Improve the Performance of Semi-Supervised Side-channel Analysis Using HWFilter Method

Hong Zhang[1,2], Lang Li[1,2*] and Di Li[1,2]
[1] College of Computer Science and Technology, Hengyang Normal University Hengyang 421002, China
[2] Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang, 421002 China
[e-mail: zhng572@gmail.com; lilang911@126.com; lidi9007@163.com]
[*]Corresponding author: Lang Li

Received May 7, 2023; revised December 18, 2023; accepted February 26, 2024; published March 31, 2024

Abstract

Side-channel analysis (SCA) is a cryptanalytic technique that exploits physical leakages, such as power consumption or electromagnetic emanations, from cryptographic devices to extract secret keys used in cryptographic algorithms. Recent studies have shown that training SCA models with semi-supervised learning can effectively overcome the problem of few labeled power traces. However, the process of training SCA models using semi-supervised learning generates many pseudo-labels. The performance of the SCA model can be reduced by some of these pseudo-labels. To solve this issue, we propose the HWFilter method to improve semi-supervised SCA. This method uses a Hamming Weight Pseudo-label Filter (HWPF) to filter the pseudo-labels generated by the semi-supervised SCA model, which enhances the model's performance. Furthermore, we introduce a normal distribution method for constructing the HWPF. In the normal distribution method, the Hamming weights (HWs) of power traces can be obtained from the normal distribution of power points. These HWs are filtered and combined into a HWPF. The HWFilter was tested using the ASCADv1 database and the AES_HD dataset. The experimental results demonstrate that the HWFilter method can significantly enhance the performance of semi-supervised SCA models. In the ASCADv1 database, the model with HWFilter requires only 33 power traces to recover the key. In the AES_HD dataset, the model with HWFilter outperforms the current best semi-supervised SCA model by 12%.

Keywords: Side-channel analysis, Semi-supervised learning, Hamming weight, Pseudo-label filter, Normal distribution.

1. Introduction

Deep Learning-based Side-Channel Analysis(DL-SCA) has now become a major research direction in Side-Channel Analysis(SCA) field. At present, a large number of studies on the combination of deep learning and SCA have appeared [1]. These studies can be classified into three types: DL-SCA with supervised learning, DL-SCA with unsupervised learning, and DL-SCA with semi-supervised learning. DL-SCA with supervised learning is well-suited for scenarios in which a substantial number of labeled power traces are available [2-4]. However, when the number of labeled power traces is limited, DL-SCA with supervised learning is susceptible to overfitting. This may result in the inability to recover the key. DL-SCA with unsupervised learning is suitable for scenarios where the key is unknown and a substantial volume of unlabeled power traces are available [5-7]. Although this type of DL-SCA exhibits significant advantages over traditional non-profiling SCAs such as DPA [8] and CPA [9], it still presents certain gaps when compared to DL-SCA with supervised learning. DL-SCA with semi-supervised learning overcomes the limitations of the previous two types of DL-SCA, enabling key recovery even when labeled power traces are scarce.

Semi-supervised learning is a machine learning method that leverages a small amount of labeled data and a large amount of unlabeled data for training [10-12]. It allows the SCA model to extract valuable information from the unlabeled data, thereby enhancing the performance of the SCA model. Semi-supervised learning is proven to be more powerful than supervised learning when labeled data is scarce [13]. Research by Picek S et al. suggests that exploring SCA with semi-supervised learning can improve the performance of SCA models [14]. These studies demonstrate that semi-supervised learning can play a significant role in SCA. SCA with semi-supervised learning is based on a hypothesis that might occur in real situations. Picek S et al. first proposed this hypothesis and described the scenario in which it exists [14]. Under this hypothesis, the attacker has many constraints during the profiling phase, but only a few constraints during the attack phase. This allows the attacker to obtain only a small number of labeled power traces during the profiling phase. If there are not enough labeled power traces, it is difficult for the attacker to obtain a SCA model with better performance. Based on this hypothesis, The main challenge of SCA with semi-supervised learning is how to use very few labeled power traces to improve the possibility of recovering keys. Some researchers have employed techniques such as self-training and graph-based learning to address this challenge [15-17]. Biao Liu et al. proposed an attack algorithm based on collaborative learning. This algorithm has been able to improve accuracy by about 20% [18]. These studies show that SCA with semi-supervised learning can successfully recover keys even with a limited number of labeled power traces.

Currently, there are several challenges in SCA with semi-supervised. One of the challenges is how to filter out pseudo-labels generated by the SCA model. These pseudo-labels encompass both correct and incorrect labels. Correct labels can enhance the model's performance, while incorrect ones can degrade it, potentially to the extent that the model fails to recover the key. A common solution is to add a confidence level during the training phase [19-21]. This solution is only applicable to models with two categories. Confidence level is unreliable for SCA models with 256 categories. Developing a method to effectively filter out the pseudo-labels generated by SCA models is a significant challenge for researchers. In this study, we propose a HWFilter method to filter out these pseudo-labels. Our experimental results indicate that the HWFilter method can filter out many incorrect pseudo-labels, thereby enhancing the performance of SCA model. Importantly, in scenarios with a limited number of labeled power traces, the models trained using the HWFilter method outperform those trained

with supervised learning methods. Our contributions can be summarized as follows:

(1) We propose a HWFilter method, which is designed to enhance the performance of semi-supervised SCA models. This method uses a Hamming Weight Pseudo-label Filter(HWPF) to filter out the pseudo-labels generated by the semi-supervised SCA model, thereby reducing the incorrect pseudo-labels involved in model training.

(2) We propose a normal distribution method for constructing HWPFs. The method can compute the HWs for each power consumption trace from the normal distribution of power consumption points. These HWs are formed into a HWPF for filtering pseudo labels.

(3) We used the ASCADv1 database and the AES_HD dataset to test the HWFilter method. The experimental results show that the performance of the semi-supervised SCA model trained using the HWFilter method is higher than the performance of other SCA models.

The structure of the remaining parts of this paper is as follows. Section 2 describes the process of SCA with semi-supervised learning. Section 3 discusses the normal distribution method, HWFilter method and the design of semi-supervised SCA models. Section 4 introduces the ASCADv1 database, AES_HD dataset and the preprocessing process of the dataset, followed by a description of the training process of the SCA model. In section 5, we first present the evaluation metrics and results of the experiment. Then, we compared the results of our experiment with other studies. Section 6 summarizes the contributions of this paper.

