DOI:10.1587/transfun.2024CIP0014

Publicized:2024/09/11

PAPER *Special Section on Cryptography and Information Security*

CA-SCA: Non-profiled Deep Learning-based Side-channel Attacks by using Cluster Analysis

Yuta FUKUDA[†a)], ***Student Member***, **Kota YOSHIDA**[††], ***and*** **Takeshi FUJINO**[††], ***Members***

SUMMARY Differential deep learning analysis (DDLA) was proposed as a side-channel attack (SCA) with deep learning techniques in non-profiled scenarios at TCHES 2019. In the proposed DDLA, the adversary sets the LSB or MSB of the intermediate value in the encryption process assumed for the key candidates as the ground-truth label and trains a deep neural network (DNN) with power traces as an input. The adversary also observes metrics such as loss and accuracy during DNN training and estimates that the key corresponding to the best-fitting DNN is correct. One of the disadvantages of DDLA is the heavy computation time for the DNN models because the number of required models is the as same as the number of key candidates, which is 256 in the case of AES. Therefore 4096 DNNs are required for revealing keys of 16 bytes. Furthermore, the DNN models have to be trained again if the adversary changes a ground-truth label function from LSB to other labels such as MSB or HW. We propose a new deep-learning-based SCA in a non-profiled scenario to solve these problems. Our core idea is to extract feature of the leakage waveform using DNN. The adversary reveals the correct keys by conducting cluster analysis using the feature vectors extracted from power traces using DNN. We named this method as CA-SCA (cluster-analysis-based side-channel attacks), it is advantageous that only one DNN needs to be trained to reveal all key bytes. In addition, once the DNN is trained, multiple label functions can be tested without the additional cost of training DNNs. We provide four case studies of attacking against AES, including two software implementations and two hardware implementations. Our attacks against software implementations provide methods using a concatenated dataset that efficiently train the DNN. Also, our attack on the hardware implementation introduces multitask learning to exploit the Hamming distance leakage model. The results show that the proposed method requires fewer waveforms to reveal all key bytes than DDLA owing to the efficient learning performance on the above methods. Comparing the computation time to process the same number of waveforms, the proposed method requires only about 1/75 and 1/25 of the time when attacking software and hardware implementations, respectively, due to the significant reduction in the number of training models.

key words: *side-channel attacks, deep-learning, cluster analysis.*

1. Introduction

1.1 Background and related works

Side-channel attacks (SCAs) reveal secret information, such as cryptographic keys, by observing leakage waveforms, such as power consumption and electromagnetic radiation. The adversary constructs a statistical model between leakage waveforms and values calculated from the internal state of the encryption process The methods used to calculate intermediate values depend on the assumed leakage models and implementation of the target device. When attacking against software implementation, Hamming weight (HW) of intermediate value is used most often. On the other hand, when attacking against hardware implementation, Hamming distance (HD) between registers on consecutive clock edges is used most often.

SCA involves two types of scenarios: profiled and non-profiled attacks. In a profiled scenario, the adversary uses a profiled device whose cryptographic key is known. The adversary constructs a statistical model using leakage waveforms acquired from the profiled device (profiling phase) and reveals the cryptographic key in a targeted device by utilizing the model (attack phase). This approach has the drawback that the experimental conditions for acquiring waveforms must be consistent between the profiling and attack phase. Conventional attacks in profiled scenarios include template attacks [1]. In a non-profiled scenario, the adversary has the advantage of not requiring a profiled device, but has the difficulty of building a statistical model of side-channel information from the targeted device. Typical attacks in non-profiled scenarios include differential power analysis focusing on the difference of leakage waveforms [2] and correlation power analysis that assumes a correlation between leakage waveforms and intermediate value [3]. In this paper, we deal with non-profiled scenarios.

Since 2016, deep-learning-based side-channel attacks (DL-SCAs) have been discussed as an approach that differs from conventional SCAs [4] [5] [6] [7] [8]. In many cases, the attack based on the profiled scenario had been studied. On the other hand, differential deep learning analysis (DDLA) was proposed in the non-profiled scenario by Timon [9]. In DDLA, the adversary must train as many deep neural networks (DNNs) as key candidates. The DNN is trained to predict the intermediate values calculated from plaintext and each key candidate. Then, the adversary guesses that the key corresponding to the DNN model with the best learning metrics (loss, accuracy, etc.) among those models is the correct key. Timon evaluated its attack performance with ASCAD, which is a public dataset [10], and software-implemented AES without SCA countermeasures in Atmel XMEGA128, software-implemented AES with jitterling countermeasure, and software-implemented AES with second-order masking. Kuroda et al. studied practical DDLA aspects against software-implemented AES with two kinds of masking countermeasure, including rotating s-boxes masking, and the full key bytes (16 bytes) were successfully revealed [11] [12].

[†]The author is with the Graduate School of Science and Technology, Ritsumeikan University, Kusatsu-shi, 525-8577, Japan.
[††]The author is with Department of Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525-8577, Japan.
a) E-mail: ri0073pi@ed.ritsumei.ac.jp

Alipour et al. provided hiding countermeasures to interfere with model learning in DDLA [13]. Kwon et al. applied an early stopping method to prevent over-fitting in DDLA [14]. They also improved the speed by training DNNs in parallel. Hoang et al. introduced DDLA based on multi-output classification [15]. This enables faster attacks than parallel networks. Do et al. also introduced DDLA using multi-output classification (MOC) and multi-output regression (MOR) [16]. It performs faster and achieves higher attack performance than single-output approaches. Do et al. investigated DNN models for DDLA and the effect of hiding countermeasures due to noise generation [17]. Meanwhile, Wu et al. took a different approach which used plaintext labeling to conduct non-profiled DL-SCA [18].

1.2 Our contribution

In this study, we propose a non-profiled DL-SCA based on an approach different from DDLA called CA-SCA (cluster-analysis-based side-channel attacks). CA-SCA uses DNN to extract the feature of waveform and reveals the key via cluster analysis of extracted feature vectors. We provide a method using two kinds of DNNs, autoencoder (AE) and convolutional neural network (CNN). In our attacks against software implementations, we provide an efficient method for training DNNs by taking advantage of the fact that encryption is a sequential process. In our attacks against hardware implementations, we introduce DNNs to handle multiple tasks in order to focus on the HD leakage model.

The contributions of this study are summarized as follows.

- We propose a new non-profiled DL-SCA called CA-SCA which is a different approach from DDLA. We provide two types of attacks: AE-based attacks and CNN-based attacks. The latter method which train CNNs using plaintext as the ground-truth label can successfully revealed the software- and hardware- implemented AES protected with SCA countermeasures.
- For attacking software-implemented AES, a ”concatinated dataset” procedure in which one waveform is divided into 16 datasets is applied. This approach take advantage of the fact that the software-implemented AES is sequentially processed byte-by-byte. This method improves the training efficiency of the DNN, and only one DNN training is required to reveal 16. We provide two case studies on software implementation (Case study S-1 and S-2). Case study S-1 involves AES without SCA countermeasures. Case study S-2 involves the ASCAD database.
- For attacking hardware-implemented AES, we apply a ”multi-task learning method” where one DNN outputs 16 labels. This method is advantageous for training a single waveform containing 16 bytes parallel processing. Furthermore, this method also reveal 16 byte keys by training a single model. We also provide two case studies focusing on hard- ware implementation (Case study H-1 and H-2). Case study H-1 involves ASIC-implemented AES without SCA countermeasures. Case study H-2 involves ASIC- implemented AES with RSM countermeasures.
- From the computation times of CA-SCA measured in these 4 case studies, it is clear that the time required to reveal 16-byte keys is significantly reduced. The main reason for these results is that the number of DNN models is 1 for CA-SCA compared to 4096 for DDLA.

This paper is an extended version of [19] that adds attacks against hardware implementations and comparisons to DDLA variants.

1.3 Paper organization

The remainder of this paper is organized as follows. Section 2 provides preliminary information on non-profiled attacks such as CPA and DDLA. Section 3 describes a new non-profiled attack using cluster analysis as the proposed method. Section 4 provides two case studies describing attacks against software implementations. Section 5 provides two case studies describing attacks against hardware implementations. Section 6 discusses comparisons with other non-profiled attack methods. Section 7 summarizes our work.

2. Preliminary

2.1 Correlation power analysis

Correlation power analysis (CPA) is a side-channel attack that does not use deep-learning techniques in a non-profiled scenario [3]. The following Pearson's correlation coefficients p are used in this attack.

$$p = \frac{\mathbb{E}[(W-\mu_W)(H-\mu_H)]}{\sqrt{\mathbb{E}[(W-\mu_W)^2]\cdot\mathbb{E}[(H-\mu_H)^2]}}, \tag{1}$$

where W, H, μ_W, μ_H, and $\mathbb{E}[\cdot]$ denote the waveform, the intermediate value when the key is assumed, the average value of W, the average value of H, and the expectation. The adversary calculates the correlation coefficient p for each candidate key and estimates that the key corresponding to the highest p is the correct key.

Why CPA works well?: Targets without SCA countermeasures correlate with the internal state during the encryption process (e.g., Hamming weight (HW) of intermediate values, Hamming distance (HD) of register transitions) and the leakage waveform. When assuming the incorrect key, there is no correlation between the leakage waveform and the internal state considering the key. In contrast, when assuming the correct key, there is a correlation between the leakage waveform and the HW/HD of the intermediate value considering the key. Therefore, an adversary can estimate the key by comparing each correlation coefficient. However, the adversary cannot attack when there is no correlation between the waveform and HW/HD of the intermediate value due to some countermeasures such as masking or hiding.

