermeasures against SCA. The first two sbox (sbox0 and sbox1) are implemented with a constant masking value of 0, essentially an unmasked implementation. The rest are randomly masked to be resilient to first-order SCA. Notably, the values of the mask and output of each sbox function are also stored in the file, though they are not used in our analysis. A code to mimic the effect of jitter, which can be implemented as an SCA countermeasure, is also available. That is done by additionally processing the data to introduce random desynchronization to traces. The maximum magnitude of desynchronization is tuned by a parameter called desync-param. If the parameter is set to 0, the data corresponds to the case where the countermeasures were not taken. As each sbox function is executed serially in time, DL-SCA can be performed on each sbox by roughly cutting out the corresponding portion from the trace consisting of 100,000 sampling points. The portion size is decided by the input size of the network to be used, which is 700 in all our experiments. The rough location of the portion corresponding to the target sbox can be estimated through an a priori check of the location of the point of interest using SNR, as shown in Figs. 2 and 3 in [21]. For our numerical experiments, combining these conditions, we prepared the four traces; (A), (B),(C), and (D), as listed in Table 1.

B. NETWORK MODELS

The following two network architectures were adopted as typical examples in the experiments. These initially appeared in [21], where several architectures were numerically examined and compared in terms of efficiency. One is the multilayer perceptron (MLP), and the other is the convolutional neural network (CNN). Following [21], we refer to these networks as $\mathtt{MLP_{best}}$ and $\mathtt{CNN_{best}}$. The main components of their architecture are as follows:

- $\mathtt{MLP_{best}}$: This network consists of six layers of perceptrons and a softmax function layer. Layer 1 consists of 700 nodes corresponding to the length of the input trace. Layers 2 through 5 consist of 200 nodes each, and layer 6 consists of 256 nodes. The sixth layer is further connected to a softmax function with 256 outputs in [0, 255]. This final layer ensures that we obtain a positive normalised numerical output that is interpretable as an estimated probability. Each node in a layer within the six layers is fully connected to all nodes in the following adjacent layer as reflecting input–output relations. Each node represents the so-called activation function, which is chosen as ReLU in this network model. In all our experiments with $\mathtt{MLP_{best}}$, the network is trained with RMSProp optimiser at a learning rate of 10^{-5}, and batch size of 200.
- $\mathtt{CNN_{best}}$: This network consists of three parts, *i.e., convolution part*, *fully connected layers*, and *softmax function*. The convolution part has five blocks, each consisting of (64,128,256,512,512) filters, followed by a pooling layer for each block. All the filters have the same kernel size of 11. The padding scheme is the so-called *same padding*. The convolution part is connected to the two dense layers of 4,096 units before being connected to a softmax function with 256 outputs in [0, 255], similar to the above case. The network is also trained with RMSProp optimiser at a learning rate of 10^{-5}, and batch size of 200.

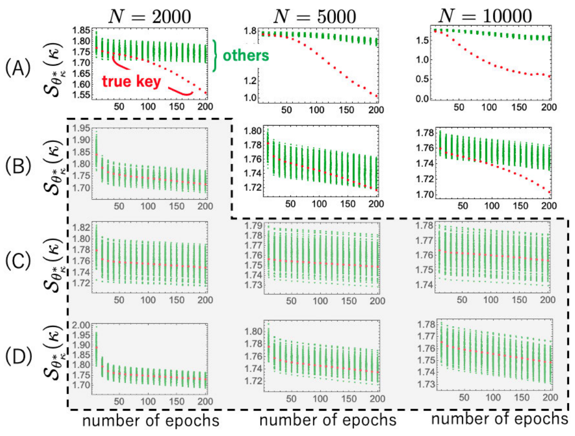
See [21] for more details. An example of python code executing the training of these two networks is attached to the ASCAD dataset, and we use of these codes for our numerical experiments.

C. RESULTS

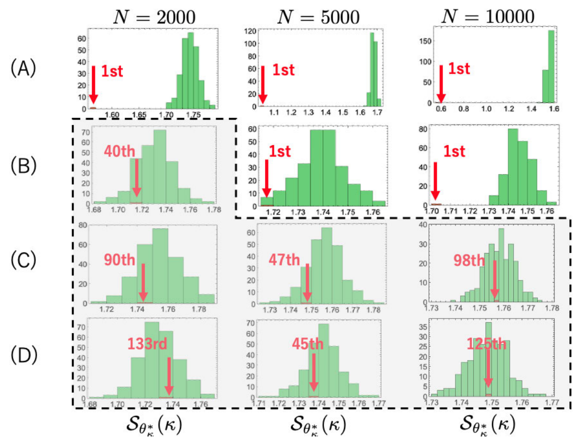
This section presents the results of applying our two protocols proposed in Sec.III-C to the four data types listed in Table 1 using the two networks introduced in Sec.IV-B.