2. Semi-supervised learning and SCA

The basic idea of semi-supervised learning is building a learner to label unlabeled samples using model assumptions on the data distribution. Semi-supervised learning is a process in which a semi-supervised model is trained using labeled and unlabeled data sets. In the process, the labeled dataset consists of multiple samples, each paired with a corresponding label. The unlabeled dataset contains samples that lack corresponding labels. The process of semi-supervised learning encompasses three phases. In the first phase, a semi-supervised model is pre-trained using a labeled dataset. After several pre-training, this model can generate pseudo-labels. In the second phase, this model generates pseudo-labels for the unlabeled dataset. These pseudo-labels are combined with samples from the unlabeled dataset to form a pseudo-labeled dataset. This pseudo-labeled dataset and the labeled dataset are combined into a new training set. In the third phase, this new dataset is used to train a pre-trained semi-supervised model. The second phase and the third phase are iteratively executed until the pre-trained semi-supervised model reaches a predetermined number of training epochs. The process of semi-supervised learning is shown in **Fig. 1**. The semi-supervised model in **Fig. 1** is a neural network model used to perform classification or regression tasks. The labeled dataset and the unlabeled dataset in **Fig. 1** are the train set of this model. The labeled dataset contains samples and corresponding labels, while the unlabeled dataset only contains samples without corresponding labels. We can distinguish between labeled and unlabeled datasets by checking whether there are labels corresponding to samples in the dataset.

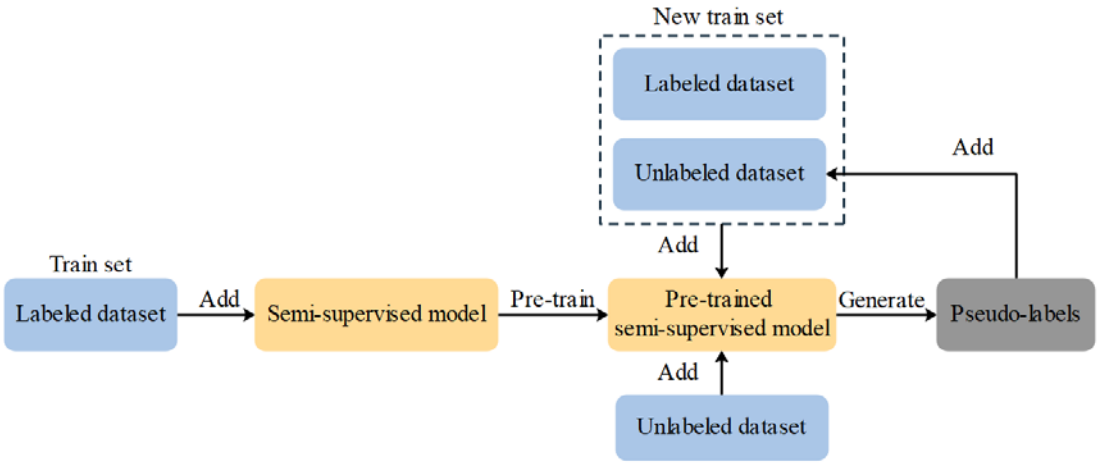

Fig. 1. The process of semi-supervised learning

Semi-supervised learning includes self-training and graph-based learning, among others. Self-training involves iteratively labeling unlabeled data using the model's predictions, while graph-based learning constructs a graph structure to exploit the geometric relationship between labeled and unlabeled data. The research results of Stjepan et al. indicate that the self-training method is most advantageous for SCA [14]. The self-training process comprises two phases: the profiling and the attack phase. In the profiling phase, the attacker can collect power traces $\boldsymbol{T^L}$ and intermediate values. In addition, an attacker can also collect a large number of power traces $\boldsymbol{D^U}$ that do not have corresponding intermediate values. These two types of power traces are used to create two datasets: labeled dataset $\boldsymbol{D^L}$and unlabeled dataset $\boldsymbol{D^U}$. First, the attacker uses $\boldsymbol{D^L}$ to pre-train an SCA model. This endows the model with the capability to predict intermediate values. Then, the attacker continues to train the model on $\boldsymbol{D^L}$ and $\boldsymbol{D^U}$. When $\boldsymbol{D^U}$ is used for training, the model generates some pseudo-labels that correspond one-to-one with the power traces within $\boldsymbol{D^U}$. Each pseudo-label is associated with a corresponding confidence score. The attacker can define a confidence interval to filter these pseudo-labels. When a confidence score falls within this interval, the pseudo-label corresponding to this confidence score is added to $\boldsymbol{D^U}$. $\boldsymbol{D^L}$ and $\boldsymbol{D^U}$ with pseudo-labels are used to continue training the model. The model is trained repeatedly until it achieves the expected performance. In the attack phase, the attacker uses the trained model to predict intermediate values on the test set. Then the attacker uses the intermediate values and Guess Entropy (GE) [22] to recover the key. The process of SCA using the self-training method is shown in **Fig. 2**.

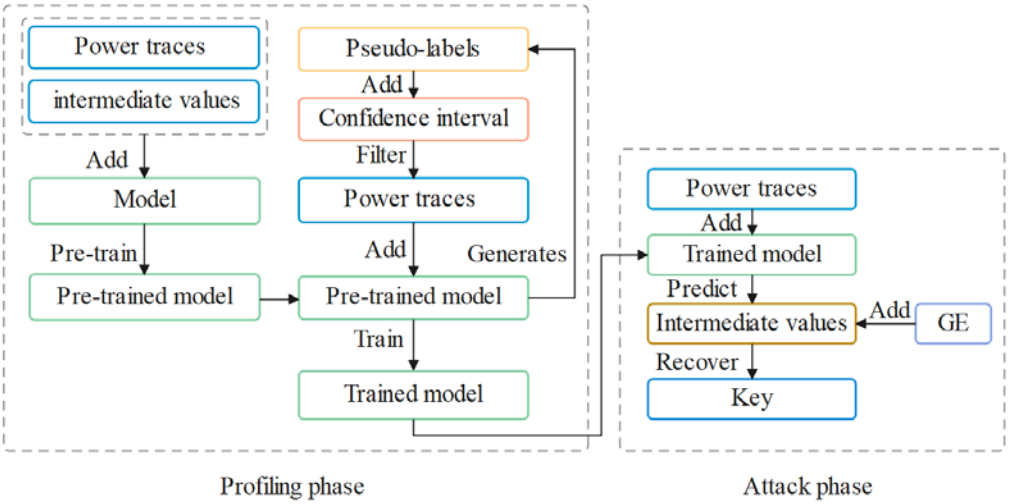

Fig. 2. The process of using the self-training method in SCA.

