SCA-CGAN: A New Side-Channel Attack Method for Imbalanced Small Samples

Wan WANG [1,2], Jun-Nian WANG [1,2], Fan-Liang HU [1,2], Feng NI [1,2]*

[1] School of Physics and Electronics, Hunan University of Science and Technology, Xiangtan, China
[2] Hunan Province Key Laboratory of Intelligent Sensors and Advanced Sensing Materials, Xiangtan, China

jnwang@hnust.edu.cn

Submitted March 20, 2022 / Accepted November 22, 2022 / Online first February 27, 2023

Abstract. *In recent years, many deep learning and machine learning based side channel analysis (SCA) techniques have been proposed, most of which are based on the optimization of existing network models to improve the performance of SCA. However, in practice, the attacker often captures unbalanced and small samples of data due to various environmental factors that limit and interfere with the successful implementation of SCA. To address this problem, in this paper, we firstly introduced the Conditional Generation Adversarial Network (CGAN). We proposed a new model SCA-CGAN that combines SCA and CGAN. We used it to generate a specified number and class of simulated energy traces to expand and augment the original energy traces. Finally, we used the augmented data to implement SCA and achieved a good result. Through experiments on the unprotected ChipWhisperer (CW) data and the ASCAD jittered dataset, the results shown that the SCA using the augmented data is the most efficient, and the correct key is successfully recovered on both datasets. For the CW dataset, the model accuracy is improved by 20.75% and the traces number required to recover the correct key is reduced by about 79.5%. For the ASCAD jittered dataset, when the jitter is 0 and 50, the traces number required to recover the correct key is reduced by about 76.8% and 75.7% respectively.*

Keywords

Deep learning side channel analysis, SCA-CGAN, unbalanced small samples, data augmentation

1. Introduction

SCA is a cryptanalytic attack that exploits the physical environment of an encrypted system to recover some information about its secrets [1]. Realistic cryptographic devices inevitably leak some physical information, such as electromagnetic radiation [2], power consumption [3], and time consumption during encryption and decryption [4], in the process of running encryption algorithms, and all these leaked physical information are related to the computation of secret information during encryption and decryption, which can be exploited by attackers to recover secret information. So far, numerous works on SCAs are based on these leakages' information [5–8].

SCA was originally proposed [9] by Paul Kocher in 1996, who used the method to perform a timing attack on encrypted devices and successfully recovered secret information, and since then, countries around the world have developed research plans for SCA, defense, evaluation, and application, and the research on SCA theories and methods have achieved fruitful results. With the further development of SCA, Machine Learning (ML) based SCAs [10], [11] have emerged, where the two most commonly used machine learning models are Support Vector Machine (SVM) [12] and Random Forest (RF) [13]. In recent years, due to the rapid development of Deep Learning (DL) [14], [15] techniques, deep learning techniques for SCAs have been strongly developed and many excellent works have emerged [16–18]. Compared with the classical Template Attack (TA) [19], deep learning-based side channel attacks (DLSCAs) generally do not require finding Points of Interest (POIs) or performing energy trace curve alignment work, and in addition, it can map high-dimensional data into a low-dimensional vector matrix space by learning from deep learning networks. However, almost all of the above works focus on how to propose better model algorithms or improve existing ones in order to reduce the number of energy traces required for a successful attack to recover the key. In fact, the number of energy traces used in the profiling phase of these works is sufficient to ensure that the training and testing of deep learning models is properly and reasonably done. In theory, this makes sense, since an important prerequisite for a SCA in the profiling phase is that the attacker has a fully controllable cryptographic device and access to enough energy traces to build a SCA model, but, thinking in terms of hardware security defense, a major factor in implementing a successful SCA is the quality and quantity of the leaked information obtained during the profiling phase. Generally, embedded engineers and security analysts strive to improve defensive countermeasures at the hardware and software levels to minimize side-channel data leakage from encrypted hardware. Therefore, data collection during the profiling phase

DOI: 10.13164/re.2023.0124

plays an important role in the execution of a successful and robust SCA.

