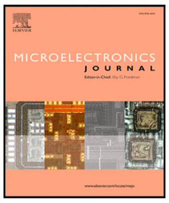
Microelectronics Journal 162 (2025) 106721

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo



Optimizing label correlation in deep learning-based side-channel analysis

Shengcheng Xia, Lang Li *, Yu Ou, Jiahao Xiang

College of Computer Science and Technology, Hengyang Normal University, Hengyang, 421002, China
Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang, 421002, China

ARTICLE INFO

Keywords:
Side-channel analysis
Sample correlation locally
Deep learning
Kernel density estimation
Profiling analysis

ABSTRACT

Label distribution learning techniques can significantly enhance the effectiveness of side-channel analysis. However, this method relies on using probability density functions to estimate the relationships between labels. The settings of parameters play a crucial role in the impact of the attacks. This study introduces a non-parametric statistical method to calculate the distribution between labels, specifically employing smoothing with the Gaussian kernel function and adjusting bandwidth. Then, the aggregation of the results from each label processed by the Gaussian kernel facilitates a hypothesis-free estimation of the label distribution. This method accurately represents the actual leakage distribution, speeding up guess entropy convergence. Secondly, we exploit similarities between profiling traces, proposing an analysis scheme for sample correlation locally of label distribution learning. Furthermore, Signal to-Noise Ratio (SNR) is employed to re-extract and reduce dataset dimensions to 500 power consumption points, resulting in noise reduction. Our results showcase the successful training of 800 profiling traces using our method for sample correlation locally of label distribution learning, with the findings indicating its exceptional performance.

1. Introduction

Side-channel analysis(SCA) is acknowledged as one of the most potent techniques for attacking encryption algorithms. This analysis is divided into two types: non-profiling analysis and profiling analysis. Non-profiling analysis encompasses techniques like simple power analysis [1], differential power analysis [2] and correlation power analysis [3]. The other category is profiling analysis, which is based on the assumption that the attacker has access to the same device as the target [4]. In this method, the attacker assumes access to the same device as the target device. They collect a substantial number of profiling traces during the encryption process, construct a probabilistic model, and then endeavor to match features to crack the key.

In recent years, deep learning based side-channel analysis has gradually supplanted traditional approaches, since Maghrebi et al. showcased the exceptional performance of neural networks, particularly convolutional neural networks (CNN) [5]. Subsequently, Ryad Benadjila et al. conducted an experiment to evaluate CNN hyperparameters using the ASCAD dataset [6]. G. Zaid proposed a strategy for selecting CNN hyperparameters, focusing on aspects like filter counts, sizes of kernels, step lengths, and the count of neurons within fully connected layers [7]. Lei Ni et al. proposed a side-channel attack method based on CNN fusion, which enhances attack performance and simplifies the process of hyperparameter tuning [8]. Wouters and their team achieved a similar attack efficacy through techniques like data normalization and simplifying the network structure [9]. Additionally, Perin and colleagues showcased the superiority of stochastic model ensembles by investigating model generalization [10]. Suvadeep Hajra and colleagues proposed a new shift-invariant TN-based model, EstraNet, which offers a more efficient and scalable solution, significantly enhancing the performance of side-channel attacks [11]. Wu and Rijsdijk explored strategies for automatic hyperparameter tuning using Bayesian optimization [12] and reinforcement learning [13]. These methods led to performance enhancements, despite the computational expense involved. However, methods that combine deep learning with side channels heavily rely on having a sufficient number of profiling traces. An often overlooked challenge in side-channel analysis is the difficulty in acquiring numerous profiling traces.

Zhimin Luo proposed the mixup technique to reduce the reliance on a large quantity of profiling traces for successful attacks. This aims to enhance the effectiveness of side-channel analysis that employs deep learning methods [14]. This method augments the dataset by generating new virtual training data. Jun-Nian Wang introduced a novel model that combines side-channel analysis with Conditional Generative Adversarial Network. They also utilized simulated profiling traces to augment and complement the original profiling traces dataset, thereby reducing the number of profiling traces needed to obtain the

* Corresponding author.
E-mail address: lilang@hynu.edu.cn (L. Li).

https://doi.org/10.1016/j.mejo.2025.106721
Received 12 May 2024; Received in revised form 27 December 2024; Accepted 5 May 2025
Available online 23 May 2025

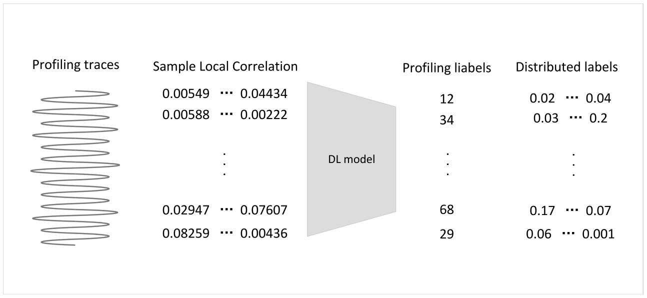


Fig. 1. Sample correlation locally based side-channel analysis.

correct key [15]. However, this approach demands significant computing resources and time to train complex models. In response, many researchers have opted to transform the classification problem into a similarity problem or utilize features within the dataset to diminish the number of profiling traces. Li et al. accomplished this by converting the critical classification problem into a similarity measurement task. They reformulated the secret key recovery problem as a similarity determination task in the Siamese network [16].

In recent work, Wu et al. proposed enhancing performance by shifting labels encoded as one-hot vectors to a distribution function centered on the corresponding labels of the drive [17]. Researchers utilize the probability density function to compute the distribution relationship between labels. However, when using the probability density function [18], assuming parameters in advance is necessary, and different parameter assumptions can impact attack performance. Additionally, in the attack scenario, the distribution of leaked traces is uncertain, making it challenging to achieve optimal performance using the probability density function. Further discussion on this aspect will be provided in Section 3.1. Machine learning methods are employed to compute the correlation between labels, enhancing attack performance significantly compared to traditional deep learning combined with side-channel analysis. However, only the correlation between labels is calculated to expedite convergence, while the correlation between profiling traces remains unexploited for performance enhancement. As widely acknowledged in the side-channel academia, when the intermediate value is the same, its corresponding profiling traces are roughly similar, indicating correlation between profiling traces. A more thorough discussion on this topic can be found in Section 3.2.

We utilize the aforementioned challenges as the driving force behind our work. Firstly, to address the affect of different parameter assumptions on the performance of attack and the issue of uncertain leak traces distribution, the nonparametric statistical method of kernel density estimation (KDE) [19] is employed. Using a smooth Gaussian kernel function [20] and appropriate bandwidth, the label distribution is estimated without making assumptions. The implementation details can be found in Section 3. Experimental results show that our method not only resolves the problem of parameter assumptions but also yields improved attack performance.

To harness the correlation between profiling traces, we intro duce Label Distribution Learning by Exploiting Sample Correlations Locally (LDL-SCL) [21] to execute the attack. The proposed analysis scheme is depicted in Fig. 1. Our experiments demonstrate that this method exhibits superior adaptability and generalization ability, along with excellent performance. In our experimental setup, training 800 power consumption traces is necessary to execute the attack successfully.

To summary, the main contributions of this paper are as follows:

- The proposed method utilizes kernel density estimation to compute the distribution relationship between labels, resulting in a more accurate representation of the actual leakage distribution. This approach significantly accelerates the convergence speed of the guess entropy(GE). Compared to the label distributed learning proposed by Wu's team, our method, which combines kernel-density estimation with label distribution learning, enhances attack performance.
- A novel and efficient side-channel analysis scheme tailored for scenarios with limited profiling traces is introduced. By distance algorithms in machine learning to calculate the local correlation of samples, our approach significantly enhances attack performance compared to conventional deep learning schemes that combine side-channel analysis with learning distributed labels.
- We re-extracted the locations of the power consumption points for each data in the ASCAD dataset using SNR and conducted dimensionality reduction. The attack experiments performed using this re-extract dataset demonstrate excellent performance compared to the original dataset.