2.2 Differential deep-learning analysis

In TCHES 2019, Timon proposed differential deep-learning analysis (DDLA) as deep-learning-based side-channel analysis in non-profiled scenarios [9]. The adversary prepares the DNN models $\mathcal{M}_{\theta_k}$ with the same number of candidate keys, where each DNN is trained with intermediate values $\mathrm{I}(k, \mathbb{P})$ as ground-truth labels calculated from plaintext $\mathbb{P}$ and each candidate key k. Cryptographic keys are estimated by comparing the evaluation metrics (loss and accuracy) when each model is trained. When focusing on loss, the candidate key corresponding to the model with the lowest loss score is estimated as the correct key. $L(\mathcal{M}_{\theta_k}(\mathbb{T}), \mathrm{I}(k, \mathbb{P}))$ denotes the loss with the ground-truth label $\mathrm{I}(k, \mathbb{P})$ when the leakage waveforms $\mathbb{T}$ are input to the DNN model $\mathcal{M}_{\theta_k}$ with model parameter θ_k. When focusing on accuracy, the candidate key corresponding to the model with the highest accuracy score is estimated as the correct key. The algorithm of DDLA, when focusing on loss metrics, is shown in Algorithm 1, where $\mathrm{I}(\cdot)$ denotes the label computation function, which is used to compute intermediate values during the encryption process using the plaintext $\mathbb{P}$ and key candidate k. In the case of attacks against software-implemented AES at the 1st round of encryption, $\mathrm{I}(\cdot)$ is given as follows.

$$\mathrm{I}(k, \mathbb{P}) = \mathrm{LSB}(\text{S-Box}(k \oplus \mathbb{P})), \tag{2}$$

where k denote the key candidate, $\mathbb{P}$ denote the plaintext, S-Box$(\cdot)$ denote the S-Box function of the AES and $\mathrm{LSB}(\cdot)$ is a function used to get the least significant bit (LSB) when the argument is converted to binary. Adversary can change $\mathrm{LSB}(\cdot)$ to $\mathrm{MSB}(\cdot)$ that means the most significant bit (MSB).
Advantage of DDLA: In this attack, the DNN builds the relationship between leakage waveforms and intermediate values from scratch during the key-dependent encryption process. Unlike 1st-order CPA, this relationship is not limited to correlation at the single point in the time series waveform. For example, it is possible to construct a DNN that combines the features of the two points where the mask and masked values are processed during masking countermeasures. The relationship exists when the key candidate is correct and the training loss is reduced. In contrast, when the key candidate is incorrect, the relationship does not exist, and the training loss is not reduced. Therefore, the adversary can estimate the key by comparing the training loss.
Disadvantage of DDLA: DDLA requires 256 × 16 models to reveal all key bytes (128bit) in AES. Thus, a very long computation time is required. Furthermore, the re-training of the model is necessary when the adversary attempts to change the label function applied to the intermediate values.

3. Proposed method

3.1 Core idea

We propose a non-profiled SCA method via cluster analysis of feature vectors which are extracted from the waveforms

Algorithm 1 Differential deep-learning analysis (using loss metrics)

```
Input: Leakage Traces $\mathbb{T}$, Plain-text $\mathbb{P}$, Key space $\mathcal{K}$,
    Number of epoch $N_{\mathrm{ep}}$, DNN model parameters $\theta_k\,(k \in \mathcal{K})$,
    Base DNN model parameters $\theta_{\mathrm{base}}$
Output: Estimated key $k^*$
 1: for $k$ in $\mathcal{K}$ do
 2:     $\theta_k = \theta_{\mathrm{base}}$
 3:     for epoch = 1 to $N_{\mathrm{ep}}$ do
 4:         Training $\mathcal{M}_{\theta_k}$ with $(\mathbb{T}, \mathrm{I}(k, \mathbb{P}))$
 5:         $\mathrm{l}_{\mathrm{loss}}[N_{\mathrm{ep}}] = L(\mathcal{M}_{\theta_k}(\mathbb{T}), \mathrm{I}(k, \mathbb{P}))$
 6:     end for
 7:     $\mathrm{L}_{\mathrm{loss}}[k] = \min \mathrm{l}_{\mathrm{loss}}$
 8: end for
 9: $k^* = \arg\min_i \mathrm{L}_{\mathrm{loss}}[i]$
10: return $k^*$
```

by using deep-learning techniques. We called this method CA-SCAs.

The core ideas of CA-SCAs are as follows. Sufficiently trained DNN models obtain a relationship between the acquired waveforms and the cryptographic intermediate values. We expect that the distribution of the feature vectors extracted by DNNs to follow the intermediate values. The feature vectors are distributed based on the correct intermediate values when the adversary's hypothetical key is correct, as shown $\hat{k}$ = correct_key in Fig. 1. In contrast, the feature vectors are mixed by the incorrect intermediate values when the adversary's hypothetical key is incorrect, as shown $\hat{k}$ = 0x00 in Fig. 1. As described above, an adversary can select the most suitable relationship between the feature vectors and intermediate values from the key candidates by making hypotheses about the relationship and comparing the fitness of each hypothesis. Each fitness is calculated using the Calinski-Harabasz index, which is one of the metrics used to evaluate clusters, described in section 3.2.

CA-SCAs offer the following advantages.

- The number of deep learning models trained for feature extraction is greatly reduced compared to DDLA, which requires many deep learning models. This is highly effective for the reduction of computing resources for model training.
- The adversary can try many kinds of cluster analysis by changing different leakage models such as HW and HD (details will be described in section 3.3) once the feature vectors have been extracted from DNN models.
- SCA countermeasures such as masking can be successfully analyzed by introducing supervised training on models for feature extraction.

The basics of CA-SCAs are explained below. An overview of CA-SCAs is shown in Fig. 1. First, the adversary trains a DNN model for feature extraction of the leakage waveform. Next, the adversary extracts feature vectors from the leakage waveforms by using the trained DNN. It is noted that each feature vector is corresponding to the intermediate values which is determined by plaintext and key candidate. Finally, the adversary conducts cluster analysis on the feature

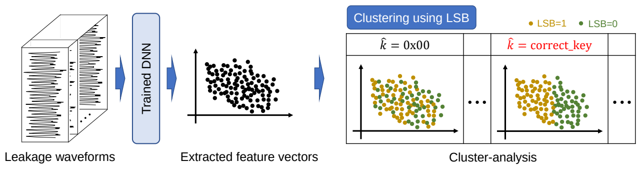


Fig. 1: Overview of cluster-analysis-based side-channel attacks (CA-SCAs)

vectors by using the label (i.e, LSB, MSB, HW, HD) of the intermediate values. Fig. 1 shows the schematic diagram of cluster analysis using LSB. If the labeling is performed with the correct key, the dots with the same label are clustered together, and the dots with different labels are plotted apart from each other as shown $\hat{k}$ = correct_key in the figure. On the other hand, the dots are randomly scattered in case of incorrect key as shown as $\hat{k} = 0x00$ in the figure. The adversary estimates the candidate key corresponding to the most appropriately clustered plot to be the correct key.

The quantitative score used in the cluster analysis is the Calinski-Harabasz index, which is described in section 3.2. In this study, auto-encoder (AE) and convolutional neural network (CNN) are used as DNN models, and their respective attack procedures are described in sections 3.4 and 3.5, respectively.

3.2 Calinski-Harabasz index

We used the Calinski-Harabasz index (CH index) for cluster analysis. The CH index is the variance ratio criterion, and a higher score indicates a better-defined cluster [20]. Given a dataset of N points: $\{x_1, \cdots, x_N\}$, and the assignment of these points to k clusters: $\{C_1, C_2, \cdots, C_k\}$, the CH index is formulated as follows.

$$\text{CHindex} = \frac{B_k}{W_k} \times \frac{N-k}{k-1}, \tag{3}$$

$$B_k = \sum_{i=1}^{k} n_i ||m_i - m||_2^2, \tag{4}$$

$$W_k = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - m_i||_2^2, \tag{5}$$

where B and W between-class variance, and within-class variance, respectively. where m_i, m, and n_i denote the center of cluster C_i, the overall center of source points, and the number of points assigned to i-th cluster, respectively. Meanwhile, $||x - m_i||_2$ and $||m_i - m||_2$ denote the L2 norm of the two vectors. In this paper, given a set of feature-extracted vectors $\mathcal{X} = \{x_1, \cdots, x_N\}$ and a set of their labels $\mathcal{L} = \{l_1, \cdots, l_N\}$; the label represents the assignment of the corresponding waveform to clusters. The function to calculate the CH index is denoted as $\text{CH}(\mathcal{X}, \mathcal{L})$. In our experiments, we used a scikit-learn implementation[†], where the CH index is calculated with multivariate data as input data $\mathcal{X}$.

The score has no upper limit. A higher score indicates that the cluster is well formed. In contrast, a score closer to 0 indicates an inappropriate cluster, i.e., a mixture of classes.

3.3 Selected function for calculation of CH index

Once a DNN model is trained for feature extraction, CA-SCA can try multiple attacks using various leakage models. The calculation of the CH index requires labels indicating which cluster the input vector belongs to, and the function that calculates the labels can be changed. The leakage functions for CH index used in this study are explained below.

When attacking a software-implemented application on a microcontroller, the adversary typically uses the following intermediate value.

$$v = \text{S-Box}(k_t \oplus \mathbb{P}_t), \tag{6}$$

where S-Box$(\cdot)$, k_t, and $\mathbb{P}_t$ denote the S-Box function, the t-th byte cryptographic key, and the t-th byte plaintext, respectively.

Although the intermediate value itself can be used as labels, three labeling functions from the intermediate values were used in this study and are described below.

- **Hamming weight of intermediate value:** The adversary conducts cluster analysis on the label of HW, as same as attack with CPA. This exploits the fact that the value of the all microcontroller's precharge bus affects the power consumption. In this case, the number of label is nine from HW=0 to HW=8. In this paper, the function is defined as follows.

  $$f_{\text{hw}}(v) = \text{HW}(v), \tag{7}$$

  where HW$(\cdot)$ is a function that calculates the Hamming weight of its arguments. CH index s focused on HW labeling is computed using the feature-extracted vectors $\mathcal{X}$ as follows.

  $$s = \text{CH}(\mathcal{X}, f_{\text{hw}}(v)). \tag{8}$$

- **Mono-bit including LSB/MSB:** The adversary conducts cluster analysis in mono-bit labeling, including LSB/MSB, as it does when attacking with DDLA. This exploits the fact that the value of the single microcontroller's precharge bus affects the power consumption. For example, in the case of LSB labeling, there are two classes which are LSB=0 group and LSB=1 group. In this paper, the function is defined as follows.

  $$f_{\text{mono}}(v, b) = v \oplus 2^b, \tag{9}$$

  where b is the bit position to be selected. For example, when LSB labeling is selected, b is 0. CH index s focused on LSB labeling is computed using the feature-extracted vectors $\mathcal{X}$ as follows.

  $$s = \text{CH}(\mathcal{X}, f_{\text{mono}}(v, 0)). \tag{10}$$

[†]See the following URL for a detailed implementation.
`https://scikit-learn.org/stable/modules/clustering.html`

When all bits (i.e., multi-bit) are used, the CH index s is the sum of the respective CHindex, as in Eq.(11).

$$s = \sum_{b=0}^{7} \mathrm{CH}(\mathcal{X}, f_{\mathrm{mono}}(v, b)), \tag{11}$$

where $\mathcal{X}$, v and b denote the feature-extracted vectors, the value calculated using Eq. (6), and the bit position, respectively. The adversary calculated and compared these sum values for each candidate key.