In the profiling phase of **Fig. 2**, pseudo-labels contain correct labels and incorrect labels. Correct labels can speed up the convergence of the model and improve the performance of the model. Incorrect labels can hinder the training of the model, which leads to the model not being able to recover the key in the attack phase. Therefore, filtering these pseudo-labels is important to recover the key. In the profiling phase, confidence intervals are used to filter the pseudo-labels. Correct labels correspond to a higher confidence score and incorrect labels correspond to a lower confidence score. Determining whether the confidence score falls within the confidence interval can distinguish between correct and incorrect labels. However, some incorrect labels correspond to a higher confidence score, which makes the confidence interval unable to discriminate these incorrect labels. Adding a filter to further filter these pseudo-labels can solve this problem.

3. HWFilter method

In this section, we present a HWFilter method that can effectively improve the performance of SCA models. This method consists of two parts. The first part is about how to construct the HWPF. The second part is about how to use the HWPF. In Section 3.1, we propose a normal distribution method to construct the HWPF. In Section 3.2, we delineate the detailed procedure for employing the HWFilter method. Section 3.3 describes the structural design of the SCA model in detail.

3.1 Normal distribution method

The normal distribution method is a statistical method that utilizes the characteristics of power points following a normal distribution to construct a HWPF. This method is inspired by template attacks [23]. The normal distribution method consists of three steps:

Step 1. First, we obtain N power traces from the raw power, each power trace contains K points. Among them, $M(M < N)$ power traces have M intermediate values. The M power traces T_M and their corresponding intermediate values Z_M form a set $T^L = \{(T_i, Z_i)|0 \leq i < M\}$. The remaining power traces $T_{(N-M)}$ form a set $T^U = \{T_j|M \leq j < N\}$. Each power trace in T^L contains K power consumption points. The signal-to-noise ratio (SNR)

values of these power consumption points are calculated using (1) [24]. In (1), Var and E denote variance and mathematical expectation, respectively. Z represents intermediate value. L_t represents the i-th power trace, where $i \in [0, K)$. Each power point has a SNR value. The index corresponding to the highest SNR value is denoted by IN_{max}.

$$snr_i = \frac{Var\left[E[L_t|Z]\right]}{E\left[Var[L_t|Z]\right]} \tag{1}$$

Step 2. Take out the power point corresponding to IN_{max} from M power traces and plot a probability distribution graph. The probability distribution contains 9 parts, each part corresponds to a Hamming weight(HW). The power consumption points in each part have the same HW. Based on the distribution of power points with known Z_M, determine the HW corresponding to each part. For example, if the Z corresponding to a power point is 209, then the HW of Z is 4. This power point falls in the area of the probability distribution with HW equal to 4.

Step 3. In the previous step, we calculated the HW corresponding to each part of the probability distribution graph. In this step, we can calculate the HW for each power trace in the $\boldsymbol{T^U}$ using this probability distribution graph. These HW values are used to construct a HWPF. First, take out all the power traces in the $\boldsymbol{T^U}$ and calculate the $\boldsymbol{IN_{max}}$ corresponding to each power trace. Then, the HW corresponding to each power trace can be calculated based on $\boldsymbol{IN_{max}}$. As an example, if the power point corresponding to $\boldsymbol{IN_{max}}$ is $\boldsymbol{p_j}$. This $\boldsymbol{p_j}$ is located in the area where HW is equal to 4 in the probability distribution graph, then the HW of the power trace corresponding to $\boldsymbol{IN_{max}}$ is also equal to 4.

3.2. Design of HWFilter method

The idea of HWFilter method is to construct a HWPF to filter the pseudo-labels generated by the SCA model during the training phase. This subsection focuses on the HWFilter method. **Fig. 3** describes the HWFilter method in detail. In **Fig. 3**, the HWFilter method consists of a construct HWPF phase and a Use HWPF phase. In the construct HWPF phase, we use the normal distribution method to construct a HWPF. In the use HWPF phase, we add the HWPF to the training process of the model. After that, we describe the two phases in detail.

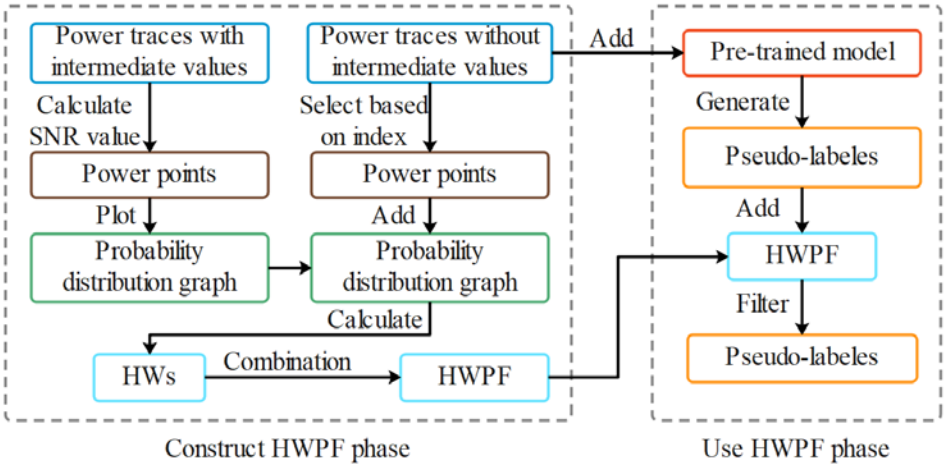

Fig. 3. The HWFilter method.

Construct HWPF phase: Constructing a HWPF by the normal distribution method. A detailed description of the normal distribution method is presented in Section 3.1. Take the construction of HWPF using the ASCADv1 with a fixed key dataset [24] as an example. First, we need to plot the SNR graph based on (1). The ASCADv1 with a fixed key dataset contains 60,000 power traces and intermediate values. Each power trace consists of 100,000 power points. We extracted 40,000 power traces from the ASCADv1 with a fixed key dataset, including 80 with intermediate values and the rest without intermediate values. The SNR value corresponding to each power point on the power traces can be calculated using (1). The 100,000 power points can generate 100000 SNR values. The 100,000 SNR values are shown in **Fig. 4**.