The remainder of this article is structured as follows. In Section 2, related side-channel analysis, commonly used deep learning combined with SCA schemes, and information related to the ASCAD dataset are introduced. Section 3 presents the analysis method using kernel density estimation combined with label distribution learning, followed by the proposal of the attack scheme based on label distribution learning of local correlation of samples. Section 4 experimentally validates our proposed method using different datasets, training models, and leakage models. Section 5 discusses the limitations. Finally, in the conclusion, we summarize the article and suggest future research directions.

2. Preliminary

In this section, the notation used and profiling side-channel analysis are introduced. Additionally, the leakage model and the dataset are discussed.

2.1. Notation

We utilize letters such as X and Y to represent sets, and lowercase letters like x and y to denote data within the corresponding sets, respectively. Sans-serif fonts are employed for functions (eg.,f).

A training set T is defined as a set T_i of traces, each of which is linked with a corresponding label y from the complete set of labels denoted by $Y = \{y_1, y_2, y_3, \dots y_i\}$.

2.2. Profiling side channel analysis

Profiling side-channel analysis involves two phases. the learning analysis phase and the testing attack phase.

During the learning phase, analytical models are constructed to map inputs to a set of outputs. This model is built by assessing the leakage model of sensitive operations, with parameters tuned to optimally map input power traces to the correct label output. Typically, a validation set is employed to determine when the learning process is complete.

Various datasets are utilized in the test attack phase to obtain predictions of labels to evaluate the model. The training model handles each analysis trace, producing an output in the form of a probability vector. Each index of the vector represents the probability that the label is associated with the leakage profiling trace. With this probability matrix, the attack performance can be assessed. Commonly used modeling attacks based on deep learning utilize one-hot encoding to represent intermediate values as labels during the analysis phase. This method of representing labels closely resembles the application of deep learning in other fields and is the most commonly employed approach for combining side-channel analysis and deep learning. During the test attack phase, the deep learning model outputs a probability vector, and the rank of the key is determined using GE metric.

Definition 1 (*Guessing Entropy (GE)*)**.** Guessing entropy measures the average position of the correct key $k*$ within the key guessing vector [17] g:

$$GE = \mathbb{E}(\text{rank}_{k^*}(g)) \tag{1}$$

where $\text{rank}_{k^*}(g) \in \{0, \dots, |K| - 1\}$. $\mathbb{E}$ is the mean value obtained from several instances of key ranking, typically calculated by repeatedly attacking with a profiling model using randomly chosen traces.

2.3. Leakage models

In side-channel analysis, the actual power consumption needs to be mapped to specific operands, which involves simulating the energy usage of the device. Specifically, the leakage model simulates the power consumption associated with processing one byte. Two widely used leakage models, namely the Hamming weight (HW) [22] and identity (ID) [23] leakage models, are frequently considered in the side channel domain.

The attacker assumes that the energy consumption is proportional to the number of bits set in the processed data, within the HW leakage model. This leakage model divides a single intermediate byte of the AES cipher into 9 classes, disregarding the numerical values processed before and after this data. This model is simpler than the ID leakage model and is typically chosen when the attacker lacks information about the table. It is also used when the attacker only knows a portion of the table, but not the consecutive numerical values processed by that part.

In contrast, the ID leakage model calculates the total number of transitions between 0 and 1, as well as between 1 and 0, in a digital circuit within a specific time frame. This total count of transitions is used to represent the energy consumption of the circuit during that period. By simulating the entire circuit and dividing it into smaller time periods, an power trace can be generated. This model results in 256 classes for a single intermediate byte of the AES cipher. Unlike the Hamming weight model, the Hamming distance model can simulate the energy consumption values for that part of the table when the attacker knows the data processed continuously in that section of the table.

2.4. Datasets

The ASCAD dataset comprises the HDF5 file “ATMega8515_raw_traces” generated by the implementation running AES-128 and measuring power consumption [6]. It consists of two main parts: the raw data and the power consumption trace. The former contains plaintext, ciphertext, key, and mask, while the latter is a scalar dataset stored in HDF5 format. Three HDF5 files extracted from the “ATMega8515_raw_traces” file are available: ASCAD.H5, ASCAD_desync 50.H5, and ASCAD_desync100.H5. ASCAD.H5: this file contains synchronized traces comprising train and attack datasets without jitter. ASCAD_desync50.H5: it also contains train and attack datasets but with 50 sampling window jitter traces. ASCAD_desync100.H5: this file contains two 50-sampled window jitter traces. To facilitate the evaluation and testing of side-channel analysis techniques, researchers divided the ASCAD dataset into two versions based on the encryption operation and environment.

The ASCAD fixed dataset consists of 50,000 profiling traces and 10,000 attack traces, with each trace containing 700 breach points per data. The analysis and test sets utilize the same fixed key. This dataset is commonly referred to as ASCAD_f.

The ASCAD random dataset differs from the ASCAD fixed dataset in several ways. It comprises 200,000 traces for use in the analysis phase, with random plaintext and keys. In the attack phase, 100,000 traces are employed, with fixed keys and plaintext. Each data point in the ASCAD random dataset contains 1400 leak points. This dataset is commonly denoted as ASCAD_r. In our experiments, we utilized varying numbers of profiling and attack traces from both datasets. The dataset can be accessed at https://github.com/ANSSI-FR/ASCAD.

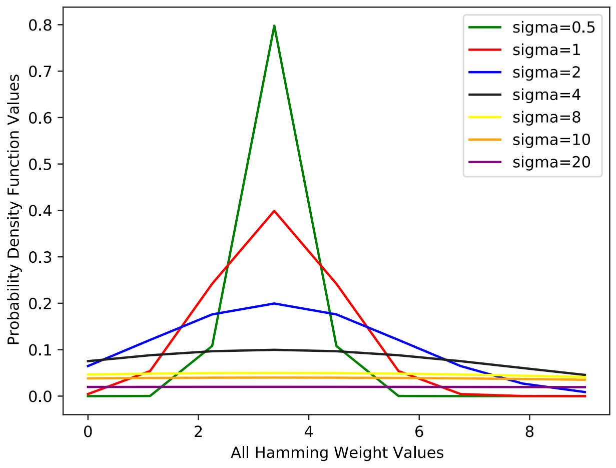


Fig. 2. The difference in distribution relationship between the Hamming weight value of 3 and other label values when parameters vary.

The TinyPower dataset differs from the ASCAD random and ASCAD fixed datasets. This dataset was collected using an STM32F3 target chip on the ChipWhisperer platform, with each power trace consisting of 10,000 sampling points [24]. To increase the dataset’s complexity and improve the practicality of the experiments, we introduced random Gaussian noise with a mean of 0 and a variance of 5. Based on the calculated SNR, we selected power points within the range of [6400:7100]. The dataset can be used to evaluate side-channel attack methods and is suitable for training and testing deep learning models.

3. Optimizing label distribution learning (OLDL)

In this section, two novel analysis methods are proposed. Combining kernel density estimation with label distribution learning is suggested to enhance attack performance in Section 3.1. An attack scheme leveraging the local correlation of samples is introduced in Section 3.2. Additionally, we delve into specific attack details and offer illustrations.

3.1. Kernel density estimation

In this subsection, we initially explore the rationale behind proposing kernel density estimation in combination with side-channel analysis and label distribution learning. Subsequently, the method is introduced, and ultimately, our attack scheme is presented.

Side-channel analysis based on label distribution learning is a multi-classification task. In this task, the probability density function is utilized to calculate the distribution relationship between labels. Labels are transformed into probabilistic relationships to train the deep learning model. However, the probability density function requires preset parameters and clear distribution types to express distribution relationships. It is only after extensive experiments and parameter tuning that a more realistic distribution can be revealed under ideal conditions. For illustration, Fig. 2 show the difference between the probability density distributions of Hamming weight 3 other Hamming weight values for different assumed parameters. Computing the distribution relationship of labels is crucial in label distribution learning. This process involves using a probability density function to represent the distribution leakage throughout the training scheme. Additionally, the proposed method is utilized to perform label prediction.