When attacking against a hardware-implemented application on FPGA or ASIC, the adversary typically uses transition value of register. This exploits the fact that the transition of the register on the device affects power consumption. In general, the adversary focuses on the last round (10th round in case of AES-128) of AES when they know the ciphertext $\mathbb{C}$. The transition value of register is formulated as follows.

$$v = \mathbb{C}_{s(t)} \oplus \text{S-Box}^{-1}(k_t \oplus \mathbb{C}_t), \tag{12}$$

where $\text{S-Box}^{-1}(\cdot)$, k_t, $\mathbb{C}_t$, and $\mathbb{C}_{s(t)}$ denote the inverse S-Box function, the t-th cryptographic key, the t-th ciphertext, and the ciphertext of corresponding to the target byte t considering ShiftRows, respectively. In this case, it is possible to conduct CA-SCA by adapting the register transition v to the above three label functions.

3.4 Procedure of CA-SCA with autoencoder

Auto-encoder (AE) is a type of DNN for achieving feature extraction via unsupervised training [21]. AE learns to make the input and output the same, then the difference between them are often used for anomaly detection. In this study, we train AE by the leakage waveforms to obtain feature vectors which will be used for clustering on the CA-SCAs.

AE consists of an encoder and decoder, as shown in Fig. 2. The input data are compressed into latent variables (feature vectors) using the encoder and then reconstructed to its original dimensions using the decoder. AE is unsupervised learning in which the loss between the output and input data is minimized.

The attack procedure of CA-SCA with AE is shown in Algorithm 2. First, an AE consisting of an encoder $\mathcal{M}_{\theta_e}$ with parameters θ_e and a decoder $\mathcal{M}_{\theta_d}$ with parameters θ_d is trained unsupervised using waveforms set $\mathbb{T}$ (lines 1-3 in algorithm 2). Next, the waveforms in the S-Box process corresponding to the target byte are input to the trained encoder $\mathcal{M}_{\theta_e}$, and their latent variables $\mathcal{X}$ are calculated (line 4 in algorithm 2). Finally, the adversary performs a cluster analysis of $\mathcal{X}$ using the label computed from the candidate keys and plaintext according to the assumed leakage model (e.g., LSB of S-Box out in the first round). This cluster analysis is performed for each candidate key, and the candidate key with the highest CH index is estimated to be the correct key.

3.5 Procedure of CA-SCA with convolutional neural network

CA-SCA, described in the previous section, can be extended to a method using supervised learning. One method of supervised learning is convolutional neural networks (CNNs). The procedure of CA-SCA with CNNs is described below.

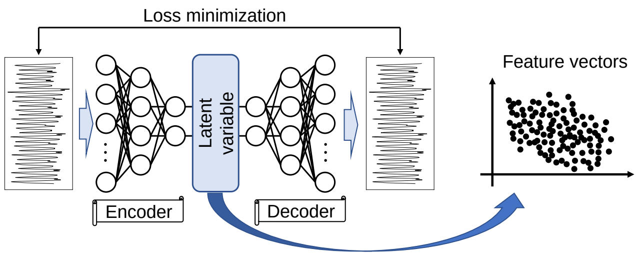


Fig. 2: Feature vectors extraction using auto-encoder in CA-SCA

Algorithm 2 Cluster-analysis-based side-channel attacks with auto-encoder

```
Input: Leakage Traces $\mathbb{T}$, Plain-text $\mathbb{P}$, Key space $\mathcal{K}$,
    Number of epoch $N_{ep}$, Encoder parameter $\theta_e$,
    Decoder parameter $\theta_d$
Output: Estimated key $k^*$
 1: for epoch = 1 to $N_{ep}$ do
 2:     Training $\mathcal{M}_{\theta_e}$, $\mathcal{M}_{\theta_d}$ with $\mathbb{T}$          // Training of AE
 3: end for
 4: $\mathcal{X} = \mathcal{M}_{\theta_e}(\mathbb{T})$    //Extract feature vectors (Latent Variables) on AE
 5: for target_byte = 0 to 15 do
 6:     for $k$ in $\mathcal{K}$ do
 7:         $v = \text{S-Box}(k \oplus \mathbb{P}_{\text{target\_byte}})$
 8:         $\mathrm{S_{cal}}[k] \leftarrow$ CH-index is calculated with $v$ and $\mathcal{X}$ by Eq. (8), (10),
            or (11)
 9:     end for
10:     $k^*[\text{target\_byte}] = \arg\max_i \mathrm{S_{cal}}[i]$
11: end for
12: return $k^*$
```

It is necessary to consider how to label the leakage waveforms since CNN is supervised learning that requires ground-truth label. CA-SCA with CNN uses plaintext/ciphertext as labels. Plaintext is used for 1st round in software implementation and ciphertext is used for last round in hardware implementation. Here, we explain why plaintext labeling is effective in CA-SCA against software-implemented AES. SCAs against software-implemented AES generally focus on the S-Box output v at the first round shown below.

$$v = \text{S-Box}(k^* \oplus \mathbb{P}), \tag{13}$$

where k^*, $\mathbb{P}$ and $\text{S-Box}(\cdot)$ denote correct key, plaintext, and S-Box function. In case of the DDLA training, the intermediate value v is used as the supervised output label. In our CA-SCAs, the plaintext $\mathbb{P}$ is used as the supervised output label to avoid different labels for each key candidate, but it has been described in [22] that a similar training model can be obtained.

The procedure of CA-SCAs using CNN is explained using Algorithm 3. At first, a CNN consisting of a feature extraction layer $\mathcal{M}_{\theta_f}$ and a classification layer $\mathcal{M}_{\theta_c}$ as shown 3 is trained using a waveform set $\mathbb{T}$ and plaintext $\mathbb{P}$ (lines 1-3 in algorithm 3). The trained CNN outputs the

Algorithm 3 Cluster-analysis-based side-channel attacks with convolutional neural network

```
Input: Leakage Traces $\mathbb{T}$, Plaintext $\mathbb{P}$, Key space $\mathcal{K}$,
    Number of epoch $N_{\mathrm{ep}}$, feature extractor layers parameter $\theta_{\mathrm{f}}$,
    classification layers parameter $\theta_{\mathrm{c}}$
Output: Estimated key $k^*$
 1: for epoch = 1 to $N_{\mathrm{ep}}$ do
 2:     Training $\mathcal{M}_{\theta_{\mathrm{f}}}, \mathcal{M}_{\theta_{\mathrm{e}}}$ with $(\mathbb{T}, \mathbb{P})$      // Supervised training of CNN
 3: end for
 4: $\mathcal{X} = \mathcal{M}_{\theta_{\mathrm{f}}}(\mathbb{T})$
            // $\mathcal{X}$ is the feature vectors which is used for cluster-analysis.
 5: for target_byte = 0 to 15 do
 6:     for $k$ in $\mathcal{K}$ do
 7:         $v$ = S-Box$(k \oplus \mathbb{P}_{\mathrm{target\_byte}})$
 8:         $\mathrm{S_{cal}}[k] \leftarrow$ CH-index is calculated with $v$ and $\mathcal{X}$ by Eq. (8), (10),
            or (11)
 9:     end for
10:     $k^*[\mathrm{target\_byte}] = \arg\max_i \mathrm{S_{cal}}[i]$
11: end for
12: return $k^*$
```

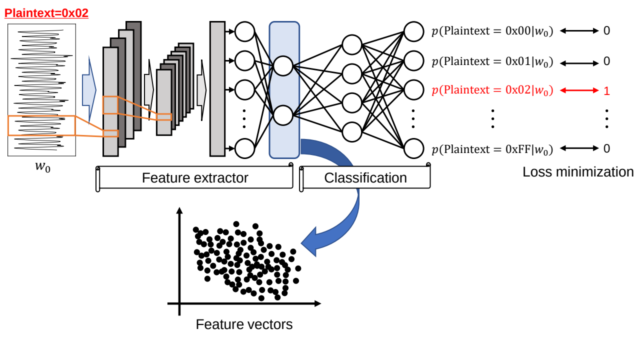

Plaintext=0x02
w_0
Feature extractor
Classification
p(Plaintext = 0x00|w_0) ⟷ 0
p(Plaintext = 0x01|w_0) ⟷ 0
p(Plaintext = 0x02|w_0) ⟷ 1
p(Plaintext = 0xFF|w_0) ⟷ 0
Loss minimization
Feature vectors

Fig. 3: Feature vectors extraction using convolutional neural network in CA-SCA

probability of plaintext given the waveforms w (it is illustrated as $p(\mathrm{Plaintext} = 0\mathrm{x}02|w_0)$ in Fig. 3). Next, feature maps are calculated using the trained feature extraction layer $\mathcal{M}_{\theta_{\mathrm{f}}}$ (line 4 in algorithm 3). The feature map corresponds to feature vectors and can be treated the same as the latent variables in the previous section. Finally, the target byte and key candidates are set, and the CH index is calculated for each. The adversary assumes that the key candidate with the highest CH index is the correct key.

4. Evaluation with software-implemented AES

4.1 Overview

This section describes the attack evaluation of CA-SCAs against software-implemented AES. Section 4.2 describes concatenated dataset which improve learning efficiency for DNN models when attacking against software-implemented AES with CA-SCA using AE, as explained in section 3.4, and CA-SCA using CNN, as described in section 3.5. Section 4.3 evaluates attacks against AES without SCA countermeasures. Section 4.4 evaluates attacks against the ASCAD database. Section 4.5 discusses these evaluations.