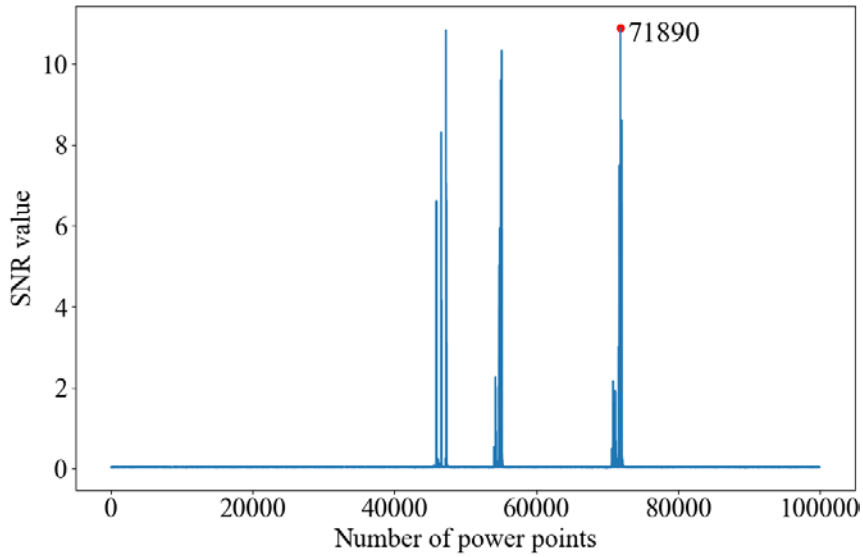

Fig. 4. One hundred thousand SNR values are related to the intermediate values, the red dot represents the index of power point with the maximum SNR value.

Then, the 80 power points are obtained sequentially from the 80 power traces. The index values of these power points are equal to the index values of the red points in **Fig. 4**. These power consumption points are plotted as a probability distribution graph, known as **Fig. 5**. The probability distribution graph contains 9 parts, each corresponding to a HW. The HWs corresponding to the power traces without intermediate values can be determined in **Fig. 5**. For example, if the power value of a power point is 21, the HW corresponding to this power consumption trace is 4. Following the above step, we can obtain the HW corresponding to all power traces without intermediate values. These HWs are combined into a HWPF that filters out pseudo-labels.

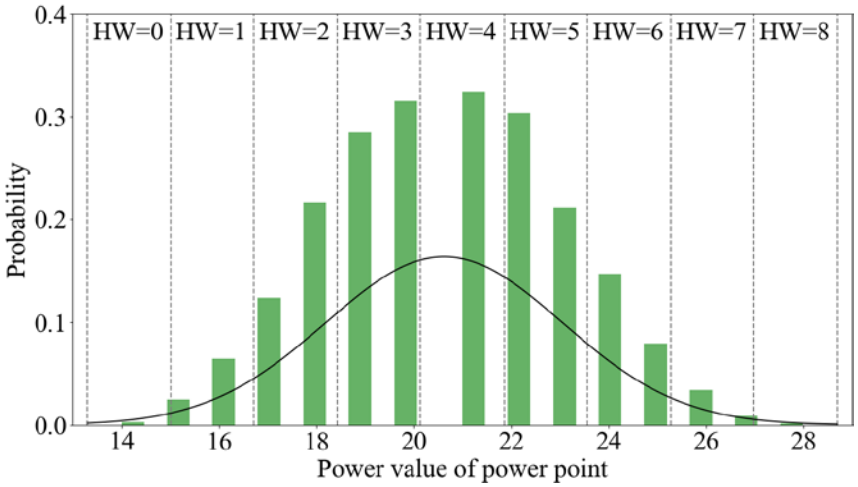

Fig. 5. The probability distribution graph of 100,000 power points

Use HWPF phase: This step describes the process of using HWPF. **Fig. 6** shows how HWPF filters pseudo-labels during the training phase of the model. In **Fig. 6**, the pre-trained model generates some pseudo-labels. These pseudo-labels are first filtered through a confidence interval and then filtered using HWPF. If the HW of a pseudo-label is equal to the corresponding HW in HWPF, the pseudo-label is paired with the corresponding power trace. Finally, this pseudo-label and the corresponding power trace are added to the new train set. This process is repeated until the performance of the model reaches expectations.

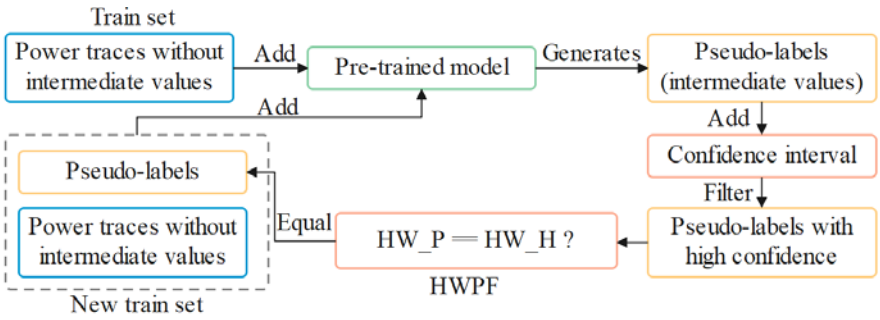

Fig. 6. The process of using HWPF.

3.3. Structural design of SCA model

The SCA model used in this work is based on the VGG network design [25]. The model contains a convolutional module and a fully connected module. The convolutional module is used to remove noise in power traces and extract local feature values. The activation function in the convolution module is the ReLU function. These raw power traces are processed by the convolution layer to generate a large number of local features. Some of these local features have little relevance to the labels or are noisy. After each convolution layer, an average pooling layer is added to reduce the size of the local features and prevent overfitting. We used a grid search method to optimize multiple hyperparameters, including the number of convolutional layers, learning rate, and batch size, from which the optimal hyperparameters were selected. The hyperparameter search spaces of the SCA model are shown in **Table 1**.

Table 1. The hyperparametric search spaces of SCA models.

Hyperparametric	Options
Convolution layers	1 to 5 in a step of 1
Convolution kernel size	7 to 17 in a step of 2
Pooling size	2 to 8 in a step of 1
Learning Rate	1e-4, 8e-5, 6e-5, 4e-5, 2e-5
Loss function	NLLLoss, MSE, CEL, FLR
Batch size	50 to 300 in a step of 20
Training epoch	50 to 200 in a step of 20

We constructed SCA models with different structures using the hyperparameters in **Table 1** and tested these models. We selected a model with the best performance from these models. The model contains a convolutional module and a fully connected module. The convolution module includes four one-dimensional convolution layers, four activation functions, four maximum pooling layers and one flatten layer. This flatten layer is used to flatten the output of the convolutional layer. Since the dimension of the output of the last convolutional layer is 3, the output needs to be dimensionally reduced using the flatten layer. The flatten layer flattens the output of the convolutional layer, which facilitates the fully connected layer to map this output to the sample space. The model classifies classes of 256 and requires more fully connected layers. The last fully connected layer has 256 neurons. The size of each convolutional kernel is (11x1) and the size of each max-pooling layer is (2x2). The activation function of the convolutional module and the fully connected module is the RELU function. The learning rate and the loss function of the model are 0.0001 and the NLLLoss function, respectively. The batch size and training epoch are 200 and 300, respectively. The structure of the model is shown in **Fig. 7**.