In reality, due to the environmental restrictions and interference, which lead to the attackers often collected unbalanced small sample energy traces [20], [21], in this situation, the SCA using traditional deep learning algorithms will be very ineffective, and cannot even implement the SCA successfully. To address this problem, in this paper, we propose a SCA method based on conditional adversarial generative networks to address the above problems and successfully implemented the SCA, and the results show that excellent performance on both CW dataset and ASCAD dataset. Specifically, in the experiments, we first collected the CW dataset through the ChipWhisperer platform, and then obtained unbalanced small samples through data preprocessing. To enhance the original data, we constructed a SCA-CGAN model with Hamming weight labels for conditioning. Then, we used the pre-trained SCA-CGAN model to generate simulated data with specified classes and specified quantities, and mixed the generated simulated data with the original data to obtain the class balanced samples at last. After finish the data augmentation task, we further investigated the classification experiments with Hamming weight labels to evaluate the performance for the augmented dataset by the result of the guess entropy. In the experiments, we have set up four sets of experimental cases to quantitatively investigate the effect of the SCA based on SCA-CGAN model. In addition, we also evaluated the quality of the simulated data by appearance shape, leakage information correlation, feature clustering, and classification performance.

Our contributions are as follows:

1. There is still no better solution to the problem of SCA for unbalanced small samples, while this paper uses the SCA-CGAN model to achieve the purpose of data enhancement and successfully implements the SCA.

2. In this paper, under the external condition of Hamming weight, the SCA-CGAN model can automatically generate simulated data of specified class and quantity for the expansion and augmentation of the original data, which addresses the problem of original unbalanced small samples.

3. In this paper, we found that the SCA technique based on SCA-CGAN is also a threat to the first-order mask and jitter strategy by studying the experiment of SCA-CGAN model for ASCAD dataset, which can successfully recover the correct key.

2. Deep Learning Based Profiled SCA

2.1 Profiled SCA

Profiled side-channel analysis [22] is considered to be the most powerful SCA. Profiled SCA assumes that the attacker has a programmable cryptographic device that is identical to the target cryptographic device, and the attackers have all the privileges of this device to extract the physical leakage information of the target cryptographic chip precisely through this device. This type of attack mainly includes Template Attack (TA) and Random Model Analysis (RMA) (e.g. Linear Regression Analysis). The process of general profiled side-channel analysis is divided into two phases:

(1) Profiling Phase

The attacker has a programmable encryption device identical to the target encryption device and has all the privileges of this device. The attacker uses the known plaintext set $\mathbf{P}_{\text{profiling}} = \{\mathbf{p}_i \mid i = 1,2, ..., N_\text{p}\}$ and a fixed key k^* to encrypt this cryptographic device and obtain a total of N_p power consumption leakage data $\mathbf{T}_{\text{profiling}} = \{\mathbf{t}_i \mid i = 1, 2, \ldots, N_\text{p}\}$, where $\mathbf{t}_i$ denotes the i-th power consumption trace vector. Assume $\mathbf{v}_i = \text{g}\ (\mathbf{p}_i, k_i^*)$ is a random intermediate value variable to represent the intermediate value of the encryption operation when the i-th power trace $\mathbf{t}_i$ corresponds to the fixed key k_i^* with known plaintext $\mathbf{p}_i$. Thus, the attacker gets the profiled model $\{\mathbf{t}_i, \boldsymbol{v}_i\}_{i=1,2,\ldots,N_\text{p}}$ for power consumption traces and intermediate values.

(2) Attack Phase

The attacker collects N_a new power consumption traces from the actual target device (which is structurally identical to the encryption device in the analysis phase), denoted by the set $\mathbf{T}_{\text{attack}} = \{\mathbf{t}_i \mid i = 1, 2, ..., N_\text{a}\}$. $\mathbf{T}_{\text{attack}}$ and $\mathbf{T}_{\text{profiling}}$ are independent of each other, and the key corresponding to each power consumption trace is k_a^*, which is fixed and unknown. To recover k_a^*, it is first necessary to compute all intermediate values for all possible guess keys $k_{\text{guess}} \in \mathbf{K}$ and the corresponding plaintexts, and then compute the posterior probability of each power consumption trace $\mathbf{t}_i$ and the corresponding intermediate values according to Bayes' theorem as follows:

$$\mathbf{Pr}\left[g\left(\mathbf{p}_i, k_{\text{guess}}\right)\middle|\mathbf{t}_i\right] = \frac{\mathbf{Pr}\left[\mathbf{t}_i\middle|g\left(\mathbf{p}_i, k_{\text{guess}}\right)\right]\cdot\mathbf{Pr}\left[g\left(\mathbf{p}_i, k_{\text{guess}}\right)\right]}{\mathbf{Pr}\left[\mathbf{t}_i\right]}. \tag{1}$$