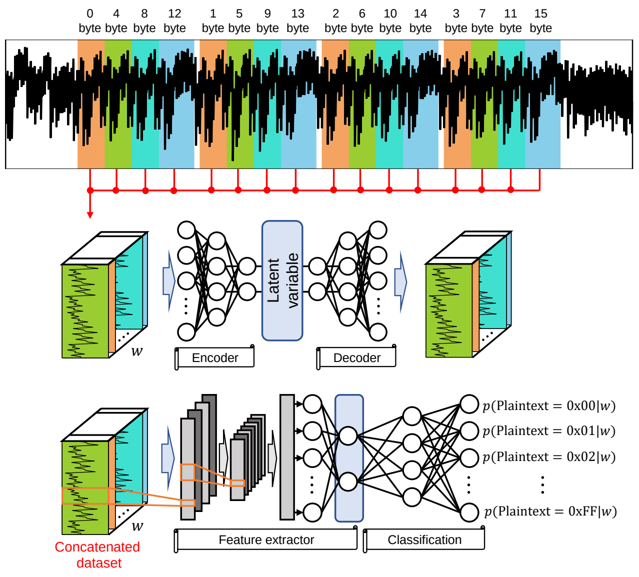


Fig. 4: Overview of how to improve learning efficiency at DNN by using a concatenated dataset.

4.2 Efficient training method of AE and CNN

As discussed in section 3.1, CA-SCA is advantageous in terms of computational cost because the number of trained models is significantly reduced compared to DDLA. In this section, we discuss another advantage of CA-SCA in terms of learning efficiency, i.e., the ability to train models with fewer waveforms.

Each byte is processed sequentially when encryption is processed on a microcontroller with a single core. In other words, the attacked SubByte is processed byte by byte. Therefore, SCA leakage corresponding to the 16 key bytes appears in 16 separate locations in the leakage waveform. The SubBytes processing details at these locations are generally the same. In the proposed method, the leakage waveforms of 16 bytes are concatenated, and AE and CNN are trained with these waveforms dataset as shown in Fig. 4. We called it a concatenated dataset. It is possible to obtain 16 waveforms from a single encryption as training data then the training efficiency is improved, and the number of training models can be reduced to one.

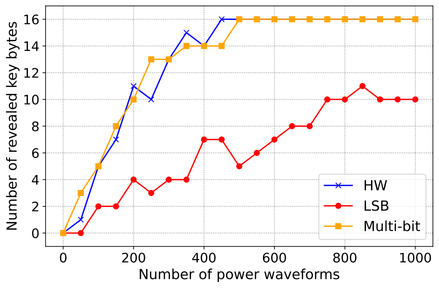
In CA-SCA with AE, the model is trained to minimize reconstruction error. On the other hand, in CA-SCA with CNN, the model is trained the plaintext of the byte of interest is output. The trained AE or CNN is input with the waveforms focused on the target byte, and feature vectors are extracted. Cluster analysis is then applied to reveal cryptographic keys.

4.3 Case study S-1: Software-implemented AES without SCA countermeasures

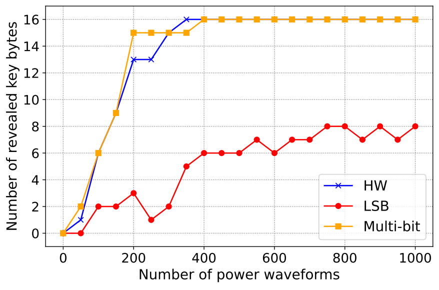
This section provides the results of attack evaluations of software-implemented AES without SCA countermeasures.

Table 1: Setup for CA-SCA in case study S-1

CA-SCA with AE	Network (Total params 106.07 KB)		
		Layer type	Output Shape
	Encoder	Input	(None, 70, 1)
		Conv1D (filters = 4, kernel_size = 7, strides = 7) + SeLU	(None, 10, 4)
		Flatten	(None, 40)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(64) + SeLU	(None, 64)
	Decoder	Full Connected(128) + SeLU	(None, 128)
		Full Connected(40) + SeLU	(None, 40)
		Reshape	(None, 10, 4)
		Conv1D Transpose (filters = 4, kernel_size = 7, strides = 7)	(None, 70, 4)
		Conv1D Transpose (filters = 1, kernel_size = 5, strides = 1)	(None, 70, 1)
	Other setting		
	Number of epochs: 50 Loss function: MeanSquaredError Optimizer: Adam(learning_rate=0.001) batch size: 256		
CA-SCA with CNN	Network (Total params 117.88 KB)		
		Layer type	Output Shape
	Feature extractor	Input	(None, 70, 1)
		Conv1D (filters = 4, kernel_size = 7, strides = 7) + SeLU	(None, 10, 4)
		Flatten	(None, 40)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(64) + SeLU	(None, 64)
	Classification	Full Connected(256) + Softmax	(None, 256)
	Other setting		
	Number of epochs: 50 Loss function: CrossEntropy Optimizer: Adam(learning_rate=0.001) batch size: 256		



(a) Results of CA-SCA w/ AE



(b) Results of CA-SCA w/ CNN

Fig. 5: Comparison results of the selected functions on the CA-SCAs

We evaluated six attack methods: CA-SCA with AE, CA-SCA with CNN, DDLA, MOR[16], MOC[16], and CPA. The setup for the experiments is described in detail below.

4.3.1 Details of waveforms

ChipWhisperer-Lite (CW1173), developed by new AE Technology, was used as the environment for waveform acquisition. CW303-XMEGA target, also developed by New AE Technology, was used as the target microcontroller board. This board is equipped with an 8-bit microcontroller, ATXmega128D4-AU. Power consumption during encryption operations was acquired using the A/D converter on CW1173, which has a sampling rate of 29 MS/s.

4.3.2 Setup for analysis

The number of waveforms used in the analysis was set from 50 to 1,000 at 50 intervals, and multiple runs were performed. Other DNN settings are shown in Table 1.

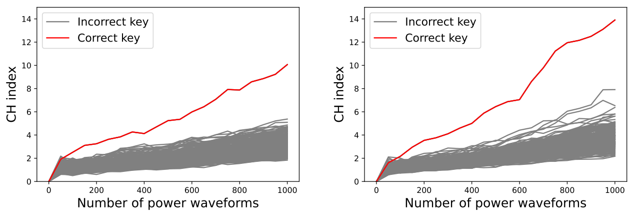
4.3.3 Experimental results

For the initial evaluation, attacks were conducted when the selected function for the calculation of the CH index was varied. The functions explained in section 3.3 were evaluated as follows.

- **HW labeling:** The CH index is calculated using Eq. (8).
- **LSB labeling:** The CH index is calculated using Eq. (10).
- **Multi-bit:** The CH index is calculated using Eq. (11).

The results of CA-SCA with AE and CA-SCA with CNN are shown in Fig. 5a and 5b, respectively. The horizontal axis shows the number of waveforms used in the analysis, and the vertical axis shows the number of revealed key bytes. All key bytes were revealed in about 500 waveforms in CA-SCA with AE which selected clustering with HW labeling or multi-bit. On the other hand, only 11 key bytes were revealed by clustering with LSBs, even with 1,000 waveforms. All key bytes were revealed in about 400 waveforms in CA-SCA with CNN which selected clustering with HW labeling or multi-bit. On the other hand, only eight key bytes were revealed by clustering with LSBs, even with 1,000 waveforms. These results indicate that the selected function for CA-SCA with AE and CNN is important. It is also shown that HW labeling, multibit, which focuses on 8 bits, is more efficient than LSB labeling, which focuses on 1 bit. However, as mentioned above, once the model has been trained, the selected function can be changed, and the attack can be conducted in many kinds. The computational cost for clustering is very small compared to that of DNN training.

For the next evaluation, we observed the value of the CH index during the attack, with emphasis on analyzing the 0th key byte. HW labeling was adopted as the selected function. The transitions of the CH index when CA-SCA with AE and CA-SCA with CNN are conducted are shown in Fig. 6a and 6b, respectively. The horizontal axis indicates the number of waveforms used in the analysis, and the vertical axis indicates the CH index. The red line shows the result of using the correct key, while the gray line shows the result of using the incorrect key. In both cases of CA-SCA with AE and CA-SCA with CNN, the CH-index values of the correct key and the other keys are separated after about 100 waveforms, indicating that the attack is successful. Comparing AE and CNN, the number of waveforms required to reveal all keys is about 500 and 400, and the CH-index graphs shown in Fig.



(a) CA-SCA w/ AE (b) CA-SCA w/ CNN

Fig. 6: Transition of CH index when attacking against software-implemented AES without SCA countermeasures. Target is 0th key byte.

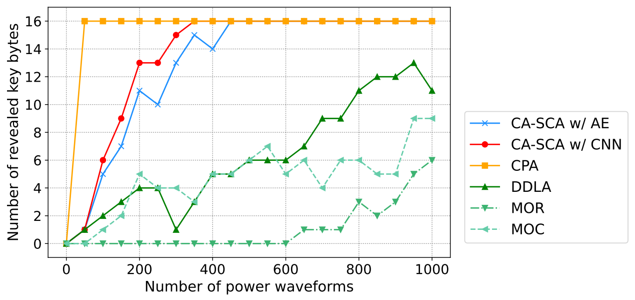


Fig. 7: Comparison with other non-profiled attacks when attacking against software-implemented AES without SCA countermeasures

6 are similar, indicating that there is no significant difference in attack performance when AES without countermeasures is the attack target.

As a final evaluation, we compared the results with other non-profiled attacks (CPA, DDLA, and DDLA variants (MOR, MOC)[16]). In CA-SCA, HW labeling was adopted as the selected function. The results of the evaluation are shown in Fig. 7, where the horizontal axis shows the number of waveforms used in the analysis, and the vertical axis shows the number of revealed key bytes. Compared to CA-SCA, which required around 400 waveforms to reveal all keys, and DDLA, which required 1000 waveforms to reveal only 13 bytes, conventional CPA was able to reveal all keys in 30 waveforms. As a result, CPA was the most powerful attack in software implementations without SCA countermeasures, where there is a strong correlation between the HW of intermediate value and the waveform. In other words, these experimental results suggest that the conventional CPA is better than deep learning when the correlation between waveform leakage and assumed model is clear. On the other hand, when comparing DDLA, including its variants and CA-SCA, CA-SCAs were more powerful, which could be due to the effect of the concatenated dataset method used in CA-SCA.