1) S-PROTOCOL

Figure 5 shows the results of applying the S-protocol with $\mathtt{MLP_{best}}$ to the four data sets. N is the number of traces used in the analysis, and the experiment was conducted for three different N: 2000, 5000, and 10000. Row names of (A), (B), (C), and (D) correspond to the different data types presented in Table 1. In Fig.5(a), the dependences of $\mathcal{S}_{\theta^*_\kappa}(\kappa)$ on the number of epochs are shown. The red plot (in color print) is for the true key and the green plot is for the other 255 keys. Figure 5(b) shows histograms corresponding to cross-sectional views at epoch number 200. An arrow and a number indicate the position and the estimation rank by the true key. The cases where the true key did not take the first place at 200 epochs are enclosed in a dashed box. Figure 6 presents similar results but for $\mathtt{CNN_{best}}$.



(a) Dependence on number of epochs



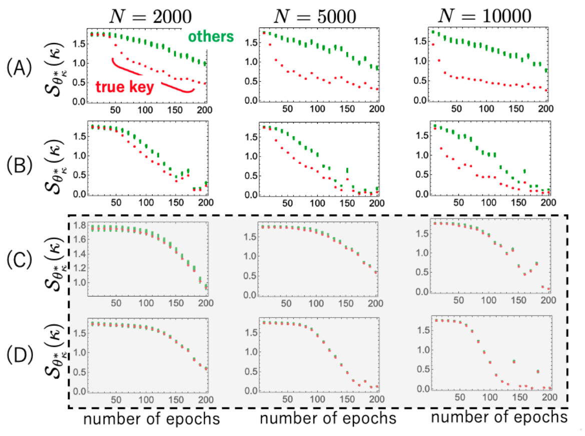
(b) Histogram at epochs= 200

FIGURE 5. S-protocol with $\mathtt{MLP_{best}}$.

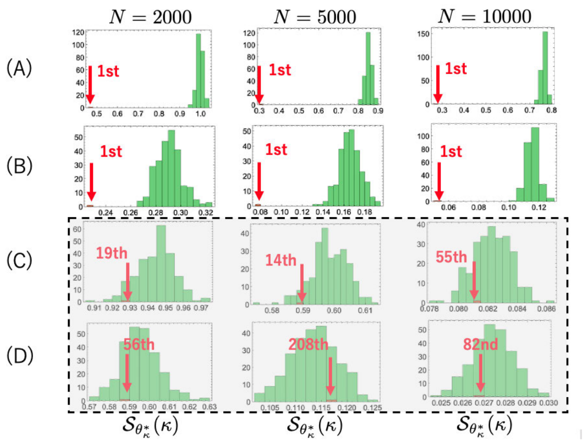
Figures 5 and 6 show that our approach works as envisioned. As expected from the general characteristics of machine learning, the larger the N, the clearer the separation of metrics by the true key, and the more favourable the situation for the attacker.

In addition, a comparison of row (B) in Fig.5 and 6, suggests that the attack can succeed with fewer traces for $\mathtt{CNN_{best}}$ than for $\mathtt{MLP_{best}}$, reflecting the general property that CNN is more resistant to translational misalignment than MLP. This means that our protocols benefit from using a high-level network that can capture and learn data features more faster because of a more complex structure. When using high-level networks, the training complexity is generally significant, and thus the effect of reduction in the number of network trainings is more critical. In other words, a more advanced network may be used when employing our approach.