Fig. 2 illustrates the distribution relation values of the assumed parameters. However, there is a significant disparity between their distribution probabilities for different parameters. The experimental results demonstrate that using the probability density function and the distribution probability of parameter Siga = 1 to model the attack yields relatively good results. Nonetheless, many parameter values around 1 can also achieve satisfactory outcomes, making it challenging to pinpoint the optimal parameter hypothesis. Therefore, based on the above discussion, we introduce kernel density estimation.

Kernel density estimation is a method used to estimate the probability density of a dataset. It does not require assuming parameters of the function but rather learns the shape of the density automatically from the data. This method finds frequent application in tasks like machine learning and data processing. Based on our discussion, kernel density estimation is integrated into label distribution learning combined with SCA.

Definition 2 (*Kernel Density Estimation*)**.** The density estimate $\hat{f}(x)$ at any point x is obtained by placing a smooth kernel function K at each simulated data point X_i, summing these functions, and then dividing by the product of the total number of data points n and the bandwidth h.

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2}$$

where n denotes the size of the profiling traces, K denotes the kernel function, h stands for the bandwidth, x signifies the point where the density requires estimation, representing the value of the leak type, and x_i represents the generated simulated data.

In our scheme, a systematic approach is followed. Simulated data is first generated. Then, the kernel function and bandwidth are determined. Subsequently, the kernel density estimation is computed based on these parameters, and finally, the distribution relationship is calculated based on the labels. To delve into specifics, the labels are utilized as the mean to generate simulated data. This simulated data facilitates fitting the kernel density model later. It is worth noting that the number of generated data points can vary. In the case, 256 values were generated to ensure a robust estimate. Next, the kernel function is selected, which significantly impacts density estimation. The Gaussian kernel is selected in the method due to its effectiveness. The width of the kernel is determined by the bandwidth, thus influencing the smoothness of the estimation. Larger bandwidths result in smoother estimates, whereas smaller bandwidths yield sharper estimates. We use Asymptotic Mean Integrated Squared Error (AMISE) [25] minimization to select the optimal bandwidth as follows.

Definition 3 (*Asymptotic Mean Integrated Squared Erro*)**.** The AMISE quantifies the trade-off between variance and bias to select the optimal bandwidth h. It is defined as follows:

$$\text{AMISE}(h) = \frac{1}{nh}R(K) + \frac{h^4}{4}\mu_2^2(K)R(f'') \tag{3}$$

In Eq. (3), n is the number of samples, corresponding to the number of labels. For instance, if the training set contains 5000 samples, then $n = 5000$. $R(K)$ is the second moment of the kernel function, and $\mu_2(K)$ is its second central moment. For the Gaussian kernel, their respective values are $\frac{1}{2\sqrt{\pi}}$ and 1.

$$R(K) = \int K^2(x)\,dx = \frac{1}{2\sqrt{\pi}}. \tag{4}$$

$$\mu_2(K) = \int x^2 K(x)\,dx = 1. \tag{5}$$

$$R(f'') = \int \left(f''(x)\right)^2\,dx. \tag{6}$$

By taking the derivative of Eq. (3) with respect to h and setting it to zero, the optimal bandwidth formula is obtained as follows:

$$h = \left(\frac{R(K)}{R(f'')\left(\mu_2^2(K)\right)^4 n}\right)^{\frac{1}{5}}. \tag{7}$$

$R(f'')$ is the integral of the squared second derivative of the true density function $f(x)$, which is typically unknown. Therefore, we use a heuristic approach as a substitute:

$$h = \left(\frac{4\hat{\eta}^5}{3n}\right)^{\frac{1}{5}}. \tag{8}$$

Here, $\hat{\eta}$ is the standard deviation of the sample data. This formula is derived under the assumption that the kernel function is Gaussian and the data distribution is approximately normal. The standard deviation of the sample data is given as follows:

$$\hat{\eta} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}. \tag{9}$$

Here, x_i represents the label values, and $\bar{x}$ is the mean of the labels. For example, if the first 5000 power traces from the ASCAD dataset are used as the training set, then $n = 5000$, where x_i corresponds to the label values and $\bar{x}$ is their mean. Using Eq. (8), the bandwidth h is calculated as 0.27. To verify whether the calculated h is the optimal bandwidth, we will discuss the bandwidth sensitivity in Section 4.1.

Subsequently, the log-probability density value of the Hamming weight is calculated by fitting the density model using the simulated data generated earlier. These kernels are then stacked to obtain an overall density estimate by placing kernel functions near the data. Finally, the exponent of the log probability density value is computed. After this transformation, the probability density value for each label is obtained. This density distribution value replaces the previous labels for subsequent training and testing.

During the attack phase, we use stochastic gradient descent with the aim of reducing the loss function L. After the network is trained, it can generate a predicted label distribution y for any input x from the attack dataset with unknown labels, including inputs that are random. The predicted label corresponds to the one with the highest probability in y.

$$i^* = \arg\max_i \hat{y}_i \tag{10}$$

3.2. *Sample correlations locally*

In this subsection, first, the reason for proposing the utilization of local sample correlation in side-channel analysis is clarified. Then, we introduce the method, and finally, we present our attack scheme.

Previous studies have employed label distribution learning combined with SCA to exploit the correlation between labels. However, it is worth noting that the profiling traces space also exhibits correlations. For illustration, Fig. 3 displays the label classification corresponding to 200 different profiling traces, where each color represents a label classification. From the figure, it can be observed that different profiling traces correspond to the same label, suggesting a certain relationship between different profiling traces.

We also assessed the complexity of the dataset structure by analyzing the correlation between features and evaluating class imbalance. The class imbalance is evident in the HW leakage model due to significant differences in the number of profiling across different categories. We selected the first 5000 entries from the ASCAD dataset to analyze the correlation between features. Fig. 4 illustrates the feature correlations for this datasets of 5000 entries.

The diagonal exhibits a relatively perfect positive correlation from the top left corner to the top right corner, indicating the correlation of each feature with itself, as shown in Fig. 4. The color of the heatmap denotes the strength of the correlation coefficient, with darker shades representing stronger correlations. From the figure, it is evident that a complex correlation structure emerges in the stripe pattern across the entire heatmap. This indicates that the features of the dataset are not independent but rather interdependent. Hence, based on the above discussion, it is reasonable to consider leveraging the correlation of the profiling traces space to enhance the attack efficiency.

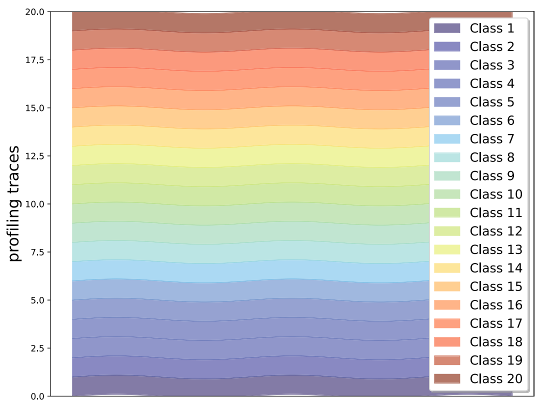


Fig. 3. Profiling traces classification plot.