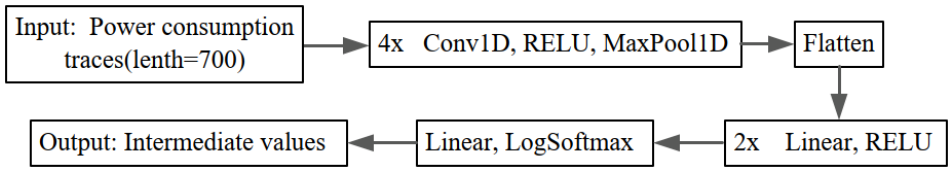

Fig. 7. The structure of the semi-supervised SCA model.

4. Experiment

In this section, we primarily present multiple public datasets and the training process of the semi-supervised SCA model. Our experiment is conducted using the Pytorch framework and Python scripts that run on an Intel Core i7-12700H CPU @ 2.3 GHz and an NVIDIA GeForce RTX 3060 Laptop GPU.

4.1. Datasets

In this study, ASCADv1 database and AES_HD dataset are used as experimental datasets. The ASCADv1 database and AES_HD dataset belong to the standard datasets in the SCA. The ASCADv1 database is publicly available at https://github.com/ANSSI-FR/ASCAD. The AES_HD dataset is publicly available at https://github.com/AESHD/AES_HD_Dataset. After that, we describe the two datasets in detail.

ASCADv1 database: The ASCADv1 database was obtained by sampling EM traces on the ATMega8515. This database includes an ASCAD with a fixed key dataset and an ASCAD with variable keys dataset. We denote ASCAD with a fixed key dataset and ASCAD with variable keys dataset as ASCAD_f and ASCAD_r, respectively. The ASCAD_f has 60,000 power traces. Each power trace contains 100,000 power points. The ASCAD_r has 300,000 power traces. Each power trace contains 200,000 power points. The detailed ASCADv1 database is described in [24]. In this experiment, we used power traces related to the 3-th byte key in the ASCAD_f. Although the ASCAD_f dataset has power traces and labels that have been pre-processed, it cannot be used directly as the dataset for our experiments. We extracted the desired power traces from raw power traces and generated the corresponding labels. The 60,000 raw power traces T consist of four parts: train set with labels T_L, train set without labels T_U, validation set T_V, and test set T_T.

$$\begin{aligned} T &= T_L + T_U + T_V + T_T, \\ T_L &= \{T_1, \dots, T_l\} \\ T_U &= \{T_{l+1}, \dots, T_u\} \\ T_V &= \{T_{u+1}, \dots, T_v\} \\ T_T &= \{T_{v+1}, .., T_{60000}\} \end{aligned} \tag{2}$$

The intermediate value labels M associated with these power traces are shown in (3).

$$\begin{aligned} \mathrm{M} &= M_L + M_V + M_T \\ M_L &= \{M_0, \dots, M_l\} \\ M_V &= \{M_{u+1}, \dots, M_v\} \\ M_T &= \{M_{v+1}, \dots, M_t\} \end{aligned} \tag{3}$$

Where l, u, v and t denote the length of each dataset, respectively. These intermediate values can be obtained by (4) [24]. In (4), M_i and $mask_i$ represent the i-th intermediate value and the i-th mask value, respectively. p_i and $mask_i$ represent the i-th plaintext and the i-th key, respectively. The $sbox$ represents the byte substitution operation in the AES algorithm.

$$M_i = Sbox(p_i \oplus k_i) \oplus mask_i \tag{4}$$

Before training the model, we need to perform the pre-processing operation to select the power interval. The Pearson Correlation Coefficient (PCC) [26] is used for pre-processing operations. The PCC is shown in (5).

$$\rho_{T,M} = \frac{E[(T-\mu_T)(M-\mu_M)]}{\sigma_T \sigma_M} \tag{5}$$

Where μ_T and μ_Z denote covariance of the power traces T and intermediate values M, respectively.The Σ_T and σ_Z denote standard deviation of T and Z, respectively. We obtained the PCCs associated with 1400 power points in ASCAD_f by (5). These PCCs are shown in **Fig. 8**. We then obtained the PCCs associated with 1400 power points in ASCAD_r using the same operation. These PCCs are shown in **Fig. 9**.

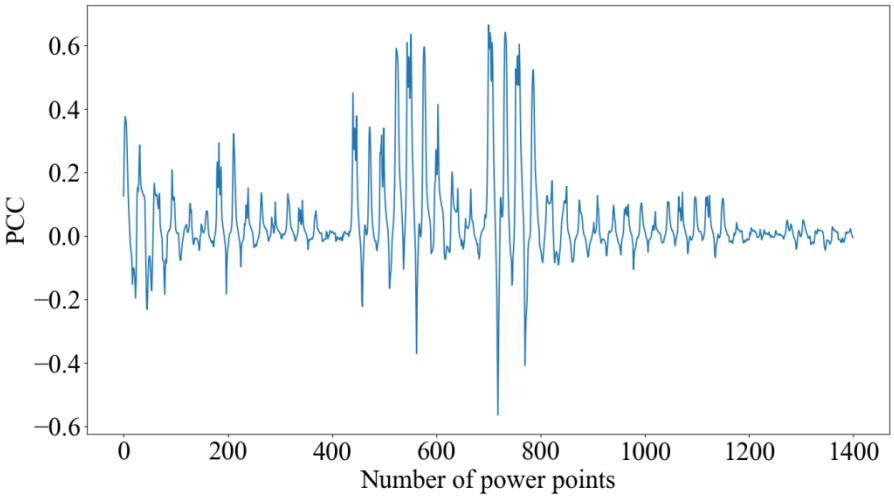

Fig. 8. About the PCC of 1400 power points in ASCAD_f.