For data (C) and (D), the estimation of the true key did not achieve first place around 200 epochs in both $\mathtt{MLP_{best}}$ and $\mathtt{CNN_{best}}$. To improve the visibility of Fig.6(a), $\mathcal{S}_{\theta^*_\kappa}(\kappa)$ may



(a) Dependence on epochs number



(b) Histogram at epochs= 200

FIGURE 6. S-protocol with $\mathtt{CNN_{best}}$.

be rescaled as

$$\Delta\mathcal{S}_{\theta^*_\kappa}(\kappa)=\frac{\mathcal{S}_{\theta^*_\kappa}(\kappa)-\mathcal{S}_{ave}}{\mathrm{SD}_S}\tag{24}$$

where

$$\mathcal{S}_{ave}=\frac{1}{|\mathcal{K}|}\sum_{\kappa\in\mathcal{K}}\mathcal{S}_{\theta^*_\kappa}(\kappa)$$

and

$$\mathrm{SD}_S=\sqrt{\frac{1}{|\mathcal{K}|}\sum_{\kappa\in\mathcal{K}}\left(\mathcal{S}_{\theta^*_\kappa}(\kappa)-\mathcal{S}_{ave}\right)^2}.$$

Through rescaling, the lower half of Fig.6(a) becomes Fig.7, showing that the estimation by the correct key achieved the first place around 100 epochs using the S-protocol with $\mathtt{CNN_{best}}$ when a sufficient number of traces is given. The region of the large epoch number where the estimation becomes unstable coincides with the area where overlearning occurs with $\mathtt{CNN_{best}}$, which was studied in detail in [21]. As with other DL applications, this indicates that the negative effect of overlearning needs to be carefully considered.

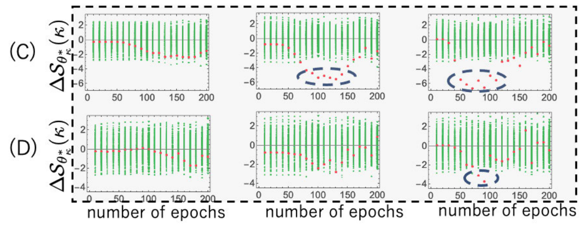


FIGURE 7. Rescaled version of the lower half of Fig.6(a).

2) T-PROTOCOL

Similar to the S-protocol, the T-protocol was applied with $\mathtt{MLP_{best}}$ and $\mathtt{CNN_{best}}$ to the four data types listed in Table 1. We obtained Figs.8,9, and 10 for the T-protocol corresponding to Figs.5,6, and 7 for S-protocol. $\Delta\mathcal{T}(\kappa)$ in Fig.10 is defined as

$$\Delta\mathcal{T}(\kappa)=\frac{\mathcal{T}(\kappa)-\mathcal{T}_{ave}}{\mathrm{SD}_\mathcal{T}}\tag{25}$$

where

$$\mathcal{T}_{ave}=\frac{1}{|\mathcal{K}|}\sum_{\kappa\in\mathcal{K}}\mathcal{T}(\kappa)$$

and

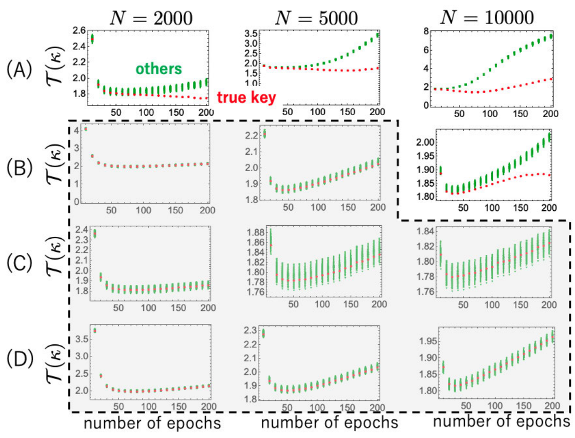
$$\mathrm{SD}_\mathcal{T}=\sqrt{\frac{1}{|\mathcal{K}|}\sum_{\kappa\in\mathcal{K}}(\mathcal{T}(\kappa)-\mathcal{T}_{ave})^2}$$

similar to (24).

As shown above, the T-protocol works similarly to or better than the S-protocol. In the case of $\mathtt{CNN_{best}}$, the analytical capability is significantly improved over that in the S-protocol. In addition to enabling the estimation of the true key with fewer N, it shows the robustness for overlearning, as discussed in the previous section. As introduced in Sec.III-C, the only difference between the S-protocol and T-protocol is the metric defined in (20) and (23), respectively; it should be stressed that the network training process is identical in both cases. In the S-protocol, we once transform the probability distribution on the plaintext space into another probability distribution on the Hamming weight space, and then introduce a metric based on the transformed probability distribution. At the same time, in the T-protocol, we obtain a natural metric directly based on the output probability distribution itself. During the transformation process in S-protocol, some information may be lost. In contrast, the information contained in the distribution can be fully utilised in the T-protocol. Thus, the T-protocol more fully utilises the networking capabilities gained in the training process.