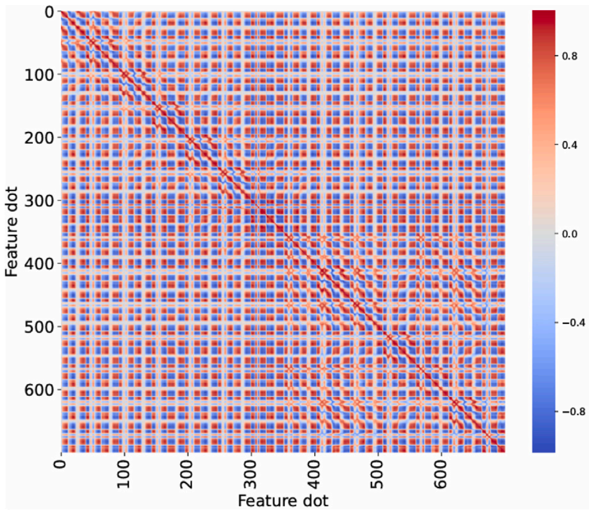


Fig. 4. Analyzing feature correlation.

Zheng et al. introduced a novel label distribution learning algorithm called LDL-SCL [21] to harness the correlation of local samples. It predicts the label distribution of unknown instances based on original features and local correlation vectors. Compared with label distribution learning alone, this approach has shown significant performance improvements. Building on our preceding discussion, we integrate this approach into SCA to formulate our attack scheme.

Let Y represent the set of labels, then

$$Y = \{y_1, y_2, y_3, \dots, y_i\}, \tag{11}$$

where i is the number of labels, consider a training set T,

$$T = \{(x_1, s_1, d_1), (x_2, s_2, d_2), \dots, (x_n, s_n, d_n)\}, \tag{12}$$

where x_i belongs to a profiling traces of X. $S_i = (s_1,\ s_2,\ s_3,\ \dots s_i)$ represents the profiling traces correlation, and $D_i = (d_1,\ d_2,\ \dots d_n)$ is the label distribution.

After obtaining the power traces and the labels, we calculate the correlation between the sample traces using Euclidean distance.

$$D = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{13}$$

where D denotes the Euclidean distance, n is the number of samples, and i represents the position of the sample traces. In this attack scheme, both x and y denote the sample power traces.

Hence, d_n is termed the degree of description from y_i to x_i, satisfying $d_n = 1$. The mapping function of LDL-SCL is denoted as:

$$f : X \to D \tag{14}$$

We establish the relevance of profiling traces by incorporating additional features into each traces. Specifically, a local relevance vector S is crafted for each profiling traces to represent the influence of local samples, augmenting the original features. It is evident that the length of the local correlation vector S equals the number of profiling traces. From our prior discussion, we discern that similar profiling traces might exhibit similar label correlations, hence, similar profiling traces may possess analogous local correlations. When computing the similarity between profiling traces, we opt to use the similarities in the label space rather than the sample space, as it is computationally more efficient. The details of our attack scheme are elaborated in Algorithm 1.

Algorithm 1 The LDL-SCL algorithm

```
    Input: training set $\{X, D\}$, parameters x, features, y and $epochs$.
    Output: the label distribution $D_t$.
 1: initialize x, $feature$, $y$;
 2: Compute the Euclidean distance between samples;
 3: Build local relevance features: Compute the mean of the label
    distribution;
 4: Build a model with additional inputs;
 5: Train( [X, features] , Y):
 6: Test( [X, features] ):
 7: for $l$ in $K$ do
 8:     $CORR_l$ = corr() ;
 9: end for
10: k = arg max corr;
```

In our attack scheme, we select the k nearest neighbors for each sample and calculate the local relevance features based on their labels. For each sample, the local relevance feature is determined by averaging the labels of its nearest neighbors. It is important to note that these local relevance features are included as inputs both during the training phase and in our predictions.

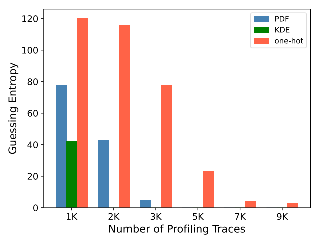
Based on the preceding discussion, the predicted distribution and the true distribution are two probability distributions. To measure the similarity between these two distributions, metrics like Kullback–Leibler (K-L) divergence [26], Jeffreys divergence [27] are commonly employed. However, in this context, K-L divergence is opted for as the loss function, defined as follows:

$$L_j(Q_a \parallel Q_b) = \sum_i Q_a^j \ln\left(\frac{Q_a^i}{Q_b^j}\right) \tag{15}$$

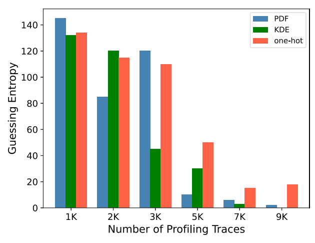
Here, Q_a and Q_b represent the predicted and true distribution probabilities, respectively, Q_a^j and Q_b^j is the number of elements in the distribution.

4. Experimental results

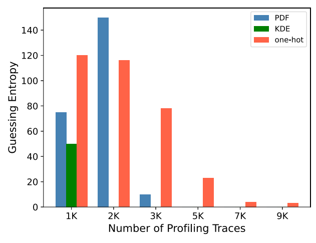
Firstly, we employ kernel density estimation combined with label distribution learning, as discussed in Section 4.1 and based on the HW leakage model. All experiments are conducted on a workstation configured with an NVIDIA GTX 1660 Ti GPU, an Intel Core i5-9300H CPU, and 32 GB of RAM, running Python 3.8 and TensorFlow 2.2.0. This method is applied separately to attack three datasets: firstly the ASCAD_f dataset, then the ASCAD_r dataset, and finally the TinyPower dataset. We also utilize the same approach under the ID leakage model to further assess the effectiveness of our method. Then, to emphasize the advantages of our approach, techniques such as L2 regularization, Dropout, and Gaussian noise are incorporated and compared with previous methods. Second, a scheme based on local sample correlation is utilized to attack the ASCAD_f, ASCAD_r and TinyPower datasets to



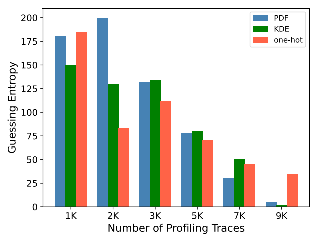
(a) MLP employing the HW leakage model



(b) MLP employing the ID leakage model

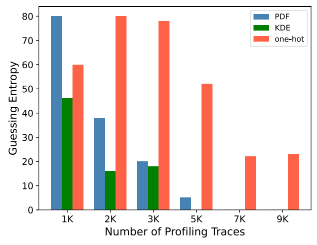


(c) CNN employing the HW leakage model

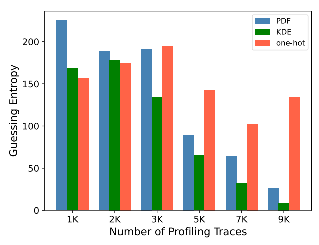


(d) CNN employing the ID leakage model

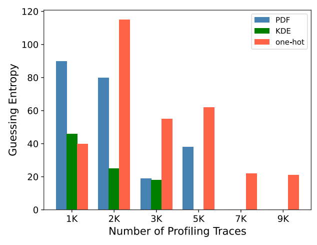
Fig. 5. Optimizing label distribution learning on the ASCAD_f dataset.



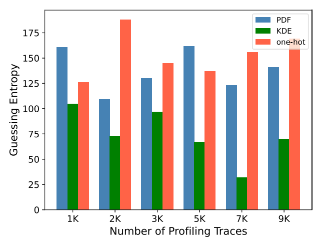
(a) MLP employing the HW leakage model



(b) MLP employing the ID leakage model



(c) CNN employing the HW leakage model



(d) CNN employing the ID leakage model

Fig. 6. Optimizing label distribution learning on the ASCAD_r dataset.

assess their attack capability, detailed in Section 4.2. Subsequently, we introduce Dropout technique to this scheme and conduct a comparative experiment with the latest attack scheme. Finally, we perform analysis experiments on the re-extracted dataset in Section 4.3.