The strategy for (1) according to the maximum likelihood function is calculated as follows:

$$k_{\text{real}} = \underset{k_{\text{guess}}\in\text{K}}{\text{argmax}} \prod_{i=1}^{N_\text{a}} \mathbf{Pr}[g(\mathbf{p}_i, k_{\text{guess}}) \mid \mathbf{t}_i]. \tag{2}$$

The result of the calculation of (2) is the guessed key corresponding to the maximum of all posterior probabilities, which is the actual real key.

2.2 Metrics

In this paper, the results of the experiments on SCA will be evaluated in the following aspects:

(1) Accuracy and Loss

The accuracy of a model is one of the most commonly

used evaluation metrics in ML to characterize the model's ability to classify data. The precision of the model refers to the probability of the model achieving the correct classification result on the validation set, which can also be understood as the ratio of the number of power traces when the guess key is equal to the correct key to the number of power traces in all validation sets. The accuracy of the general model can be defined as:

$$\mathrm{acc}(\mathbf{T}_{\mathrm{attack}}) = \frac{\left|\{\mathbf{t}_i | k_{\mathrm{real}} = \underset{k_{\mathrm{guess}} \in \mathbf{K}}{\mathrm{argmax}} \prod_{i=1}^{N_a} \mathbf{Pr}[g(\mathbf{p}_i, k_{\mathrm{guess}})]\}\right|}{\left|\mathbf{T}_{\mathrm{attack}}\right|}. \quad (3)$$

The improvement in the accuracy of the model indicates that the back propagation algorithm further optimizes the weights and bias parameters of the whole network model.

(2) Guessing Entropy (GE) [23]

GE is one of the common SCA metrics. In the implementation of DLSCA, assuming that the model to be evaluated is **M**, a score matrix on **M** can be calculated based on the power consumption dataset and the corresponding plaintext matrix as follows:

$$\mathbf{S}_{N_a}[k_{\mathrm{guess}}] = \prod_{i=1}^{N_a} \mathbf{y}_i[k_{\mathrm{guess}}]. \quad (4)$$

In (4), $\mathbf{y}_i[k_{\mathrm{guess}}]$ is the score of the prediction result of model **M** for guessing key k_{guess}. The rank function of model **M** is calculated as follows:

$$\mathrm{rank}\left(\mathbf{M}, \mathbf{T}_{\mathrm{train}}, \mathbf{T}_{\mathrm{test}}, n\right) = \left|\left\{k_{\mathrm{guess}} \mid \mathbf{S}_n\left[k_{\mathrm{guess}}\right] > \mathbf{S}_n\left[k^*\right]\right\}\right|. \quad (5)$$

In (5), $\mathbf{T}_{\mathrm{train}}$, $\mathbf{T}_{\mathrm{test}}$ are the training set and test set respectively, n is the number of power traces in the training set. When the score of (4) is the highest, the corresponding rank is zero, at which time the key k^* is the correct key k_{real}. In DLSCA, under the fixed size dataset, in order to eliminate the influence of the division of training set and test set on rank, the mean rank method is usually used, so the GE expression is obtained as follows:

$$\mathrm{GE}(n) = \mathrm{E}[\mathrm{rank}(\mathbf{M}, \mathbf{T}_{\mathrm{train}}, \mathbf{T}_{\mathrm{test}}, n)]. \quad (6)$$

Briefly, the GE is the mean rank of the correct key in multiple experiments. In the experiments of this paper, the generated new energy traces and the original energy traces are randomly divided into training and testing sets, and then the results of 50 experiments are averaged to obtain the final guess entropy results.