V. DISCUSSION

Here we clarify that our trained network, as Timon's original case, can be recycled for an attack against the same type of device with different key materials. Let us consider a situation with two DUTs as exhibited in Fig.11. The two DUTs with different key materials are supposed to


(a) Dependence on the number of epochs.

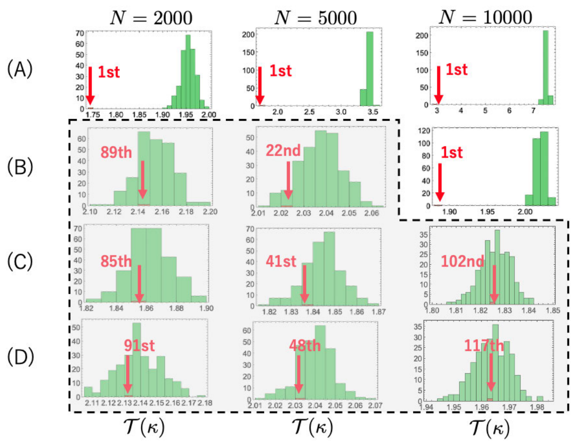
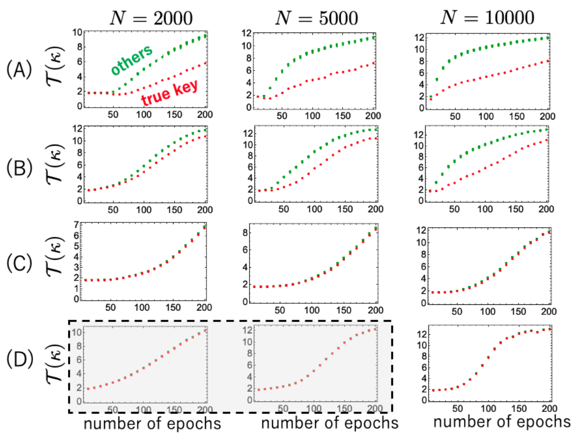

(b) Histogram at epochs= 200.

FIGURE 8. **T-protocol with** MLP$_{\text{best}}$.


(a) Dependence on the number of epochs

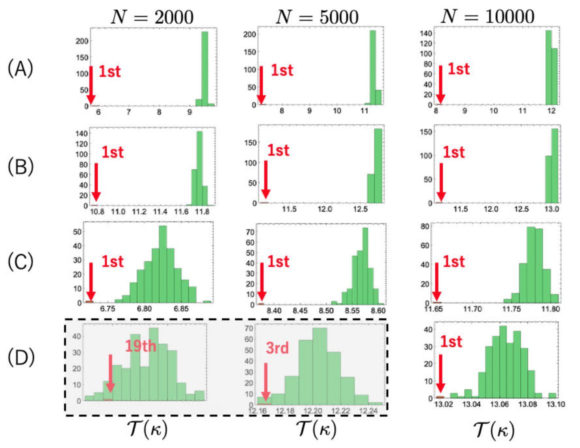

(b) Histogram at epochs= 200

FIGURE 9. **T-protocol with** CNN$_{\text{best}}$.

have similar physical characteristics, including their circuit implementations, because they are two products with the same specifications.

When we apply Timon's protocol to DUT0, we can have multiple trained networks for all key candidates; therefore we can pick a correctly trained network once we obtain the true key in the first attack. Then, using the picked network as a profiled network, we may attack DUT1 by immediately applying the attacking phase of profiled DL-SCA. Thus, we need neither the training phase nor reference devices for the second profiled DL-SCA. Such recycling can be performed rather straightforwardly with Timon's approach. However, the required computational resources, including the first attack, are considerable compared with the ordinary profiled DL-SCA; therefore it might not have a significant impact in practise.