4.1. *Label distribution learning for kernel density estimation*

We contend that kernel density estimation does not require assuming parameters in advance and can be efficiently learned even with significantly less training data in Section 3.1. We validate this idea through deep learning models by conducting training on MLP [12] and CNN [10] models with a diverse range of profiling traces and varying data volumes.

Figs. 5 and 6 respectively show the results for the ASCAD_f and ASCAD_r datasets. The traditional training method (one-hot encoded label) is denoted by 'one-hot' in the figures. The previous attack scheme using label distribution learning is represented by 'PDF'. The new method, kernel density estimation computing the distribution relationship between labels, is represented by 'KDE', with a bandwidth of 0.65 as discussed in the previous Section 3.1. Cross-entropy is employed as the loss function when one-hot encoding is used to represent the leakage relationship in the experiment. For PDF and KDE, Kullback–Leibler (KL) divergence serves as the loss function. The parameters assumed by PDF are the optimal parameters recommended in Wu's paper [17].

In Fig. 5, the distribution relationship between labels for the ASCAD_f dataset is calculated using HW kernel density estimation. With a given 5000 attack traces, both MLP and CNN models analyze fewer than 2000 traces to achieve a GE value of zero. This number is 1000 traces fewer than required when employing the probability density function, which needs 3000 traces according to literature [17]. In the case of the ID leakage model, although the GE value does not reach zero with the given 5000 attack traces, kernel density estimation facilitates a faster convergence towards the GE value.

The ASCAD_r dataset is more difficult to attack than the ASCAD_f dataset. As shown in Fig. 6, kernel density estimation significantly improves the performance of the attack compared to the probability density function. Under the HW leakage model, it takes approximately 5000 attacked traces for the MLP and CNN models to reduce the GE value to zero. This requirement is 1000 traces fewer than when using the probability density function, which needs 6000 traces [17], and it is more than 10 times less than what is required when employing the commonly used one-hot encoding method. In the ID leakage model, even though the GE value does not reach zero within the given 5000 attack traces, kernel density estimation yields a smaller GE value compared to the probability density function with the same number of traces. This indicates superior performance of our method.

Finally, similar results were obtained when attacking the TinyPower dataset. According to the literature [24], this dataset exhibits limited ID leakage, while the HW leakage model better reflects the actual leakage characteristics. Therefore, we focused solely on HW-based attacks using MLP and CNN models.

We train a specific number of datasets to achieve a guess entropy value of zero within a given number of attack traces in order to demonstrate the superiority of our proposed method.

Various methods and techniques have been developed for conducting attacks under conditions of limited power trace availability in previous studies. To better illustrate the advantages of our approach, we conducted experimental comparisons by combining our method with several commonly used techniques from previous studies. These techniques include L2 regularization, Dropout, and adding Gaussian noise. In the comparison experiments, we computed label distribution relationships using two methods: kernel density estimation and probability density function. The bandwidth used for kernel density estimation was calculated through the AMISE method, while the sigma value in the probability density function was set to 1. The total number of attack traces in our experiments was set to 10,000. We assessed the effectiveness of the attack by determining how many attack traces were required to achieve a GE value of zero. If the GE value did not reach zero, it was indicated with a '-'.

The comparison results are presented in Tables 1 and 2. Our experimental comparisons are solely focused on the HW leakage model. For each analysis, we highlight the best results in bold. In comparison with

Table 1
Benchmark the attack performance (TGE0) with MLP.

Reference	Traces	Label	ASCAD_f	ASCAD_r	TinyPower
[6]	10 000	One-hot	–	–	–
[17]		PDF	1618	3623	1745
This work		KDE	**1098**	**2707**	**1001**
[28]		Multi-label	1125	2796	1245
[29]		Smoothed	3484	–	–
[6]	10 000	One-hot	–	–	–
[17]	(L2)	PDF	–	–	4556
This work		KDE	–	–	**2234**
[28]		Multi-label	**2564**	**3021**	2315
[6]	10 000	One-hot	–	–	–
[17]	(Dropout)	PDF	2264	–	2134
This work		KDE	**893**	**2139**	**974**
[28]		Multi-label	1014	2451	1102
[6]	50 000	One-hot	1432	1771	1455
[17]		PDF	1421	919	1464
This work		KDE	864	906	**624**
[28]		Multi-label	**348**	**546**	670
[6]	50 000	One-hot	**936**	2333	1012
[17]	(Augmented)	PDF	1095	2784	1154
This work		KDE	1051	**1267**	**854**
[28]		Multi-label	1101	1816	1014

Table 2
Benchmark the attack performance (TGE0) with CNN.

Reference	Traces	Label	ASCAD_f	ASCAD_r	TinyPower
[6]	10 000	One-hot	2262	–	2156
[17]		PDF	1252	1939	1285
This work		KDE	1954	**998**	**1045**
[28]		Multi-label	**1105**	1245	1174
[29]		Smoothed	2994	–	–
[6]	10 000	One-hot	3451	–	3358
[17]	(L2)	PDF	1096	2034	1065
This work		KDE	**865**	**1589**	**846**
[28]		Multi-label	1558	2297	1148
[6]	10 000	One-hot	5932	–	5456
[17]	(Dropout)	PDF	1338	–	1368
This work		KDE	**1064**	9725	**1054**
[28]		Multi-label	1075	**2145**	1147
[6]	50 000	One-hot	1074	–	1058
[17]		PDF	779	**553**	780
This work		KDE	665	648	**540**
[28]		Multi-label	**202**	564	558
[6]	50 000	One-hot	1254	–	1124
[17]	(Augmented)	PDF	1201	2190	1024
This work		KDE	**968**	**1077**	**831**
[28]		Multi-label	1105	1541	951

the label distribution based on probability density function as proposed in Wu's paper, our proposed method, which utilizes kernel density estimation of label distribution, shows a significant performance improvement. This enhancement is evident in the reduced number of attack traces required.

When 10,000 and 50,000 profiling traces are utilized as input for the model across multiple experiments, the performance of the four label formats generally yields reasonably good results. However, during the attack phase, the approach using label distribution based on kernel density estimation necessitates fewer attack traces. In fact, using kernel density estimation is more precise than probability density function in distinguishing correct labels. Ultimately, the label distribution using kernel density estimation surpasses that of the probability density function by requiring fewer profiling and attack traces alike.

To further validate the impact of bandwidth on model performance, we conducted a sensitivity analysis on the ASCAD_f and ASCAD_r datasets. The experiments were based on the HW leakage model, with both the training set and the attack set containing 10,000 traces. A MLP model was employed for the attack. Using the computational method discussed in Section 3.1, the optimal bandwidths for kernel density estimation were determined to be 0.327 and 0.271 for the two datasets,

Table 3
Impact of different bandwidths on attack performance(TGE0) with MLP.

Dataset	H	Number of attack traces
ASCAD_f	0.1	2776
	0.327	**1084**
	0.4	1591
	0.6	2779
	0.8	2385
ASCAD_r	0.1	3149
	0.27	**2790**
	0.4	3073
	0.6	3661
	0.8	3345

respectively. The attack performance was evaluated by calculating the number of attack traces required to achieve a GE of zero. The best result for each dataset is highlighted in bold. The results are shown in Table 3.

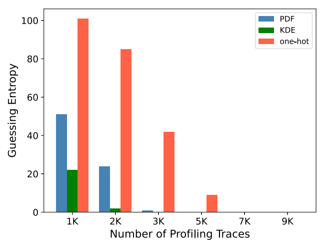
Based on the results shown in Table 3, which display the attack performance on the ASCAD_f and ASCAD_r datasets under different bandwidth settings, it is clear that the choice of bandwidth has a significant impact on attack performance. When the bandwidth is set too small, the kernel density estimate distribution curve becomes too sharp, leading the model to overfit. As a result, more traces are required to reduce the GE value to zero. On the other hand, when the bandwidth is too large, the distribution curve becomes overly smooth, which similarly degrades the attack performance. The bandwidth selected by the AMISE method outperforms other settings on both datasets. This is because AMISE balances bias and variance, effectively capturing the distribution characteristics between labels. This also confirms the theoretical effectiveness of the AMISE method in providing the optimal bandwidth.