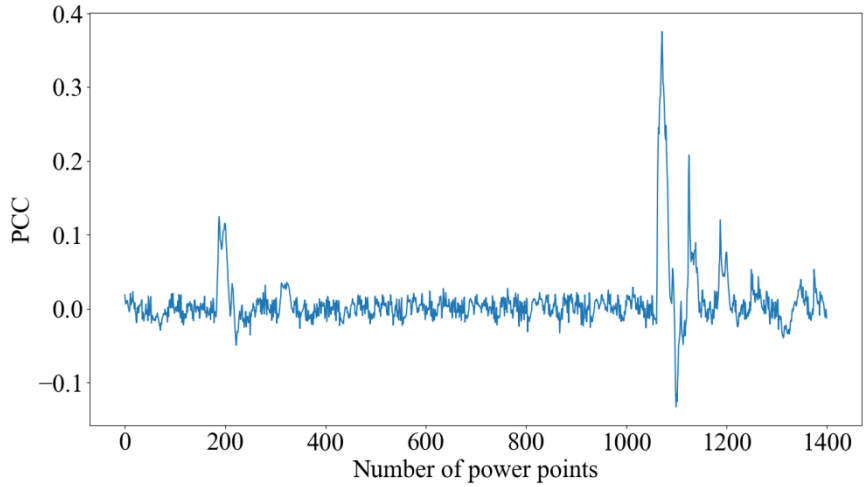

Fig. 9. About the PCC of 1400 power points in ASCAD_r.

According to PCC, we can find the corresponding power points in ASCAD_f and ASCAD_r respectively. The power interval composed of these power points is used as the sample in our experiment.

AES_HD dataset: The AES_HD dataset targets an unprotected implementation of AES-128. This dataset contains 100,000 power traces, each power trace containing 1250 power points. We took 700 power points with high PCC using (5) for the experiment. These 700 power points are shown in **Fig. 10**.

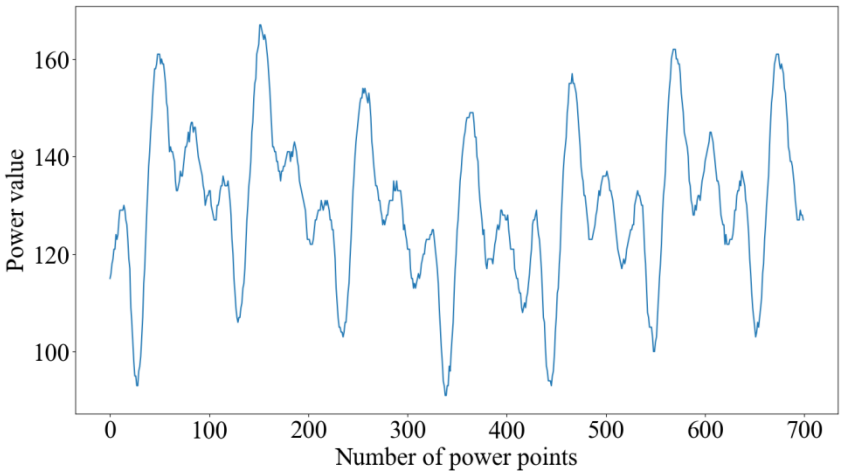

Fig. 10. The power values of 700 power points in the AES_HD dataset.

4.2. The training process of semi-supervised SCA model

The training process of semi-supervised SCA model differs from that of supervised SCA model. This training process includes two phases: the pre-training phase and the training phase. In the pre-training phase, the model is trained on a labeled dataset where the labels are intermediate values. After a period of time, the model reaches the level of predicting intermediate values based on power traces. In the training phase, the model is trained on a mixed dataset that contains both labeled and unlabeled power traces. When this model is trained on the unlabeled power traces, it generates some pseudo-labels. After these pseudo labels are filtered, some of them are added to the mixed dataset. This dataset is used as a train set for the model. The training process of the semi-supervised model is shown in **Algorithm 1**.

Algorithm 1 The training process of semi-supervised SCA model.

Input: Train set with labels: $D^L = \{(t_i^L,\ l_i^L) \mid i \in [0,\ N)\}$.
Train set without labels: $D^U = \{t_j^U \mid j \in [0,\ M)\}$.

for $e_{pre-train} \leftarrow 0$ **Do**
  for $i \leftarrow 0$ **Do**
    $(t_i^L, l_i^L) \leftarrow D^L$.
    Model_pre-train(t_i^L, l_i^L).
  end for
end for
for $e_{train} \leftarrow 0$ **Do**
  for $j \leftarrow 0$ **Do**
    $t_l^U \leftarrow D^U$.
    Model_train(t_l^U).
    The model generates pseudo-label p_j^U.
    Determine whether the confidence level of p_j^U is greater than 0.1.
    Determine whether the HW of p_j^U is equal to the corresponding HW in the HWPF.
    This p_j^U is added to the pseudo-label dataset D^P.
  end for
  for $k \leftarrow 0$ **Do**
    $(t_k^P,\ p_k^P) \leftarrow D^P$.
    Model_train($t_k^P,\ p_k^P$).

19: **end for**
20: **end for**

5. Experimental results and analysis

The GE [22] was used as the evaluation metric for this experiment. In SCA, GE is a frequently used metric for evaluating the security of cryptographic implementations. GE quantifies the number of attempts an attacker needs to make to successfully guess the secret key. The use of GE allows for an effective evaluation of the attack effects of an SCA model. The GE indicates the mean rank of real key sorted by predicted probabilities or scores. For each power trace, the model predicts all probability that guesses key k^* are real key k^c. These possibilities are accumulated and ranked. The GE is 0 when k^* with the highest probability is equal to the k^c. The formula corresponding to GE is as follows.

$$GE = average(\sum_{i=1}^{50} rank(k^* = k^c)) \tag{6}$$

This section focuses on evaluating the performance of HWFilter on the semi-supervised SCA model. Semi-supervised SCA models include the HW model, the Hamming Distance (HD) model, and the identity(ID) model. The ID model is the most widely used semi-supervised SCA model. In this experiment, we tested the performance of HWFilter using the ID model. We first compared the performance of ID model with HWPF and the performance of ID model without HWPF by using the ASCAD_f. We classify the power traces into 3 levels, where each level contains labeled power traces and unlabeled power traces. These three levels are shown in (7).

$$\begin{aligned} &Level1 \quad T_L{:}\,80, T_U{:}\,7920\ (1\%\ vs\ 99\%) \\ &Level2 \quad T_L{:}\,320, T_U{:}\,7680\ (4\%\ vs\ 96\%) \\ &Level3 \quad T_L{:}\,1600, T_U{:}\,6400\ (20\%\ vs\ 80\%) \end{aligned} \tag{7}$$

We used ASCAD_f and ASCAD_r to conduct the experiments, respectively. **Fig. 11** shows the GE of different ID models on ASCAD_f. The performance of ID models with HWPF is better than those without HWPF when the number of power traces with intermediate values is very small.