Here we show that a similar recycling use of the trained network can also be possible in our case but implies a different impact from Timon's. In the first attack, we obtain a trained network that outputs $\rho^{(w)}_{\eta^*}(\xi)$ for the given trace w, as explained in (12). Once we obtain the true key k_0^* in the first attack, we may use (16) with k_0^* to acquire a probability distribution for the estimation of the output of the sbox

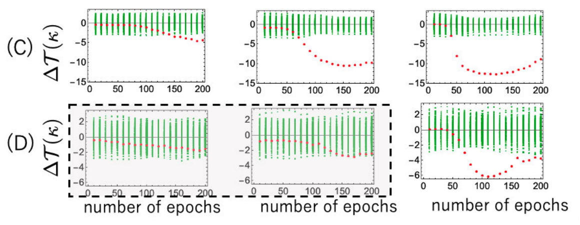


FIGURE 10. **Rescaled version of the lower half of Fig.9(a).**

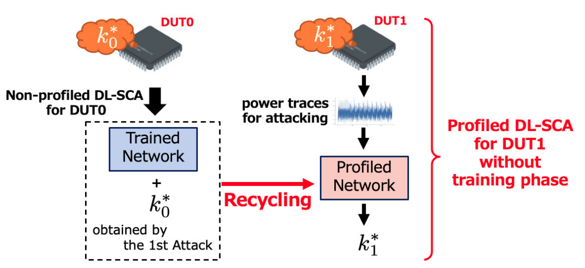

DUT0
DUT1
k_0^*
k_1^*
Non-profiled DL-SCA for DUT0
power traces for attacking
Trained Network
+
k_0^*
obtained by the 1st Attack
Recycling
Profiled Network
k_1^*
Profiled DL-SCA for DUT1 without training phase

FIGURE 11. **Recycling use of the trained network.**

function as

$$\nu^{(w)}_{\sigma^*_{k_0^*}}(r) \simeq \sum_{\xi \in \Xi_{r,k_0^*}} \rho^{(w)}_{\eta^*}(\xi) \qquad (26)$$

where

$$\Xi_{r,k_0^*} := \{\xi \mid \mathrm{sbox}(\xi \oplus k_0^*) = r\}. \tag{27}$$

The distribution is identical to the output of a profiled network in profiled DL-SCA with the same network settings, where the sbox function correctly labels the data. Then, we can execute the second attack. By inputting the trace w'_j from DUT1 into the profiled network, we obtain the probability distribution $\nu_{\sigma_{k_0^*}^*}^{(w'_j)}(r)$. Based on the distribution, we may introduce the following cross-entropic metric

$$U(\kappa) = -\sum_j \log \nu_{\sigma_{k_0^*}^*}^{(w'_j)}\left(\mathrm{sbox}\left(x'_j \oplus \kappa\right)\right) \tag{28}$$

and estimate the true key of DUT1 as

$$\arg\min_{\kappa \in \mathcal{K}} U(\kappa).$$

Notably, this recycling approach is similar to Timon's case, but with much lower training complexity.

The above can be understood more clearly when the sbox function is bijective, as in AES. Let us recall the training process, which is symbolically denoted as

$$\mathbf{DL}(\mathrm{Net}_\eta, D_{\mathrm{DUT}}, n_{epochs}) \to \mathrm{Net}_{\eta^*} \tag{29}$$

in Algorithms 2 and 3, where

$$D_{\mathrm{DUT}} := \left\{\langle x_j, w_j\rangle \mid j \in \{1, \cdots, N\}\right\}$$

as in (1). Network Net_{η^*} in (29), corresponds to the trained network in Fig.11. Let us consider a similar training process described as

$$\mathbf{DL}(\mathrm{Net}_\eta, D_{k^*}, n_{epochs}) \to \mathrm{Net}_{\eta_{k^*}^*} \tag{30}$$

where

$$D_{k^*} := \left\{\langle \mathrm{sbox}(x_j \oplus k^*), w_j\rangle \mid j \in \{1, \cdots, N\}\right\}. \tag{31}$$

It should be noted that $\mathrm{Net}_{\eta_{k^*}^*}$ in (30) corresponds to the network profiled on data correctly labelled by the sbox function. As there is a one-to-one correspondence between x_j and $\mathrm{sbox}(x_j \oplus k^*)$ because of the bijectivity of the sbox function, the above two training processes are equivalent to each other, except for the interpretation of the labeling values. To be more concrete, considering the two probability distribution outputs from the two networks with the same input w

$$\langle \mathrm{Net}_{\eta^*}, w\rangle \to \rho_{\eta^*}^{(w)}(\xi), \text{ and } \langle \mathrm{Net}_{\eta_{k^*}^*}, w\rangle \to \rho_{\eta_{k^*}^*}^{(w)}(\xi), \tag{32}$$

we can introduce a translation rule as

$$\rho_{\eta^*}^{(w)}(\xi) = \rho_{\eta_{k^*}^*}^{(w)}(\mathrm{sbox}(\xi \oplus k^*)), \tag{33}$$

and

$$\rho_{\eta_{k^*}^*}^{(w)}(\xi) = \rho_{\eta^*}^{(w)}(\mathrm{sbox}^{-1}(\xi) \oplus k^*). \tag{34}$$