4.2. *Sample correlations locally*

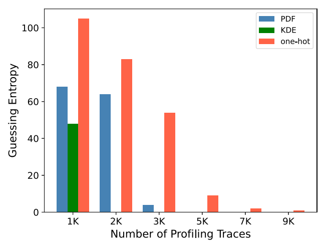
In the previous section, we suggested that the correlation between local samples implies that similar profiling traces likely share similar label distributions. This principle allows for the prediction of label distributions from similar samples, facilitating more effective learning, even with a limited number of profiling traces. The approach was first applied to MLP models in our experiments, training them with the hyperparameters from previous Label Distribution Learning (LDL) studies combined with side-channel analysis. We then extended the same methodology to CNN models, ensuring they were also trained under identical conditions. These models were exposed to various quantities of profiling traces. The results confirmed our hypothesis: employing kernel density estimation significantly enhances attack performance compared to the traditional probability density function approach. Based on this finding, we decided to solely rely on kernel density estimation for illustrating the relationship between labels in this set of experiments.

Results for the ASCAD_f, ASCAD_r and TinyPower datasets are displayed in Figs. 8–10 respectively. We apply two distinct leakage models in our analysis: the ID leakage model and the HW leakage model. Additionally, we use the KL divergence function as the loss function, which allows us to quantify the discrepancy between the actual and predicted distributions.

Fig. 8 illustrates the results obtained from attacking the ASCAD_f dataset using MLP and CNN models, respectively. Both models were trained and tested using 800 profiling traces. In Fig. 8, (a) and (b) represent the changes in GE values caused by the MLP model based on ID leakage and HW leakage, respectively. Similarly, in Fig. 7, (c) and (d) depict the evolution of GE values in the CNN model using ID leakage and HW leakage, respectively. From the figures, it is evident that the GE value can reach zero with just 400 attack traces using 800 profiling traces, regardless of whether the MLP model or the CNN

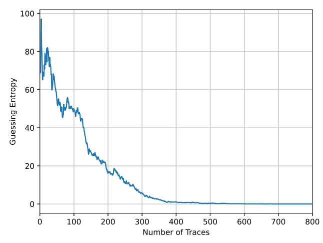


(a) MLP employing the HW leakage model

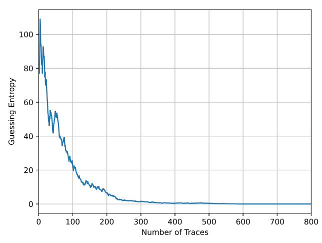


(b) CNN employing the HW leakage model

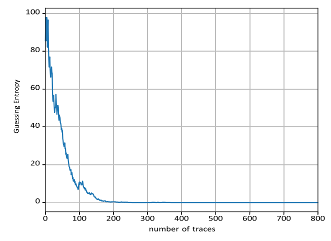
Fig. 7. Optimizing label distribution learning on the TinyPower dataset.



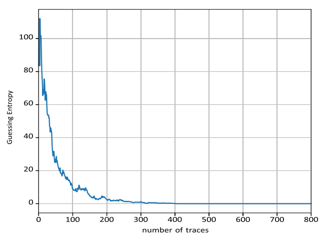
(a) MLP with the ID leakage model



(b) MLP with the HW leakage model

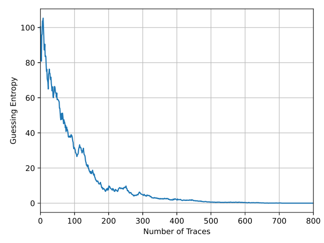


(c) CNN with the ID leakage model

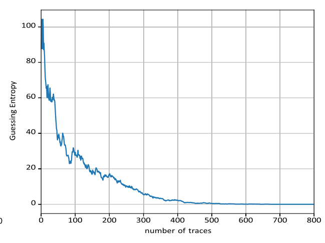


(d) CNN with the HW leakage model

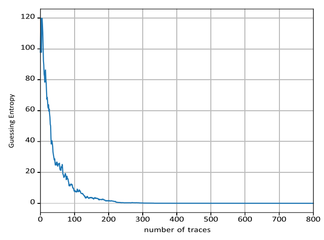
Fig. 8. Label distribution learning sample local correlation on the ASCAD_f dataset.



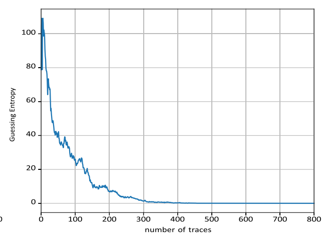
(a) MLP with the ID leakage model



(b) MLP with the HW leakage model

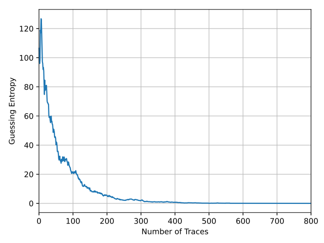


(c) CNN with the ID leakage model

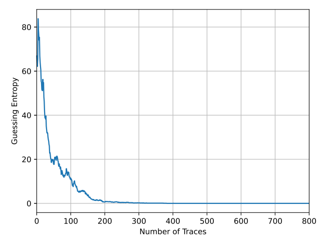


(d) CNN with the HW leakage model

Fig. 9. Label distribution learning sample local correlation on the ASCAD_r dataset.



(a) MLP with the HW leakage model



(b) CNN with the HW leakage model

Fig. 10. Label distribution learning sample local correlation on the TinyPower dataset.

model is employed. This is more than three times fewer profiling traces compared to the literature using LDL combined with SCA (3000 traces) [17]. Moreover, it is more than twice as efficient in profiling traces compared to the previously mentioned optimization of LDL using kernel density estimation combined with SCA. These results validate the faster convergence rate achieved by using LDL-SCL on the ASCAD_f dataset, demonstrating strong attack capability across various attack scenarios.

In Fig. 9, the results obtained from attacking the ASCAD_r dataset demonstrate that combining LDL-SCL with SCA significantly enhances attack performance. The MLP and CNN models require approximately 1000 profiling traces to reach GE zero for both the ID leak and HW leak models. This requirement is about six times less than the 6000 profiling traces needed when using LDL combined with SCA in previous related literature [17]. In terms of leakage models, the attack performance using the ID leakage model tends to be superior to that of the HW leakage model. We speculate that this difference arises because the ASCAD_r dataset contains 1400 power consumption points per data, which may exhibit stronger correlations. Despite the slightly reduced convergence speed compared to attacking the ASCAD dataset, both models achieve a GE value of zero within 500 tests. This indicates that the LDL-SCL combined with SCA approach exhibits robust attack capability across various attack scenarios.

As shown in Fig. 10, this method also demonstrates excellent attack performance on the TinyPower dataset. As mentioned in the previous subsection, due to the characteristics of the TinyPower dataset, the experiments were conducted using only the HW leakage model.

Table 4
Assess the attack's effectiveness (TGE0) using MLP. The outcomes for the ID and HW leakage models are divided by '/'.

Reference	Trace	Label	ASCAD_f	ASCAD_r	TinyPower
[6]	800	One-hot	–/–	–/–	–/–
[17]		LDL	–/–	–/–	–/–
This work		LDL-SCL	**315/276**	**448/423**	–/–
[28]		Multi-label	–/–	–/–	–/–
[6]	800	One-hot	–/–	–/–	–/–
[17]	Dropout	LDL	–/–	–/–	–/–
This work		LDL-SCL	**123/198**	**376/265**	–/–
[28]		Multi-label	–/–	–/–	–/–

We introduce Dropout technology for experimental comparison, based on both our proposed attack scheme and the previous one, to better illustrate the advantages. The profiling traces are set at 800, with attack performance evaluated by the number of attacks needed for the analysis trace to achieve a GE value of zero. Optimal outcomes from each analysis method are highlighted in bold. Should the GE value not drop to zero after 500 traces, the outcome is marked as '-'. The test results are presented in Tables 4 and 5. Compared to the three label formats proposed in previous studies,our attack method demonstrates significant advantages.