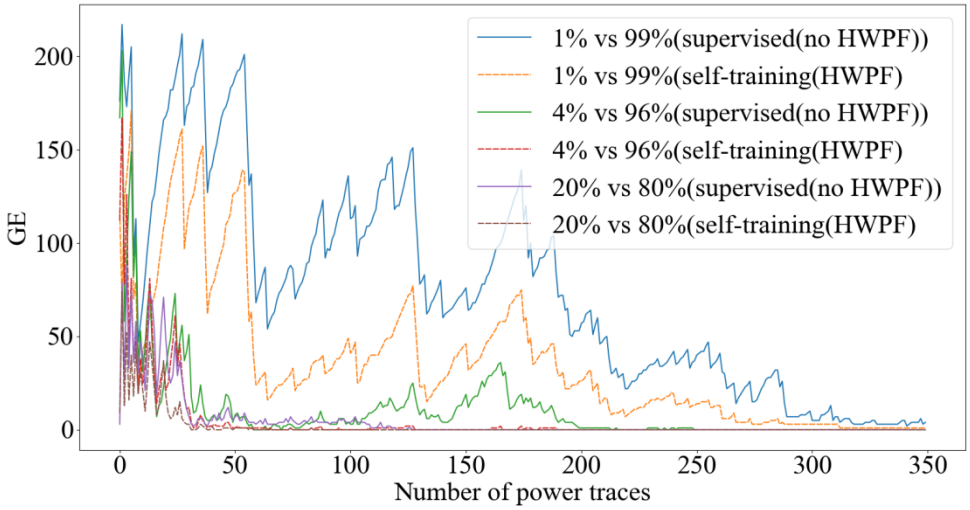

Fig. 11. The GE results of different ID models on ASCAD_f.

We then tested the performance of ID model with HWPF on the ASCAD_r. Although the ASCAD_r is more difficult than the ASCAD_f dataset, ID model with HWPF still performs well. The ID model with HWPF can use fewer power traces to recover key when the number of power traces with intermediate values is reduced. The performance of different ID model on the ASCAD_r is shown in **Fig. 12**.

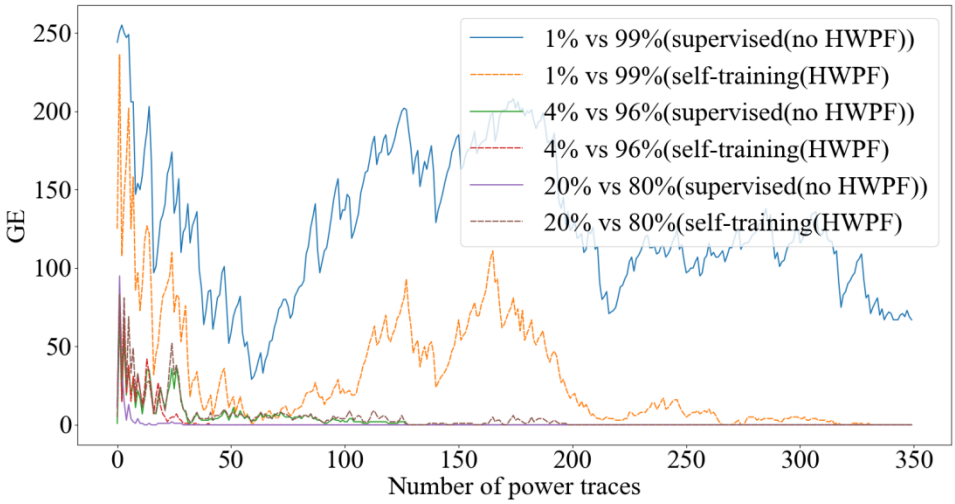

Fig. 12. The GE results of different ID models on ASCAD_r.

Table 2 shows the performance of HWPF on the AES_HD dataset. Although the maximum PCC of the AES_HD dataset is only 0.041, HWPF can still improve the performance of the ID model. When the number of labeled power traces is only 10,000, the performance of the model with HWFilter is 12% higher than that of Picek et al. model [14].

Table 2. GE results on AES_HD dataset (– if not reached).

Dataset size	Proposed method	Picek et al., "2019[14]
500+24.5k	-	-
1k+24.5k	27523	-
10k+15k	10971	12385

Table 3 shows the performance of HWPF on the ASCAD_f. The performance of the semi-supervised SCA model with HWPF is higher than that of the supervised SCA model for a smaller number of labeled power traces. As the number of labeled power consumption trajectories increases, the performance of the semi-supervised SCA model with HWPF and the supervised SCA model converge to be equal.

Table 3. GE results on ASCAD_f (– if not reached).

Dataset size	Proposed method	Ryad et al., "2020 [24]
250+11.75k	321	-
500+11.5k	54	265
5k+7k	33	36

Table 4 shows the performance of HWPF on the ASCAD_r. Semi-supervised SCA models with HWPF still have advantages over supervised SCA models on ASCAD_r. The semi-supervised SCA model with HWPF has maintained good performance under the condition of being trained with different labeled power traces.

Table 4. GE results on ASCAD_r (– if not reached).

Dataset size	Proposed method	Ryad et al., "2020 [24]
250+11.75k	-	-
500+11.5k	201	375
5k+7k	47	53

6. Conclusions

In this paper, a HWFilter method is proposed to improve the performance of semi-supervised SCA models. This method can reduce the number of erroneous pseudo-labels and accelerate the convergence of the model. In addition, we propose a normal distribution method for constructing HWPF. This HWPF can be used to filter pseudo-labels generated by semi-supervised SCA models. We also designed the structure of the semi-supervised SCA model using a grid search method. Finally, we conducted semi-supervised SCA experiments using the ASCADv1 database and the AES_HD dataset. These experimental results show that HWFilter can promote the training of semi-supervised SCA models. In particular, when the number of labeled power traces is small, the model trained using the HWFilter method outperforms the model trained using the supervised method. In this paper, we present some methods to solve the problem of erroneous pseudo-labels that hinder the training of semi-supervised SCA models. More research is needed in the future to investigate how semi-supervised learning can further promote the development of the SCA field.

Acknowledgment

This research is supported by the Hunan Provincial Natural Science Foundation of China(2022JJ30103), the Science and Technology Innovation Program of Hunan Province(2016TP1020), "the 14th Five-Year Plan" Key Disciplines and Application oriented Special Disciplines of Hunan Province(xiangjiaotong[2022]351), Open fund project of Hunan Provincial Key Laboratory of Intelligent Information Processing and Application for Hengyang Normal University(2022HSKFJJ011).