4.4 Case study S-2: ASCAD database

In this section, we provide the results of attack evaluations of the ASCAD database. We evaluated the following attack methods: CA-SCA with AE, CA-SCA with CNN, DDLA, MOR[16], MOC[16], and CPA. The setup for experiments is described below.

4.4.1 Details of waveforms

The ASCAD database is a public dataset that provides the electromagnetic (EM) emission waveforms during the operation of AES with a table re-computation masking countermeasure on AVR ATMega8515, which is an 8-bit microcontroller[†]. These waveforms were acquired using an oscilloscope with a sampling rate of 2 GS/s. The waveforms corresponding to the first round of AES processing are available in this dataset. In general, the evaluation is often limited to the SubBytes processing of the 2nd byte of the first round, but waveforms corresponding to SubBytes processing for all bytes are used in this study. This dataset has table re-computation masking countermeasure, but the mask value is fixed to 0 for the 0th and 1st byte. Note that masking is disabled for these two bytes.

4.4.2 Setup for analysis

The number of waveforms used in the analysis was set from 1,000 to 20,000 at 1,000 intervals, and multiple runs were performed. The waveforms corresponding to SubBytes other than the 2 bytes for which masking was disabled were used to train for training the DNN. Other DNN settings are shown in Table 2.

4.4.3 Experimental results

We observed the value of the CH index during the attack. In this evaluation, we focus on the analysis of the 2nd key byte. Multi-bit was adopted as the selected function. This is because it achieved the best results among HW labeling, LSB labeling, and Multi-bit. The transitions of the CH index when CA-SCA with AE and CA-SCA with CNN are conducted are shown in Fig. 8a and 8b, respectively. The horizontal axis indicates the number of waveforms used in the analysis, while the vertical axis indicates the CH index. The red line shows the result of using the correct key, while the gray line shows the result of using the incorrect key. In case of CA-SCA with CNN, the CH-index values of the correct key and the other keys are separated after about 12,000 waveforms, indicating that the attack is successful. On the other hand, in case of CA-SCA with AE, the CH index values calculated using the correct and incorrect keys were not separated. This indicates that supervised learning should be used to attack masking countermeasures.

We compared the results with other non-profiled attacks (CPA, DDLA, and DDLA variants (MOR, MOC)[16]). In CA-SCA, multi-bit was adopted as the selected function. The results of the evaluation are shown in Fig. 9, where the

[†]The ASCAD database is available at `https://github.com/ANSSI-FR/ASCAD`

Table 2: Setup for CA-SCA in case study S-2

CA-SCA with AE	Network (Total params 106.07 KB)		
		Layer type	Output Shape
	Encoder	Input	(None, 700, 1)
		Conv1D (filters = 4, kernel_size = 7, strides = 7) + SeLU	(None, 100, 4)
		Conv1D (filters = 8, kernel_size = 5, strides = 5) + SeLU	(None, 20, 8)
		Conv1D (filters = 16, kernel_size = 5, strides = 5) + SeLU	(None, 4, 16)
		Flatten	(None, 64)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(64) + SeLU	(None, 64)
	Decoder	Full Connected(64) + SeLU	(None, 64)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(64) + SeLU	(None, 64)
		Reshape	(None, 4, 16)
		Conv1D Transpose (filters = 16, kernel_size = 5, strides = 5)	(None, 20, 16)
		Conv1D Transpose (filters = 8, kernel_size = 5, strides = 5)	(None, 100, 8)
		Conv1D Transpose (filters = 4, kernel_size = 7, strides = 7)	(None, 700, 4)
		Conv1D Transpose (filters = 1, kernel_size = 5, strides = 1)	(None, 700, 1)
	Other setting		
	Number of epochs: 200 Loss function: MeanSquaredError Optimizer: Adam(learning_rate=0.001) batch size: 20000		
CA-SCA with CNN	Network (Total params: 294.09 KB)		
		Layer type	Output Shape
	Feature extractor	Input	(None, 700, 1)
		Conv1D (filters = 4, kernel_size = 7, strides = 7) + SeLU	(None, 100, 4)
		Conv1D (filters = 8, kernel_size = 5, strides = 5) + SeLU	(None, 20, 8)
		Conv1D (filters = 16, kernel_size = 5, strides = 5) + SeLU	(None, 4, 16)
		Flatten	(None, 64)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(64) + SeLU	(None, 64)
	Classification	Full Connected(256) + Softmax	(None, 256)
	Other setting		
	Number of epochs: 200 Loss function: CrossEntropy Optimizer: Adam(learning_rate=0.001) batch size: 20000		

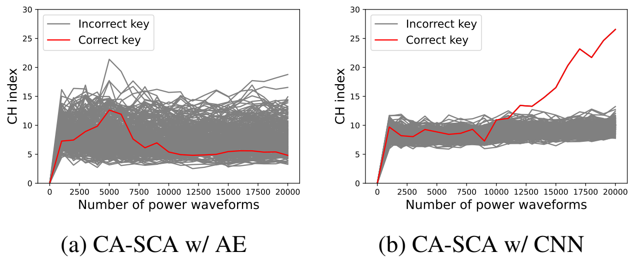


Fig. 8: Transition of CH index when attacking against ASCAD database

horizontal axis shows the number of waveforms used in the analysis, and the vertical axis shows the number of revealed key bytes. For CA-SCA using CNN, all key bytes were revealed in 12,000 waveforms. In contrast, DDLA, MOR, MOC, CPA, and CA-SCA with AE, did not reveal all key bytes even with 20,000 waveforms. The above results show that CA-SCA using CNN has the highest attack efficiency among the six methods.

4.5 Discussion

It is expected that the computation time to attack is greatly smaller than DDLA since CA-SCA trains only one DNN model. Thus, we measured and compared the computation time. We used a workstation equipped with a CPU: Intel Xeon Cold6226R (2.98GHz), DDR4 memory: 192GB, and a GPU: RTX-A5000 24GB to measure the computation time. The period of time measurement focused on parts of the DNN

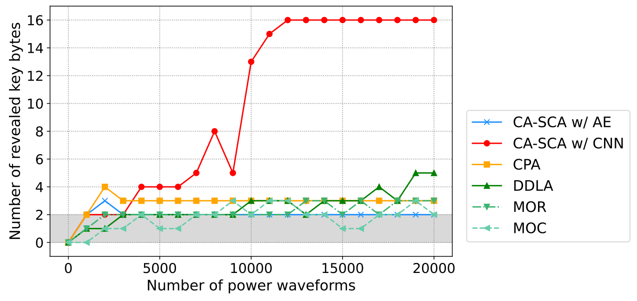
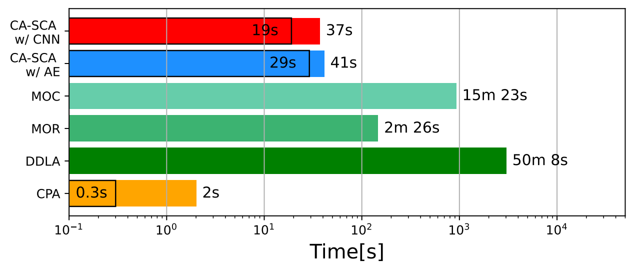


Fig. 9: Comparison with other non-profiled attacks when attacking against ASCAD database. Two of the 16 key bytes are not protected by SCA countermeasure using the masking method.

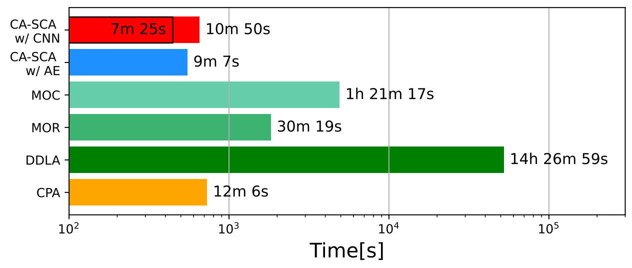
training, calculations of a correlation coefficient, and the CH index. Note that data loading and label calculation are not included in the period.

Figures 10a and 10b show the computation time required for the six attacks in case study S-1 and S-2, respectively. The hatched bars indicate the computation time when 1,000 waveforms were used in Fig. 10a. CPA assumes a correlation between intermediate values and waveforms and is very fast, taking only 2 seconds, because it only calculates the correlation coefficient. On the other hand, DDLA, which is SCA using deep learning techniques, is very slow, taking 50 minutes and 8 seconds. This is because 16×256 DNN models are trained to estimate 16 key bytes. The computation times of MOC and MOR, which are DDLA variants specifically designed to reduce computation time, are 15m 23s and 2m 26s, respectively. CA-SCAs using AE and CNN had computation times of 41 and 37 seconds, respectively. The computation time of CA-SCAs with AE and CNN was approximately 1/75 of the computation of DDLA. Also, the unhatched bars are the computation time when all key bytes are revealed. In case study S-1, the proposed CA-SCAs with CNN and AE were able to reveal all key bytes in much less time than DDLA. In addition, CA-SCAs, which require only one model to reveal all key bytes, require less computation time than MOR and MOC, which require 16 models. In case study S-2, in which attacks were evaluated using up to 20,000 waveforms shown in Fig. 10b, the proposed CA-SCA with CNN was able to reveal all key bytes in much less time than other attacks.

The CA-SCA with AE in case study S-2 did not reveal the protected keys. This is because AE is an unsupervised learning method that does not provide ground-truth labels for training. There is a correlation between the product of the two points, where the mask value and the masked value are processed, in the waveforms and the HW of the true intermediate value [23]. The model cannot learn this relationship because unsupervised learning does not provide a ground-truth label that depends on the true intermediate value. Therefore, CA-SCA with AE did not work for the ASCAD database.