Such a translation rule indicates the essential equivalence of the two networks, Net_{η^*} and $\mathrm{Net}_{\eta_{k^*}^*}$. This equivalence directly implies the precise equivalence of the computational complexities required to obtain the two networks.

The above argument shows that recycling in our protocol has a different impact than in Timon's. Our protocol can attack two DUTs without using reference devices with about the same computational complexity as a standard profiled attack on one DUT. Unlike Timon's, it gives a non-trivial and significant meaning to the recycling scenario.

VI. SUMMARY

Herein, we proposed new protocols for non-profiled DL-SCA that are improvements over Timon's protocol, resulting in a decrease in the network training complexity to the order one. The training complexity is identical to that of profiled DL-SCA, implying that non-profiled DL-SCA is superior to profiled DL-SCA in that it requires no reference device for profiling before analysing the target device.

We want to emphasize that the primary point of our research is to present a way to use networks wisely. Our approach generally applies to many network models that output guess probability distributions. In other words, the analytical capability itself depends on the network employed rather than on the proposed method. However, our approach reduces the learning complexity required for the overall analysis. In this sense, it facilitates the adoption of network models with high analytical capability but learning costs, allowing for more robust analysis.

Our protocols can be used for research to improve security, particularly for experimental evaluations of SCA countermeasures. The proposed protocols are expected to make these hands-on analyses more efficient and accelerate research and development in this area. We hope to contribute to the improvement in hardware security through this research.

Note that some of the contents of this article have been reported by the same authors as preliminary results in the non-refereed proceedings of a domestic workshop [25].

REFERENCES

[1] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2004, pp. 16–29.
[2] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual information analysis: A comprehensive study," *J. Cryptol.*, vol. 24, no. 2, pp. 269–291, Apr. 2011.
[3] B. Gierlichs, L. Batina, B. Preneel, and I. Verbauwhede, "Revisiting higher-order DPA attacks," in *Topics in Cryptology—CT-RSA*. Berlin, Germany: Springer, 2010, pp. 221–234.
[4] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1999, pp. 388–397.
[5] P. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer, 1996, pp. 104–113.
[6] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2002, pp. 13–28.
[7] J. Doget, E. Prouff, M. Rivain, and F.-X. Standaert, "Univariate side channel attacks and leakage modeling," *J. Cryptograph. Eng.*, vol. 1, no. 2, pp. 123–144, Aug. 2011.
[8] W. Schindler, "Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking," *J. Math. Cryptol.*, vol. 2, no. 3, pp. 291–310, Jan. 2008.

[9] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Cryptographic Hardware and Embedded Systems—CHES*. Berlin, Germany: Springer, 2005, pp. 30–46.

[10] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *Cryptographic Hardware and Embedded Systems—CHES*. Cham, Switzerland: Springer, 2017, pp. 45–68.

[11] L. Lerman, R. Poussier, O. Markowitch, and F.-X. Standaert, "Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version," *J. Cryptograph. Eng.*, vol. 8, no. 4, pp. 301–313, Nov. 2018.

[12] Z. Martinasek, P. Dzurenda, and L. Malina, "Profiling power analysis attack based on MLP in DPA contest V4.2," in *Proc. 39th Int. Conf. Telecommun. Signal Process. (TSP)*, Jun. 2016, pp. 223–226.

[13] Z. Martinasek, J. Hajny, and L. Malina, "Optimization of power analysis using neural network," in *Smart Card Research and Advanced Applications*. Cham, Switzerland: Springer, 2004, pp. 94–107.

[14] Z. Martinasek, L. Malina, and K. Trasy, "Profiling power analysis attack based on multi-layer perceptron network," in *Computational Problems in Science and Engineering*, vol. 343. Cham, Switzerland: Springer, 2015, pp. 317–339.