2.3 Introduction of CGAN

Generative Adversarial Network (GAN) is an unsupervised learning model. The classical GAN network consists of two parts, Generator and Discriminator. The role of the generator is to map the input random noise into simulated data with a similar distribution to the real data by learning the feature distribution of the real data. The discriminator aims to discriminate whether the input data is real data or simulated data generated by the generator. The generator generates simulated data satisfying the P_{g} distribution by simulating the feature distribution P_{data} of the real data with a priori distribution P_{noise}. The input of the discriminator is a mixture of real and simulated data, and the output represents the accuracy of the classification of the realness of the mixed data. During the model training, the generator and the discriminator are always playing against each other until the discriminator cannot determine whether the input data is real or fake, at which time the generator and the discriminator reach a balanced state and the whole network model is optimal. The objective function of GAN can be expressed as follows:

$$\min_{\mathbf{G}} \max_{\mathbf{D}} \mathrm{V}(\mathbf{D}, \mathbf{G}) = \mathrm{E}_{\mathbf{x} \sim P_{\mathrm{data}}}\left[\log \mathbf{D}(\mathbf{x})\right] + \mathrm{E}_{\mathbf{noise} \sim P_{\mathrm{noise}}}\left[\log\left(1 - \mathbf{D}(\mathbf{G}(\mathbf{noise}))\right)\right]. \quad (7)$$

In (7), P_{data} denotes the distribution of real data and P_{noise} denotes the distribution of random noise. When training the discriminator, the optimization objective of the discriminator model is to distinguish the original data from the simulated data as much as possible, specifically, to make the discriminant of the original data as close to 1 as possible, which means to increase the accuracy of **D**(**x**) as real, and at the same time, the discriminator should make the discriminant of the generated simulated data as close to 0 as possible, which means to reduce the accuracy of **D**(**G**(**noise**)) as real. When training the generator **G**, the optimization goal of the generator model is to improve the truthfulness of the generated simulated data to deceive the discriminator as much as possible, which means the generator should improve the accuracy of **D**(**G**(**noise**)) result as real as much as possible. The optimal discriminator can be calculated from (7) as (8). From (8), we can see that when $P_{\mathrm{data}} = P_{\mathrm{noise}}$, the discriminator cannot distinguish whether the input mixed data is real or fake, and the discriminator and the generator reach the Nash equilibrium state at this time, and the discriminant accuracy of the discriminator is 0.5.

$$\mathbf{D}(x) = \frac{P_{\mathrm{data}}}{P_{\mathrm{data}} + P_{\mathrm{noise}}}. \quad (8)$$

In GAN, there is no control over the pattern of the data to be generated, while CGAN is based on GAN with the addition of the control of external condition **Y** to generate simulated data under specified conditions. When using CGAN to generate simulated data, it is necessary to first add external constraints **Y** to the original data and noise data respectively, and then feed the data with constraints to the discriminator and generator respectively, and then train the discriminator and generator until the mutual confrontation process of the discriminator and generator reaches a dynamic equilibrium state, at which time the discriminator cannot judge the reality or fake of the input data.

The principle, structure and training process of CGAN are very similar to GAN, but the objective function is slightly different:

$$\min_{\mathbf{G}}\max_{\mathbf{D}} \mathrm{V}(\mathbf{D},\mathbf{G}) = \mathrm{E}_{\mathbf{x}\sim P_{\mathrm{data}}}\left[\log \mathbf{D}(\mathbf{x}|\mathbf{Y})\right] + \mathrm{E}_{\mathbf{noise}\sim P_{\mathrm{noise}}}\left[\log\left(1-\mathbf{D}(\mathbf{G}(\mathbf{noise}|\mathbf{Y}))\right)\right]. \quad (9)$$

Unlike (7), in (9), the external condition **Y** are added to both the discriminator and the generator, becoming **D(x | Y)** and **G (noise | *Y*)**.

2.4 SCA-CGAN Model

In this paper, considering the unbalanced small sample data and the training of the SCA-CGAN model as well as the complexity of data generation, the Hamming weight model (HW model) is used. There are a total of 9 classes of labels for the data under this energy model, and the corresponding SCA-CGAN external constraint **Y** is set to the HW label, which has a total of 9 classes.

The basic structure of SCA-CGAN is shown in Fig. 1 where both generator and discriminator use MLP as the basic architecture. The P_{noise} of random noise satisfies the normal distribution, and it is an $n \times 100$ dimensional vector, combined with the condition **Y** as the input data of the generator, and finally generated the simulated data by using the generator, and P_{g} is the corresponding data distribution function. The input of the discriminator is a mixture of the original data and the simulated data and the external condition variable **Y**. Under the joint action of these input data, the discriminator will discriminate the authenticity of the input data, and then give feedback to the generator and the discriminator to complete the parameter fine-tuning and the next training, when the simulated data generated by the generator achieves the effect of falsity and the discriminator cannot distinguish the authenticity of the input data. The whole SCA-CGAN model reaches the optimum.