(a) Case study S-1



(b) Case study S-2

Fig. 10: Comparison of computation time in case study S-1 and S-2. The long bars represent the attack time at the maximum number of waveforms available. The number of available waveforms is 1,000 and 20,000 in case study S-1 and S-2, respectively. The short bars surrounded by a black Box represent the time required for all key bytes to be revealed. The number of waveforms required to reveal all key bytes is shown in Table 5.

5. Evaluation with hardware-implemented AES

5.1 Overview

This section describes the attack evaluation of CA-SCAs against hardware-implemented AES. Section 5.2 describes how to adopt CA-SCA with CNN into hardware-implemented AES. Section 5.3 evaluates attacks against ASIC-implemented AES without SCA countermeasures. Section 5.4 evaluates attacks against ASIC-implemented AES with RSM countermeasures. Section 5.5 discusses these evaluations.

5.2 How to adopt CA-SCA with CNN into hardware-implemented AES

Hardware-implemented AES often runs byte processing in parallel, unlike software implementation. In this case, all bytes are processed in a single clock cycle. Therefore, it is difficult to simply apply the learning method used in section 3.5, because the ground-truth label (plaintext or ciphertext) cannot be defined.

We introduce multi-task learning (MTL) [24] to solve this problem. In MTL, a single learner handles tasks with different characteristics. This allows multiple tasks to share the same model, making the model more compact. An overview of CA-SCA using a CNN applied with MTL is shown in Fig. 11. In this model, the layers for feature extraction are common, and the layers for classification are branched into 16 sub networks. Each branched network for classification infers the ciphertext of the corresponding byte. This model structure allows 16 ciphertext labels to be given for a single waveform. Hereafter, we refer to CNN applying MTL as CNN-MTL. Note that this method also can reveal 16 byte keys by training a single model.

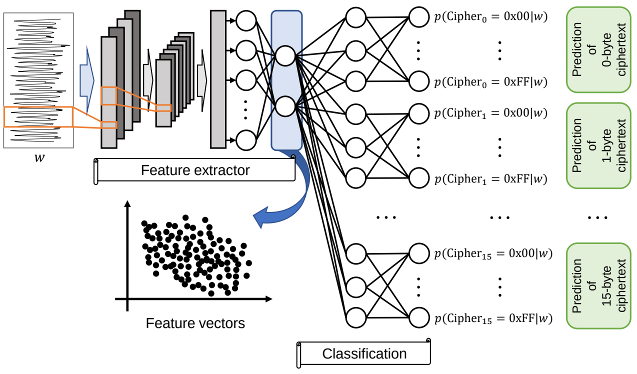

w
Feature extractor
Feature vectors
Classification
$p(\mathrm{Cipher}_0 = \mathrm{0x00}|w)$
$p(\mathrm{Cipher}_0 = \mathrm{0xFF}|w)$
$p(\mathrm{Cipher}_1 = \mathrm{0x00}|w)$
$p(\mathrm{Cipher}_1 = \mathrm{0xFF}|w)$
$p(\mathrm{Cipher}_{15} = \mathrm{0x00}|w)$
$p(\mathrm{Cipher}_{15} = \mathrm{0xFF}|w)$
Prediction of 0-byte ciphertext
Prediction of 1-byte ciphertext
Prediction of 15-byte ciphertext

Fig. 11: Overview of cluster-analysis-based side-channel attacks

To attack hardware-implemented AES, the HD model is typically used, but the computation of HD requires the use of different bytes of ciphertext, as shown in Eq. (12) when the target byte is other than the 0th, 4th, 8th, and 12th byte. CNN-MTL method is expected to successfully reveal all keys by focusing on the feature vectors which is created by considering all bytes of ciphertext.

5.3 Case study H-1: ASIC-implemented AES without SCA countermeasures

5.3.1 Target device and experimental setup

This section provides the results of attack evaluations of ASIC-implemented AES without SCA countermeasures. We evaluated six attack methods: CA-SCA with AE, CA-SCA with CNN-MTL, DDLA, MOR[16], MOC[16], and CPA.

The experimental setup for SCA evaluation is described below. The attack target is a prototype AES cryptographic circuit fabricated in 180nm CMOS process. SCA evaluation board is ZUIHO and its daughter board SASEBO-RII on which the ASIC is mounted[†]. Power consumption waveforms were acquired using an oscilloscope DSOX3104T developed by Keysight.

5.3.2 Setup for analysis

The number of waveforms used in the analysis was set from

[†]`https://www.risec.aist.go.jp/project/sasebo/`

Table 3: Setup for CA-SCA in case study H-1

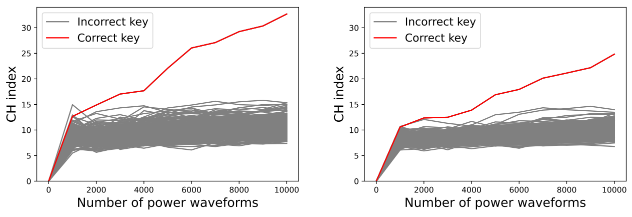
		Layer type	Output Shape
CA-SCA with AE	Network (Total params 981.66 KB)		
	Encoder	Input	(None, 40, 1)
		Flatten	(None, 40)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(128) + SeLU	(None, 128)
	Decoder	Full Connected(128) + SeLU	(None, 128)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(40) + SeLU	(None, 40)
		Reshape	(None, 40, 1)
	Other setting		
	Number of epochs: 100 Loss function: MeanSquaredError Optimizer: Adam(learning_rate=0.001) batch size: 1000		
CA-SCA with CNN-MTL	Network (Total params: 1.78 MB)		
		Layer type	Output Shape
	Feature extractor	Input	(None, 40, 1)
		Conv1D (filters = 16, kernel_size = 3, strides = 3) + BatchNorm + SeLU	(None, 13, 16)
		Conv1D (filters = 32, kernel_size = 3, strides = 3) + BatchNorm + SeLU	(None, 4, 32)
		Flatten	(None, 128)
	Classification	Full Connected(256) + SeLU	(None, 256)
		Full Connected(64) + SeLU	(None, 64)
		Full Connected(64) + SeLU	(None, 64)
		Full Connected(256) + Softmax	(None, 256)
	Other setting		
	Number of epochs: 100 Loss function: CrossEntropy Optimizer: Adam(learning_rate=0.001) batch size: 1000		

1,000 to 10,000 at 1,000 intervals, and multiple runs were performed. In the scenario in which the adversary knew the ciphertext, we chose the waveform at the 10th round of processing as the points of interest. Other DNN settings are shown in Table 3.

5.3.3 Experimental results

We observed the value of the CH index during the attack. In this evaluation, we focus on analyzing the 0th key byte. Multi-bit was adopted as the selected function. The transitions of the CH index when CA-SCA with AE and CA-SCA with CNN-MTL are shown in Fig. 12a and 12b, respectively. The horizontal axis indicates the number of waveforms used in the analysis, and the vertical axis indicates the CH index. The red line shows the result of using the correct key, while the gray line shows the result of using the incorrect key. In both cases of CA-SCA with AE and CA-SCA with CNN-MTL, the CH-index values of the correct key and the other keys are separated after about 2,000 waveforms, indicating that the attack is successful.

We compared the result with other non-profiled attacks (CPA, DDLA, and DDLA variants (MOR, MOC)[16]). In CA-SCA, multi-bit was adopted as the selected function. The results of the evaluation are shown in Fig. 13, where the horizontal axis shows the number of waveforms used in the analysis, and the vertical axis shows the number of revealed key bytes. For CPA, all key bytes were revealed in 5,000 waveforms, whereas CA-SCA with AE and CNN-MTL required 8,000 waveforms. Similar to the experimental results for unprotected software implementation described in section 4.3, the attack performance of conventional CPA is



(a) CA-SCA w/ AE (b) CA-SCA w/ CNN-MTL

Fig. 12: Transition of CH index when attacking against ASIC-implemented AES without SCA countermeasures

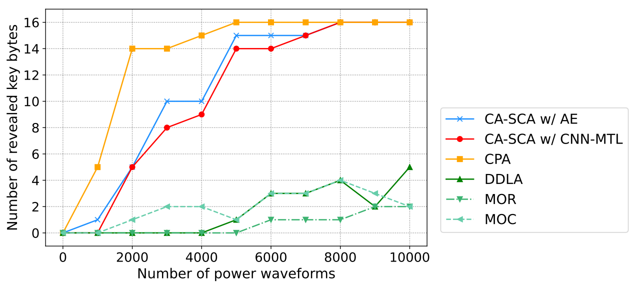


Fig. 13: Comparison with other non-profiled attacks when attacking against ASIC-implemented AES without SCA countermeasures

higher than that of methods using deep learning when there is a large correlation between the waveforms and the leakage model. In contrast, DDLA revealed only five key bytes even with 10,000 waveforms. MOR and MOC also revealed only two key bytes even with 10,000 waveforms.

5.4 Case study H-2: ASIC-implemented AES with RSM countermeasures

5.4.1 Target device and experimental setup

This section provides the results of attack evaluations of ASIC-implemented AES with RSM countermeasures. The RSM countermeasure can be implemented with small area penalty and provide some robustness against attacks using DPA, CPA, and zero-offset 2nd order DPA [25]. The RSM countermeasures uses 16 kinds of masked S-Boxes for SubBytes processing in AES. The masked S-Boxes are rotated by rounds of AES, and is connected to a different data register for each round. We evaluated six attack methods: CA-SCA with AE, CA-SCA with CNN-MTL, DDLA, MOR[16], MOC[16], and CPA. The experimental setup is the same as in section 5.3.