[15] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *Security, Privacy, and Applied Cryptography Engineering*. Cham, Switzerland: Springer, 2016, pp. 3–26.

[16] S. Picek, A. Heuser, A. Jovic, S. A. Ludwig, S. Guilley, D. Jakobovic, and N. Mentens, "Side-channel analysis and machine learning: A practical perspective," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 4095–4102.

[17] A. Alipour, A. Papadimitriou, V. Beroulle, E. Aerabi, and D. Hély, "On the performance of non-profiled differential deep learning attacks against an AES encryption algorithm protected using a correlated noise generation based hiding countermeasure," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 614–617.

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

[19] B. Timon, "Non-profiled deep learning-based side-channel attacks with sensitivity analysis," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2019, pp. 107–131, Feb. 2019.

[20] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray, "Advanced encryption standard (AES)," Federal Inf. Process. Standards (NIST FIPS)-197, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep., 2001.

[21] 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.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

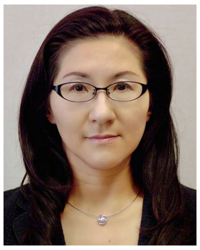
[23] Y. LeCun and F. J. Huang, "Loss functions for discriminative training of energy-based models," in *Proc. Mach. Learn. Res.*, vol. 5, 2005, pp. 206–213.

[24] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[25] K. Imafuku, S. Kawamura, H. Nozaki, J. Sakamoto, and S. Osuka, "Revisit of non profiled side channel analysis via deep learning," *IEICE Tech. Rep.*, vol. 122, no. 125, pp. 7–12, 2022.

KENTARO IMAFUKU received the Ph.D. degree in physics from Waseda University, in 2000. After professional experiences with Waseda University (Research Associate), the University of Rome Tor Vergata (JST/JSPS Overseas Young Research Fellowship), and The University of Tokyo (Postdoctoral Researcher), he is currently with the National Institute of Advanced Industrial Science and Technology, Tokyo, Japan. He is conducting theoretical research on applications of physics and information theory to computer science and engineering.

SHINICHI KAWAMURA (Fellow, IEEE) received the B.E., M.E., and D.E. degrees in electronic engineering from The University of Tokyo, Tokyo, Japan, in 1983, 1985, and 1996, respectively. He was with Toshiba Corporation, in 1985, became a Senior Fellow, in 2012, and retired from the company, in 2020. From 1992 to 1994, he was a Visiting Researcher with the Center for Telecommunications Research, Columbia University, NY, USA. From 2009 to 2011, he was the Deputy Director with the Research Center for Information Security, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo. From 2012 to 2014, he was an Invited Senior Researcher with the Research Institute for Secure Systems, AIST. He is currently the Deputy Director with the Cyber Physical Security Research Center, AIST. His research interest includes cryptography and its application to security systems. He is a fellow of IEICE, a Senior Member of IPSJ, and a member of IACR. He was a recipient of the IEICE Achievement Award, in 2021, an IPSJ Specially Selected Paper Certificate, in 2014, the Contribution Award from the IEICE Engineering Sciences Society, in 2006, and the IEICE Young Researcher's Award, in 1993.



HANAE NOZAKI received the B.S. and M.S. degrees in physics and the Ph.D. degree in science from Ochanomizu University, Tokyo, Japan, in 1988, 1990, and 1998, respectively. Then, she joined the Corporate Research and Development Center, Toshiba Corporation. She is currently an Invited Senior Researcher with the Cyber Physical Security Research Center, National Institute of Advanced Industrial Science and Technology (AIST). Her current research interests include cryptographic implementation and side-channel analysis.

JUNICHI SAKAMOTO received the M.I.S. and Ph.D. degrees from Yokohama National University, Japan, in 2017, and 2020, respectively. He is currently a Researcher with Yokohama National University and the National Institute of Advanced Industrial Science and Technology. He has engaged in various researches with respect to hardware security, efficient implementations of the cryptographic algorithms, side-channel attacks, and laser-based fault attack. His research interests include remote side-channel attacks and hardware implementation of homomorphic encryptions.

SAKI OSUKA received the M.E. and Ph.D. degrees from the Nara Institute of Science and Technology, Nara, Japan, in 2019 and 2022, respectively. She is currently a Researcher with the National Institute of Advanced Industrial Science and Technology. Her research interests include electromagnetic compatibility and information security.