Table 5
Assess the attack's effectiveness (TGE0) using CNN. The outcomes for the ID and HW leakage models are divided by '/'.

Reference	Trace	Label	ASCAD_f	ASCAD_r	TinyPower
[6]	800	One-hot	–/–	–/–	–/–
[17]		LDL	–/–	–/–	–/–
This work		LDL-SCL	**189/297**	**216/312**	–/–
[28]		Multi-label	–/–	–/–	–/–
[6]	800	One-hot	–/–	–/–	–/–
[17]	Dropout	LDL	–/–	–/–	–/–
This work		LDL-SCL	**56/68**	**62/75**	–/–
[28]		Multi-label	–/–	–/–	–/–

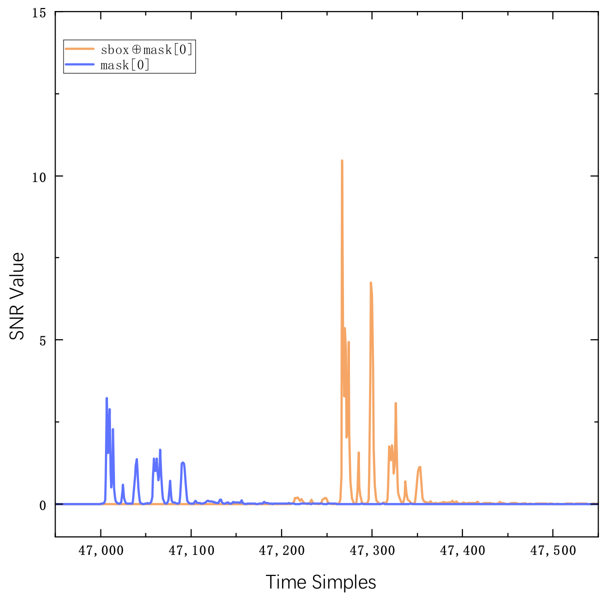


Fig. 11. Third byte SNR plot.

4.3. Re-extract datasets

In side-channel analysis, enhancing the correlation in power analysis and minimizing noise interference are critical objectives for attackers. We draw inspiration from well-known time series datasets such as UCR [30], Hong Wei-Chiang [31], Yu Ou's dataset [32], and others to curate a new dataset containing only 500 power consumption points per data. These initial profiling traces are derived from the ASCAD database.

Although ASCAD.H5 is a preprocessed dataset and can be directly used for training, we believe that there is potential for improvement. The profiling trace length in each sample is too long, and the selection of power points is not optimal. Therefore, we re-extracted the optimal power consumption points using the Signal-to-Noise Ratio. We selected profiling traces points that contain the highest point of the SNR of the third byte with the mask, as well as the highest point of the masked SNR. We kept only the points within the interval [47 000, 47 500]. This process is illustrated in Fig. 11.

We conducted experiments to test the re-extracted dataset. Only the dataset was changed, the re-extracted dataset is denoted by RE_ASCAD_f, while the original fixed dataset is denoted by ASCAD_f. We tested two sets with different numbers of profiling traces using the MLP model. The total number of attack traces is 2000. Attack effectiveness was assessed based on how many traces were required to achieve a GE value of zero. If the GE value cannot be reduced to zero after 1500 attack traces, it is marked with '-'. As illustrated in Table 6, under the re-extracted dataset, the GE value reaches zero when the number of attacked traces is 2000 using one-hot coding. This amounts to fewer than one-tenth the 50,000 traces commonly used in the literature [6]. This demonstrates the superiority of our dataset.

Table 6
MLP attack results on the re-extracted dataset are presented. The outcomes for the ID and HW leakage models are divided by '/'.

Trace	Label	RE_ASCAD_f	ASCAD_f
2000	One-hot	1282/–	–/–
	LDL	753/–	–/–
5000	One-hot	410/–	–/–
	LDL	433/–	–/–

Table 7
Comparison of memory usage and complexity across different labeling methods.

Label	Profiling traces	Memory usage	Complexity
One-hot	50 000	1.81 MB	$O(n)$
LDL	3000	1.80 MB	$O(n)$
LDL-SCL	800	2.2 MB	$O(n^2)$
Multi-label	10 000	1.85 MB	$O(n)$

5. Discussion

Firstly, the KDE method has certain limitations when applied to side-channel analysis. Specifically, selecting an appropriate bandwidth is challenging. An improper bandwidth selection may affect the estimated density values, thereby influencing the attack results. Secondly, the proposed side-channel analysis approach based on label distribution learning with local sample relevance also has a minor drawback. Specifically, it relatively increases the computational complexity.

Table 7 provides a comparison of different labeling methods in terms of memory usage and computational complexity when using different numbers of training samples. The "profiling traces" column represents the minimum number of training samples required to complete the attack. "Memory Usage" indicates the memory consumption during the training phase for each method, while "Complexity" reflects the theoretical computational complexity, expressed as a function of the input data size n. The detailed quantitative analysis is as follows:

From the table, it can be observed that although the computational complexity of the LDL-SCL attack scheme is higher than that of other methods, this is mainly due to the additional calculation of sample distribution relationships required during the training phase. In traditional side-channel attacks, such calculations are typically unnecessary. However, the LDL-SCL scheme demonstrates a significant advantage in terms of training sample requirements, achieving successful attacks with far fewer training samples. Additionally, it also shows some advantages in memory usage.

6. Conclusions and future work

In side-channel analysis, label distribution learning typically involves using the distribution relationship between labels and computing this relationship using probability density functions. This paper introduces a novel method for calculating the distributional relationship between labels, leading to improved attack performance. Furthermore, by leveraging the relationship between profiling traces, we propose local sample-dependent label distribution learning as a new attack scheme, effectively minimizing the need for numerous profiling traces. Additionally, we re-extract the dataset and perform dimensionality reduction using SNR. Our results demonstrate that enhancing the useful information of the dataset enhances the efficiency of side-channel analysis.

In future work, we aim to explore more efficient and simplified approaches for computing the distribution relationship between labels. Additionally, applying our method to non-profiling analysis could be an intriguing direction in learning label distributions for local correlations of samples. Moreover, on the dataset side, improving extraction methods would be exciting for further advancements.

CRediT authorship contribution statement

Shengcheng Xia: Conceptualization, Methodology, Software, Data curation, Writing – original draft, Visualization, Investigation. **Lang Li:** Software, Supervision, Visualization, Writing – review & editing. **Yu Ou:** Supervision, Writing – review & editing. **Jiahao Xiang:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported by the Hunan Provincial Natural Science Foundation of China (2022JJ30103), "the 14th Five-Year Plan" Key Disciplines and Application-oriented Special Disciplines of Hunan Province (Xiangjiaotong [2022] 351), the Science and Technology Innovation Program of Hunan Province (2016TP1020). Postgraduate Scientific Research Innovation Project of Hunan Province (No. CX20240977).

Data availability

No data was used for the research described in the article.

References

[1] A. Park, D. Han, Chosen ciphertext simple power analysis on software 8-bit implementation of ring-LWE encryption, in: IEEE Asian Hardware-Oriented Security and Trust, IEEE Computer Society, Yilan, Taiwan, 2016, pp. 1–6, http://dx.doi.org/10.1109/ASIANHOST.2016.7835555.

[2] T. Gellersen, O. Seker, T. Eisenbarth, Differential power analysis of the picnic signature scheme, in: Post-Quantum Cryptography, Springer, Cham, 2021, pp. 177–194, http://dx.doi.org/10.1007/978-3-030-81293-5_10.