5.4.2 Setup for analysis

The number of waveforms used in the analysis was set from 20,000 to 200,000 at 20,000 intervals, and multiple runs were performed. In a scenario in which the adversary knew the ciphertext, we chose the waveform at the 10th round of processing as the points of interest. Other DNN settings are

Table 4: Setup for CA-SCA in case study H-2

	Network (Total params 981.66 KB)		
		Layer type	Output Shape
CA-SCA with AE	Encoder	Input	(None, 40, 1)
		Flatten	(None, 40)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(128) + SeLU	(None, 128)
	Decoder	Full Connected(64) + SeLU	(None, 64)
		Full Connected(128) + SeLU	(None, 128)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(256) + SeLU	(None, 256)
		Full Connected(40) + SeLU	(None, 40)
		Reshape	(None, 40, 1)
	Other setting		
	Number of epochs: 100 Loss function: MeanSquaredError Optimizer: Adam(learning_rate=0.001) batch size: 5000		

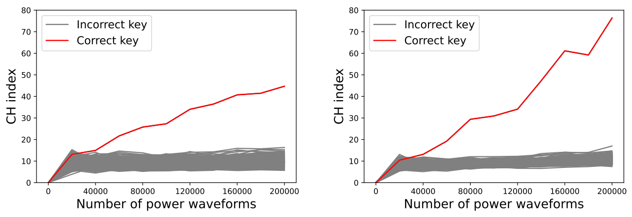
	Network (Total params: 1.78 MB)		
		Layer type	Output Shape
CA-SCA with CNN-MTL	Feature extractor	Input	(None, 40, 1)
		Conv1D (filters = 16, kernel_size = 3, strides = 3) + BatchNorm + SeLU	(None, 100, 4)
		Conv1D (filters = 32, kernel_size = 3, strides = 3) + BatchNorm + SeLU	(None, 100, 4)
		Flatten	(None, 64)
	Classification	Full Connected(256) + SeLU	(None, 256)
		Full Connected(64) + SeLU	(None, 64)
		Full Connected(64) + SeLU	(None, 64)
		Full Connected(256) + Softmax	(None, 256)
	Other setting		
	Number of epochs: 100 Loss function: CrossEntropy Optimizer: Adam(learning_rate=0.001) batch size: 5000		

shown in Table 4.

5.4.3 Experimental results

We observed the value of the CH index during the attack. In this evaluation, we focus on analyzing the 0th key byte. Multi-bit was adopted as the selected function. The transitions of the CH index when CA-SCA with AE and CA-SCA with CNN-MTL are shown in Fig. 14a and 14b, respectively. The horizontal axis indicates the number of waveforms used in the analysis, and the vertical axis indicates the CH index. The red line shows the result of using the correct key, while the gray line shows the result of using the incorrect key. In both cases, the CH-index values of the correct key and the other keys are separated after about 30,000 waveforms, indicating that the attack is successful. This value is increased from 2,000 in the results of AES without SCA countermeasures as shown in section 5.3.3, indicating that more waveforms are required to reveal the key against RSM countermeasure.

We compared the result with non-profiled attacks (CPA, DDLA, and DDLA variants (MOR, MOC)[16]). For CA-SCA, multi-bit was adopted as the selected function. The evaluated results are shown in Fig. 15, where the horizontal axis shows the number of waveforms used in the analysis, and the vertical axis shows the number of revealed key bytes. For CA-SCA using AE and CA-SCA using CNN-MTL, all key bytes were revealed in 140,000 and 80,000 waveforms, respectively. In contrast, DDLA, MOR, MOC, and CPA did not reveal all key bytes even with 200,000 waveforms. The above results show that CA-SCA using CNN-MTL has the highest attack efficiency of the six methods.



(a) CA-SCA w/ AE (b) CA-SCA w/ CNN-MTL

Fig. 14: Transition of CH index when attacking against ASIC-implemented AES with RSM

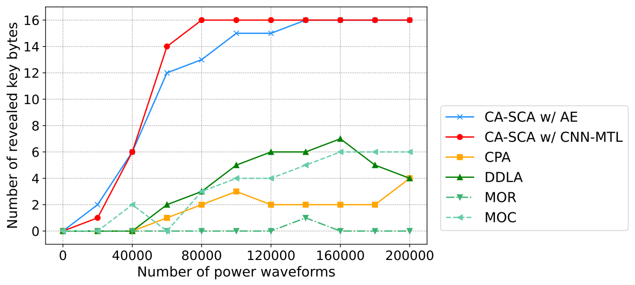
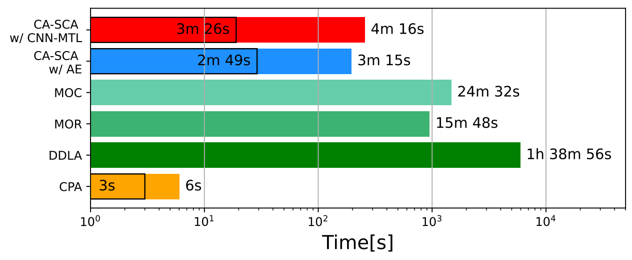


Fig. 15: Comparison with other non-profiled attacks when attacking against ASIC-implemented AES with RSM countermeasures

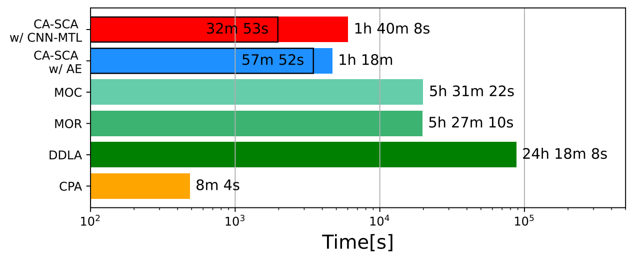
5.5 Discussion

The computation times for the six methods were measured on the same workstation and measurement conditions as described in section 4.5. Figures 16a and 16b show the computation time required for the six attacks in case study H-1 and H-2, respectively. The hatched bars in Fig. 16a indicate the computation time when 10,000 waveforms were used. CPA assumes a correlation between intermediate values and waveforms and is very fast, with a time of only 6 seconds, because it only calculates the correlation coefficient. On the other hand, DDLA, which is SCA using deep learning techniques, is very slow, with a time of 1 hours 38 minutes and 56 seconds. The computation times of MOC and MOR, which are DDLA variants specifically designed to reduce computation time, are 24m 32s and 15m 48s, respectively. CA-SCAs using AE and CNN-MTL had computation times of 3 minutes 15 seconds and 4 minutes 16 seconds, respectively. The computation times of CA-SCAs with AE and CNN-MTL were approximately 1/30 and 1/22 of those for DDLA, respectively. We also compared the time required to reveal all keys. Also, the unhatched bars are the computation time when all key bytes are revealed. The proposed methods were able to reveal all key bytes in much less time than DDLA in both case studies. In addition, CA-SCAs, which require only one model to reveal all key bytes, require less computation time than MOR and MOC, which require 16 models.

While CA-SCA with AE could not reveal all keys in



(a) Case study H-1



(b) Case study H-2

Fig. 16: Comparison of computation time in case study H-1 and H-2. The long bars represent the attack time at the maximum number of waveforms available. The number of available waveforms is 10,000 and 200,000 in case study H-1 and H-2, respectively. The short bars surrounded by a black Box represent the time required for all key bytes to be revealed. The number of waveforms required to reveal all key bytes is shown in Table 6.

the case study S-2, the CA-SCA with AE in case study H-2 successfully revealed all the keys. This is because the mask value caused a first-order SCA leakage. Moradi et al. mentioned that there is a vulnerability in the mask value of RSM [26]. Additionally, Fukuda et al. showed that the CNN learns the correlation between the HW of the upper 4-bit of the register transition and the waveforms [27]. This implies that first-order SCA leakage occurs at some bit positions. However, note that CA-SCA with AE may not be able to attack when mask values in RSM are applied such that first-order SCA leakage does not occur [28].

6. Comparison with other non-profiled attack methods

CA-SCAs can target any implementation that is attackable by DDLA. In this section, we summarized characteristics of proposed method in comparison with other non-profiled attacks such as DDLA and CPA. We discuss the following three main points: (1) optimization of hyperparameters in CA-SCA and DDLA, (2) the availability of the concatenated dataset, and (3) attack conditions required for higher-order CPA.

6.1 Optimization of hyperparameters in CA-SCA and DDLA

DDLA aims to achieve a difference in accuracy between the correct key models and the incorrect key model. For this purpose, it is necessary to set hyperparameters such as appropriate NN and regularization. On the other hand, CA-SCAs aim to improve the accuracy of the model as much as possible. For this purpose, the adversary only needs to prepare a large NN. The size of each NN is larger in CA-SCA than in DDLA from the above. Therefore, the computation time of one model for CA-SCAs may be longer than that for DDLA when concatenated datasets cannot be utilized. However, the computation time of the proposed method is shorter than that of DDLA and its variants because the number of models can be reduced by 1/4096 for DDLA and 1/16 for its variants at present.

6.2 Availability of the concatenated dataset

The concatenated datasets are an important option to increase the efficiency of CA-SCAs. This technique provides a larger dataset for CA-SCAs than the number of acquired traces. It is not available for DDLA because the adversary needs to set a label-set for each key candidate. This option allows CA-SCA to collect training data more efficiently than DDLA. Please note that CA-SCAs work even without this option, as shown in section 5.

To use the concatenated dataset option, the leakage waveforms of the target must satisfy the following requirements.

- The SW implementation is processed sequentially by the cryptographic round function, or the small-area HW implementation is processed with a few SubBytes circuits.
- The leakage waveforms cropped as PoI contain both areas where the mask and masked values are processed.
- The leakage waveforms cropped as PoI are known in which byte the SubByte is processed.

Firstly, the ASIC implementation used in case studies H-1 and H-2 does not apply to concatenated datasets. This is because the 16 SubBytes processing is done in parallel. Secondly, for example, in the RSM implementation used in [12], the mask value is not processed during SubBytes processing, which is set as PoI, and the mask value is processed before the SubBytes processing as pre-processing. Therefore, the adversary needs to include pre-processing and SubBytes processing in the PoI, and the concatenated datasets are not applicable now. Finally, it is also possible that the concatenated datasets do not work in an attack against shuffling countermeasures. These issues could be improved by inputting the section where the mask and shuffle values are processed independently into the DNN model, which will be worked on in the future.

6.3 Attack conditions required for higher-order CPA

If the high-order attacks against masked implementation [23] is possible, the computation time is shorter than that of the proposed method. However, this attack requires a precise selection of both PoIs where the mask and masked value are processed. The proposed method allows the adversary to reveal the cryptographic keys without knowing these points.