2.5 The Workflow of SCA-CGAN

The workflow of SCA-CGAN is shown in Fig. 2. It contains four main phases: data pre-processing, generation of simulated data, generation of data quality assessment, and finally classifier training and testing. The specific operational information is as follows:

(1) Data Pre-processing Phase

Data preprocessing is the core of almost all deep learning tasks. In this phase, the main operations are as follows:

- Using CPA technology to find points of interest (POIs) on the original dataset and to reduce the dimension. The dimensionality reduction operation will greatly reduce the complexity of the model calculation, and improve the training speed of the model.
- Classifying the reduced-dimensional dataset by HW labels and determining the distribution of the number of each class of unbalanced data.

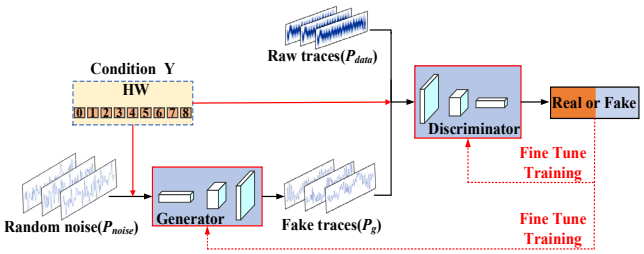


Fig. 1. The structure of SCA-CGAN.

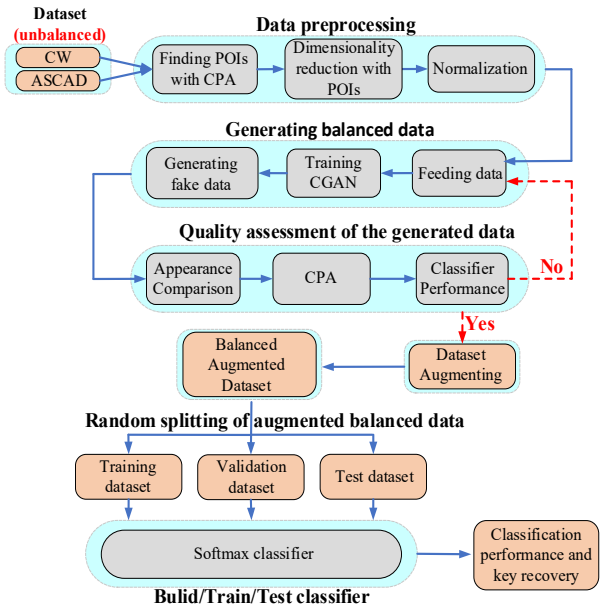


Fig. 2. SCA workflow of SCA-CGAN model.

- Normalizing data, which can speed up gradient descent of the model, which can improve the efficiency and the accuracy of model.

(2) Generating Simulated Data Phase

The final classification performance of the deep learning model is largely influenced by the balance of the input data, so it is important to ensure the balance of the classes of data in the deep learning task. The main operations in this phase are as follows:

- Feeding the pre-processed data in phase (1) into the SCA-CGAN model and setting the external condition **Y** to the HW labels.
- Iterative training of the SCA-CGAN model.
- Using a pre-trained SCA-CGAN model to generate a specified class and number of simulated power consumption traces, ensuring a balanced and sufficient number of hybrid data classes for the expansion enhancement.

(3) Simulated Data Quality Assessment Phase

After generating a certain amount of side channel energy data, the quality of the generated data needs to be further evaluated in the following steps:

- Visualizing the simulated data and the original data and observing the appearance and shape of the two sets of data.
- Pearson correlation coefficient of the leakage information in the leakage interval for the original traces and simulated traces respectively.
- The performance of the classification of the generated simulated data is evaluated using the classifier model, and if it does not satisfy the requirements, it returns to phase (2) and the SCA-CGAN model is fine-tuned with the corresponding hyper-parameters, and then the SCA-CGAN model is retrained to generate simulated energy traces and continue with the quality evaluation in this phase.