[3] A. Chakraborty, A. Mondal, A. Srivastava, Correlation power analysis attack against STT-MRAM based cyptosystems, in: IEEE International Symposium on Hardware Oriented Security and Trust, IEEE Computer Society, Mclean, VA, 2017, p. 171, http://dx.doi.org/10.1109/HST.2017.7951835.

[4] Z. Xu, O. Pemberton, S.S. Roy, D.F. Oswald, W. Yao, Z. Zheng, Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber, IEEE Trans. Comput. 71 (9) (2022) 2163–2176, http://dx.doi.org/10.1109/TC.2021.3122997.

[5] L. Alzubaidi, J. Zhang, A.J. Humaidi, A.Q. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M.A. Fadhel, M. Al-Amidie, L. Farhan, Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, J. Big Data. 8 (1) (2021) ,53, http://dx.doi.org/10.1186/S40537-021-00444-8.

[6] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, C. Dumas, Deep learning for side-channel analysis and introduction to ASCAD database, J. Cryptogr. Eng. 10 (2) (2020) 163–188, http://dx.doi.org/10.1007/S13389-019-00220-8.

[7] G. Zaid, L. Bossuet, A. Habrard, A. Venelli, Methodology for efficient CNN architectures in profiling attacks, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020 (1) (2020) 1–36, http://dx.doi.org/10.13154/TCHES.V2020.I1.1-36.

[8] L. Ni, P. Wang, Y. Zhang, H. Zhang, X. Li, L. Ni, J. Lv, W. Zheng, Profiling side-channel attacks based on CNN model fusion, Microelectron. J. 139 (2023) 105901, http://dx.doi.org/10.1016/j.mejo.2023.105901.

[9] G. Perin, L. Chmielewski, S. Picek, Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020 (4) (2020) 337–364, http://dx.doi.org/10.13154/TCHES.V2020.I4.337-364.

[10] L. Wu, G. Perin, S. Picek, I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis, IEEE Trans. Emerg. Top. Comput. (2024) 1–12., http://dx.doi.org/10.1109/tetc.2022.3218372.

[11] S. Hajra, S. Chowdhury, D. Mukhopadhyay, Stranet: An efficient shift-InvariantTransformer network for side-channel analysis, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2024 (1) (2023) 336–374, http://dx.doi.org/10.46586/tches.v2024.i1.33.

[12] J. Rijsdijk, L. Wu, G. Perin, S. Picek, Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis, IACR Trans. Cryptogr. Hardw. Embed. Syst. 2021 (3) (2021) 677–707, http://dx.doi.org/10.46586/TCHES.V2021.I3.677-707.

[13] O. Choudary, M.G. Kuhn, Template attacks on different devices, in: Constructive Side-Channel Analysis and Secure Design, Vol. 8622, Springer, Cham, 2014, pp. 179–198, http://dx.doi.org/10.1007/978-3-319-10175-0_{1}{3}.

[14] Z. Luo, M. Zheng, P. Wang, M. Jin, J. Zhang, H. Hu, N. Yu, Towards strengthening deep learning-based side channel attacks with mixup, in: IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom, IEEE, Shenyang, China, 2021, pp. 791–801, http://dx.doi.org/10.1109/TRUSTCOM53373.2021.00114.

[15] W. Wang, J.N. Wang, F.L. Hu, F. Ni, SCA-CGAN:A new side-channel attack method for imbalanced small samples, Radioeng. 32 (1) (2023) 124–135, http://dx.doi.org/10.13164/re.2023.0124.

[16] Di Li, L. Li, Y. Ou, Side-channel analysis based on siamese neural network, J. Supercomput. 80 (4) (2024) 4423–4450, http://dx.doi.org/10.1007/S11227-023-05631-3.

[17] L. Wu, L. Weissbart, M. Krcek, H. Li, G. Perin, L. Batina, S. Picek, Label correlation in deep learning-based side-channel analysis, IEEE Trans. Inf. Forensics Secur. 18 (2023) 3849–3861, http://dx.doi.org/10.1109/TIFS.2023.3287728.

[18] S. aw Węglarczyk, Kernel density estimation and its application, ITM Web Conf. 23 (2018) 00037, http://dx.doi.org/10.1051/itmconf/20182300037.

[19] Y.-C. Chen, A tutorial on kernel density estimation and recent advances, Biostat. Epidemiol. 1 (1) (2017) 161–187, http://dx.doi.org/10.1080/24709360.2017.1396742.

[20] J. Yin, T. Li, H. Shen, Gaussian kernel optimization: Complex problem and a simple solution, Neurocomputing 74 (18) (2011) 3816–3822, http://dx.doi.org/10.1016/J.NEUCOM.2011.07.017.

[21] X. Zheng, X. Jia, W. Li, Label distribution learning by exploiting sample correlations locally, in: AAAI Conference on Artificial Intelligence, AAAI Press, New Orleans, Louisiana, USA, 2018, pp. 4556–4563, http://dx.doi.org/10.1609/AAAI.V32I1.11693.

[22] C.-K. Ngai, R.W. Yeung, Z. Zhang, Network generalized hamming weight, IEEE Trans. Inf. Theory. 57 (2) (2011) 1136–1143, http://dx.doi.org/10.1109/TIT.2010.2095233.

[23] I. Buhan, L. Batina, Y. Yarom, P. Schaumont, SoK: Design tools for side-channel-aware implementations, in: Asia Conference on Computer and Communications Security, ACM, New York, NY, USA, 2022, pp. 756–770, http://dx.doi.org/10.1145/3488932.3517415.

[24] H. Li, M. Ninan, B. Wang, J.M. Emmert, TinyPower: Side-channel attacks with tiny neural networks, in: IEEE International Symposium on Hardware Oriented Security and Trust, HOST, Tysons Corner,USA, 2024, pp. 320–331, http://dx.doi.org/10.1109/HOST55342.2024.10545382.

[25] G. Igarashi, Y. Kakizawa, Generalised gamma kernel density estimation for nonnegative data and its bias reduction, J. Nonparametric Stat. 30 (3) (2018) 598–639, http://dx.doi.org/10.1080/10485252.2018.1457791.

[26] F. Nielsen, The Kullback–Leibler divergence between lattice Gaussian distributions, J. Indian Inst. Sci. 102 (4) (2022) 1177–1188, http://dx.doi.org/10.1007/s41745-021-00279-5.

[27] A. Seal, A. Karlekar, O. Krejcar, C. Gonzalo-Martin, Fuzzy c-means clustering using Jeffreys-divergence based similarity measure, Appl. Soft Comput. 88 (2020) ,106016, http://dx.doi.org/10.1016/j.asoc.2019.106016.

[28] L. Zhang, X. Xing, J. Fan, Z. Wang, S. Wang, Multilabel deep learning-based side-channel attack, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (6) (2021) 1207–1216, http://dx.doi.org/10.1109/TCAD.2020.3033495.

[29] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, C. Dumas, Deep learning for side-channel analysis and introduction to ascad database, J. Cryptogr. Eng. 10 (2020) 163–188, http://dx.doi.org/10.1007/s13389-019-00220-8.

[30] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The UCR time series archive, IEEE/ CAA J. Autom. Sin. 6 (6) (2019) 1293–1305, http://dx.doi.org/10.1109/jas.2019.1911747.

[31] Z. Zhang, W.-C. Hong, Application of variational mode decomposition and chaotic grey wolf optimizer with support vector regression for forecasting electric loads, Knowledge- Based Syst. 228 (2021) ,107297, http://dx.doi.org/10.1016/j.knosys.2021.107297.

[32] Y. Ou, L. Li, Di Li, J. Zhang, ESRM: an efficient regression model based on random kernels for side channel analysis, Int. J. Mach. Learn. Cybern. 13 (10) (2022) 3199–3209, http://dx.doi.org/10.1007/S13042-022-01588-6.