7. Conclusion

We examined new non-profiled side-channel attacks (SCAs) using deep learning techniques. We proposed cluster-analysis-based side-channel attacks (CA-SCAs) where correct key is revealed by the score of cluster analysis on feature vectors extracted from waveforms by using deep neural networks (DNNs). We used auto-encoder (AE) and convolutional neural networks (CNNs) as DNNs. Our method requires only one trained DNN model to reveal all key bytes, whereas DDLA requires 256×16 trained DNN models. Therefore, the computation time for model training is very short. Another advantage of our method is that once the DNN model is trained, it is not necessary to re-train the DNN model when trying attacks with different labels. We provided the experimental results of attacks against software and hardware implemented AES to demonstrate the effectiveness of the proposed method.

We evaluated attacks against the software-implemented AES without SCA countermeasures and the ASCAD database in case studies S-1 and S-2. We used a method that effectively trains DNNs by utilizing byte-by-byte sequential processing called concatenated dataset. In both case studies, we showed that all key bytes are revealed using fewer waveforms than DDLA. We also showed that the computation time for our attack is reduced compared to DDLA. Table 5 summarizes the number of waveforms required for the attack.

We also evaluated attacks against the ASIC-implemented AES without SCA countermeasures and the ASIC-implemented AES with RSM countermeasures in case studies H-1 and H-2. We introduced multi-task learning (MTL) to enhance attacks against hardware-implemented AES in CA-SCA with CNN. In both case studies, we showed that all key bytes are revealed using fewer waveforms than DDLA. We also showed that the computation time for our attack is reduced compared to DDLA. Table 6 summarizes the number of waveforms required for the attack.

We confirmed the effectiveness of the proposed method through the above four case studies. In our future work, we plan to study leakage assessment, e.g., test vector leakage assessment [29], using CA-SCAs.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP22H03593 and Grant-in-Aid for JSPS fellows, Grant Number JP24KJ2150.

References

[1] S. Chari, J.R. Rao, and P. Rohatgi, “Template attacks,” International Workshop on Cryptographic Hardware and Embedded Systems, pp.13–28, Springer, 2002.

[2] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” Annual international cryptology conference, pp.388–397, Springer, 1999.

[3] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” Cryptographic Hardware and Embedded Systems - CHES 2004, ed. M. Joye and J.J. Quisquater, Berlin, Heidelberg, pp.16–29, Springer Berlin Heidelberg, 2004.

[4] H. Maghrebi, T. Portigliatti, and E. Prouff, “Breaking cryptographic implementations using deep learning techniques,” Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings, ed. C. Carlet, M.A. Hasan, and V. Saraswat, Lecture Notes in Computer Science, vol.10076, pp.3–26, Springer, 2016.

[5] E. Cagli, C. Dumas, and E. Prouff, “Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing,” Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings, ed. W. Fischer and N. Homma, Lecture Notes in Computer Science, vol.10529, pp.45–68, Springer, 2017.

[6] 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. Cryptogr. Hardw. Embed. Syst., vol.2019, no.1, pp.209–237, 2019.

[7] T. Kubota, K. Yoshida, M. Shiozaki, and T. Fujino, “Deep learning side-channel attack against hardware implementations of AES,” 2019 22nd Euromicro Conference on Digital System Design (DSD), pp.261–268, 2019.

[8] A. Ito, R. Ueno, and N. Homma, “Perceived information revisited new metrics to evaluate success rate of side-channel attacks,” IACR Trans. Cryptogr. Hardw. Embed. Syst., vol.2022, no.4, pp.228–254, 2022.

[9] B. Timon, “Non-profiled deep learning-based side-channel attacks with sensitivity analysis,” IACR Transactions on Cryptographic Hardware and Embedded Systems, vol.2019, no.2, p.107–131, Feb. 2019.

[10] E. Prouff, R. Strullu, R. Benadjila, E. Cagli, and C. Dumas, “Study of deep learning techniques for side-channel analysis and introduction to ASCAD database,” IACR Cryptol. ePrint Arch., p.53, 2018.

[11] K. Kuroda, Y. Fukuda, K. Yoshida, and T. Fujino, “Practical aspects on non-profiled deep-learning side-channel attacks against AES software implementation with two types of masking countermeasures including RSM,” Proceedings of the 5th Workshop on Attacks and Solutions in Hardware Security, ASHES ’21, New York, NY, USA, p.29–40, Association for Computing Machinery, 2021.

[12] 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,” Journal of Cryptographic Engineering, pp.1–16, 2023.

[13] 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,” 2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020, pp.614–617, IEEE, 2020.

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

[15] V.P. Hoang, N.T. Do, and V.S. Doan, “Efficient non-profiled side channel attack using multi-output classification neural network,” IEEE Embedded Systems Letters, pp.1–1, 2022.

[16] N.T. Do, P.C. Le, V.P. Hoang, V.S. Doan, H.G. Nguyen, and C.K. Pham, “Mo-dlsca: Deep learning based non-profiled side channel analysis using multi-output neural networks,” 2022 International Conference on Advanced Technologies for Communications (ATC), pp.245–250, 2022.

[17] N.T. Do, V.P. Hoang, V.S. Doan, and C.K. Pham, “On the performance of non-profiled side channel attacks based on deep learning

Table 5: Summary of the number of waveforms required to reveal all key bytes in attacking against software implementation

Method	CPA	DDLA	MOR	MOC	CA-SCA with AE	CA-SCA with CNN
Case study S-1 AES w/o SCA countermeasures	30	1,000 (13 key bytes)	1,000 (6 key bytes)	1,000 (9 key bytes)	450	350
Case study S-2 ASCAD database excl. 2 bytes w/o masking	20,000 (1 key bytes were revealed)	20,000 (3 key bytes were revealed)	20,000 (1 key bytes were revealed)	20,000 (1 key bytes were revealed)	20,000 (0 key bytes were revealed)	12,000

Table 6: Summary of the number of waveforms required to reveal all key bytes in attacking against hardware implementation

Method	CPA	DDLA	MOR	MOC	CA-SCA with AE	CA-SCA with CNN-MTL
Case study H-1 AES w/o SCA countermeasures	4,500	10,000 (5 key bytes were revealed)	10,000 (2 key bytes were revealed)	10,000 (4 key bytes were revealed)	8,000	8,000
Case study H-2 AES w/ RSM countermeasures	200,000 (4 key bytes were revealed)	200,000 (7 key bytes were revealed)	200,000 (1 key bytes were revealed)	200,000 (6 key bytes were revealed)	140,000	80,000

techniques," IET Information Security, vol.17, no.3, pp.377–393, 2023.

[18] L. Wu, G. Perin, and S. Picek, "Hiding in plain sight: Non-profiling deep learning-based side-channel analysis with plain-text/ciphertext." Cryptology ePrint Archive, Paper 2023/209, 2023. `https://eprint.iacr.org/2023/209`.

[19] Y. Fukuda, K. Yoshida, and T. Fujino, "Incorporating cluster analysis of feature vectors for non-profiled deep-learning-based side-channel attacks," Applied Cryptography and Network Security Workshops, ed. M. Andreoni, Cham, pp.84–101, Springer Nature Switzerland, 2024.

[20] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," Communications in Statistics-theory and Methods, vol.3, no.1, pp.1–27, 1974.

[21] G.E. Hinton and R.R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol.313, no.5786, pp.504–507, 2006.

[22] K. Imafuku, S. Kawamura, H. Nozaki, J. Sakamoto, and S. Osuka, "Non-profiled deep learning-based side-channel analysis with only one network training," IEEE Access, vol.11, pp.83221–83231, 2023.

[23] T.S. Messerges, "Using second-order power analysis to attack DPA resistant software," Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings, ed. Ç.K. Koç and C. Paar, Lecture Notes in Computer Science, vol.1965, pp.238–251, Springer, 2000.

[24] R. Caruana, "Multitask learning," Mach. Learn., vol.28, no.1, pp.41–75, 1997.

[25] J. Waddle and D. Wagner, "Towards efficient second-order power analysis," Cryptographic Hardware and Embedded Systems - CHES 2004, ed. M. Joye and J.J. Quisquater, Berlin, Heidelberg, pp.1–15, Springer Berlin Heidelberg, 2004.

[26] A. Moradi, S. Guilley, and A. Heuser, "Detecting hidden leakages," Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings, ed. I. Boureanu, P. Owesarski, and S. Vaudenay, Lecture Notes in Computer Science, vol.8479, pp.324–342, Springer, 2014.

[27] Y. Fukuda, K. Yoshida, H. Hashimoto, K. Kuroda, and T. Fujino, "Profiling deep learning side-channel attacks using multi-label against AES circuits with RSM countermeasure," IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol.106, no.3, pp.294–305, 2023.

[28] N. Veshchikov and S. Guilley, "Implementation flaws in the masking scheme of DPA contest v4," IET Inf. Secur., vol.11, no.6, pp.356–362, 2017.

[29] T. Schneider and A. Moradi, "Leakage assessment methodology - A clear roadmap for side-channel evaluations," Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, ed. T. Güneysu and H. Handschuh, Lecture Notes in Computer Science, vol.9293, pp.495–513, Springer, 2015.

Yuta Fukuda Yuta Fukuda received his B.E. and M.E. in electronic engineering from Ritsumeikan University in 2020 and 2022. He is currently a doctoral student at the Graduate School of Science and Technology, Ritsumeikan University. His research interests include machine learning and hardware security. He is a member of IEICE, IEEE.

Kota Yoshida Kota Yoshida received his B.E., M.E., and Ph.D. in engineering from Ritsumeikan University in 2017, 2019, and 2022. He is currently an assistant professor in Department of Electronic and Computer Engineering at Ritsumeikan University. His research interests include machine learning and hardware security. He is a member of IEICE, IEEE, JSAI.

Takeshi Fujino Takeshi Fujino was born in Osaka, Japan, on March 17, 1962. He received his B.E. and M.E., and Ph.D. in electronic engineering from Kyoto University, Kyoto, Japan, in 1984, 1986, and 1994. He joined the LSI Research and Development center, Mitsubishi Electric Corp. in 1986. Since then, he had been engaged in the development of micro-fabrication processes, such as electron beam lithography, and embedded DRAM circuit design. He has been a professor at Ritsumeikan University since 2003. His research interests include hardware security such as side-channel attacks and physically unclonable functions. He is a member of IEICE, IPSJ, IEEE.