References

[1] Picek S, Perin G, Mariot L, et al., "Sok: Deep learning-based physical side-channel analysis," *ACM Computing Surveys*, vol. 55, no. 11, pp. 1-35, 2023. Article (CrossRef Link)

[2] Rijsdijk J, Wu L, Perin G, et al., "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 677-707, 2021. Article (CrossRef Link)

[3] Ito A, Ueno R, Homma N, "Perceived information revisited: New metrics to evaluate success rate of side-channel attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2022, no. 4, pp. 228-254, 2022. Article (CrossRef Link)

[4] Yap T, Benamira A, Bhasin S, et al., "Peek into the Black-Box: Interpretable Neural Network using SAT Equations in Side-Channel Analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2023, no. 2, pp. 24-53, 2023. Article (CrossRef Link)

[5] Ramezanpour K, Ampadu P, Diehl W, "SCAUL: Power side-channel analysis with unsupervised learning," *IEEE Transactions on Computers*, vol. 69, no. 1, pp. 1626-1638, 2020. Article (CrossRef Link)

[6] Cao, P., Zhang, C., Lu, X., "Cross-Device Profiled Side-Channel Attack with Unsupervised Domain Adaptation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 4, pp. 27-56, 2021. Article(CrossRef Link)

[7] Kulow A, Schamberger T, Tebelmann L, et al., "Finding the needle in the haystack: Metrics for best trace selection in unsupervised side-channel attacks on blinded RSA," *IEEE Transactions on Information Forensics and Security*, vol. 16, no. 3, pp. 3254-3268, 2021. Article(CrossRef Link)

[8] Cheng J, Liu W, Sun N, et al., "A machine learning low-dropout regulator-assisted differential power analysis attack countermeasure with voltage scaling," *International Journal of Circuit Theory and Applications*, vol. 51, no. 7, pp. 3105-3117, 2023. Article(CrossRef Link)

[9] Do N T, Hoang V P, Pham C K, "Low Complexity Correlation Power Analysis by Combining Power Trace Biasing and Correlation Distribution Techniques," *IEEE Access*, vol. 10, pp. 17578-17589, 2022. Article(CrossRef Link)

[10] Cai J, Hao J, Yang H, et al., "A review on semi-supervised clustering," *Information Sciences*, vol. 632, pp. 164-200, 2023. Article (CrossRef Link)

[11] Van Engelen J E, Hoos H H, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373-440, 2020. Article(CrossRef Link)

[12] Taha K, "Semi-supervised and un-supervised clustering: A review and experimental evaluation," *Information Systems*, vol. 114, pp. 102178, 2023. Article (CrossRef Link)

[13] Chen Y, Tan X, Zhao B, et al., "Boosting Semi-Supervised Learning by Exploiting All Unlabeled Data," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Oxford, UK, pp. 7548-7557, 2023. Article (CrossRef Link)

[14] Picek S, Heuser A, Jovic A, et al., "Improving side-channel analysis through semi-supervised learning," in *Proc. of International Conference on Smart Card Research and Advanced Applications*, Springer, Montpellier, France, pp. 35-50, 2019. Article (CrossRef Link)

[15] Batina L, Djukanovic M, Heuser A, et al., "It started with templates: The future of profiling in side-channel analysis," in *Security of Ubiquitous Computing Systems: Selected Topics*, vol. 3, 2021, pp. 133-145. Article (CrossRef Link)

[16] Panoff M, Yu H, Shan H, et al., "A Review and Comparison of AI-enhanced Side Channel Analysis," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 3, pp. 1-20, 2022. Article (CrossRef Link)

[17] Hettwer B, Gehrer S, Güneysu T, "Applications of machine learning techniques in side-channel attacks: a survey," *Journal of Cryptographic Engineering*, vol. 10, pp. 135-162, 2020. Article (CrossRef Link)

[18] Biao Liu, Zhao Ding, Yang Pan, Jiali Li, and Huamin Feng, "Side-channel attacks based on collaborative learning," in *Proc. of Data Science: Third International Conference of Pioneering Computer Scientists, Engineers and Educators*, Springer, Changsha, China, pp. 549-557, 2017. Article(CrossRef Link)

[19] Tang H, Jia K, "Towards discovering the effectiveness of moderately confident samples for semi-supervised learning," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, New Orleans, USA, pp. 14658-14667, 2022. Article(CrossRef Link)
[20] Huan P T, Thong P H, Tuan T M, et al., "TS3FCM: trusted safe semi-supervised fuzzy clustering method for data partition with high confidence," *Multimedia Tools and Applications*, vol. 81, no. 9, pp. 12567-12598, 2022. Article(CrossRef Link)
[21] Wang W, Zhang M L, "Semi-supervised partial label learning via confidence-rated margin maximization," in *Proc. of the 34th International Conference on Neural Information Processing Systems*, pp. 6982-6993, 2020. Article(CrossRef Link)
[22] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni, "The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 209-237, 2018. Article(CrossRef Link)
[23] Zhang H, Yang W, "Template Attack Assisted Linear Cryptanalysis on Outer Rounds Protected DES Implementations," *The Computer Journal*, vol. 66, no. 6, pp. 1434-1451, 2023. Article(CrossRef Link)
[24] Ryad Benadjila, "Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas, "Deep learning for side-channel analysis and introduction to ascad database," *Journal of Cryptographic Engineering*, vol. 10, no. 2, pp. 163-188, 2020. Article(CrossRef Link)
[25] Huanhuan Ran, Shiping Wen, Qian Li, Yuting Cao, Kaibo Shi, and Tingwen Huang, "Compact and stable memristive visual geometry group neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no 2, pp. 987-998, 2023. Article(CrossRef Link)
[26] Alexander Ly, Maarten Marsman, and Eric-Jan Wagenmakers, "Analytic posteriors for Pearson's correlation coefficient," *Statistica Neerlandica*, vol. 72, no. 1, pp. 4-13, 2018. Article(CrossRef Link)

Hong Zhang received the B.S. degree from Hunan Institute of Technology, Hengyang, China, in 2019 and he is currently working toward a Master's degree in Hengyang Normal University, Hengyang, China. Since 2021, his current research interests include embedded computing and information security.

Lang Li received his PhD and Master's degrees in computer science from Hunan University, China, in 2010 and 2006, respectively, and earned his BS degree in circuits and systems from Hunan Normal University, China in 1996. Since 2011, he has been working as a professor in the College of Computer Science and Technology at the Hengyang Normal University, China. His research interests include embedded computing and information security.

Di Li received the B.S. degree from Hunan University of Science and Engineering, Yongzhou, China, in 2020 and he is currently working toward a Master's degree in Hengyang Normal University, Hengyang, China. Since 2020, his current research interests include embedded computing and information security.