(4) Training and Testing Classifier Model Phase

After phase (3) is completed, the simulated data are merged with the original data to obtain an expanded and augmented balanced data set, which is randomly divided into a training set, a validation set, and a test set. After all the data are prepared is the classifier model building, training and testing phase, in which the structure of the classifier model is determined according to the data characteristics of the augmented dataset, then the model is trained and validated using the training set and validation set, and finally the trained classifier is tested using the test set. The experiments will calculate the GE of the correct key from the testing results of the classifier model to determine how many numbers of energy traces are needed to retrieve the correct key.

In this paper, all classifier models are composed of 6 fully connected layers with a classification number of 9. The loss function is *multi-classification cross-entropy*, and the corresponding output layer activation function is *Softmax*.

3. Experimental Setups and Datasets

3.1 Experimental Setups

In this paper, the details of the environment setup related to the capture of ChipWhisperer (CW) data are shown in Tab. 1. The cryptographic platform in the experiment is ChipWhisperer [24], the target cryptographic board is CW308T-STM32F3, the cryptographic chip is 32-bit Arm Cortex-M4, the cryptographic algorithm is TinyAES-128C, and the encryption mode is ECB. The data in the experiment is captured through the components of ChipWhisperer's Capture board with a sampling frequency of 40 MHz.

Experimental Configuration	Detailed Information
Encrypted Platform	ChipWhisperer
Target Board	CW308T-STM32 F3
Target Chip	32-bit Arm Cortex-M4
Target Encryption Algorithm	TinyAES-128C
Encrypted Mode	ECB
Programming Languages	Python 3.6
Deep Learning Framework	Keras_gpu 2.3.1 + Tensorflow_gpu 2.1.0

Tab. 1. Experimental configuration details.

3.2 The POIs of CW Dataset

In the implementation of SCA, it is first necessary to simulate the energy consumption of the cryptographic device when running the encryption algorithm. There are generally three main energy models, respectively, the identity model (ID model), the Hamming distance model (HD model) and the Hamming weight model (HW model), and the labels of the power consumption data are different for different energy consumption models. In this paper, the HW model is chosen, and all the data corresponding to the labels are the HW (the number of 1 contained in the 8-bit binary data) of the target byte, and there are 9 classes in total.

In this paper, the target encryption algorithm is AES-128, which consists of a total of 10 rounds of encryption operations, each with a round key length of 128 bits (16 bytes). In AES-128, the only nonlinear transformation is the byte substitution operation of the S-box, which can guarantee the security of the whole encryption algorithm significantly.

The labels of all models trained in the experiments are set to the output state of the S-box in the first round of encryption operations, denoted as:

$$\mathbf{state}_i = \mathrm{Sbox}\left(\mathbf{p}_i \oplus \mathbf{k}_i\right). \tag{10}$$

In (10), $\oplus$ indicates the XOR operation, $\mathbf{p}_i$ and $\mathbf{k}_i$ respectively represent the i-th byte of the plaintext and the i-th byte of the initial key, and $\mathbf{state}_i$ indicates the states (e.g. labels) after the output of S-box. The reason for setting the labels in this way is that when the target encryption chip runs the encryption algorithm, it needs to invoke the S-box from the internal registers of the chip to perform the SubBytes operation firstly, and then load the intermediate states after the operation to the data bus, however, the capacitive load on the data bus is usually large, which corresponds to the maximum energy consumption of the S-box operation. In this paper, for all 16 bytes of a set of keys, the attack strategy is to decrypt each byte one by one and finally achieve the recovery of all key byte.

After determining the location of the target attack point need to find the corresponding interest interval, generally use CPA techniques to find the interest interval of the target byte, the main analysis steps are as follows:

1) Capture original energy data and align all data.
2) Calculate the matrix of intermediate values. Use public information (e.g., plaintext or ciphertext) and all possible keys (guessing keys) for SubBytes.
3) To find the interval of interest. Use CPA to calculate the Pearson Correlation between the intermediate value matrix and the original energy data, and select the interval with the largest correlation coefficient.

4) Data dimensionality reduction. Based on the interest interval found, all the original energy data are downscaled, and only the energy data information within the interest interval is retained.

3.3 CW Dataset

A total of 60,000 side channel power traces were collected in the experiments, and each power trace contains 3,000 sampling points. The encryption device uses random plaintexts and fixed keys for the entire encryption process. The visualization results of the original energy trace curves and the POIs are shown in Fig. 3, where Figure 3(a) shows the traces curve of all operations in the first round of encryption, and Figure 3(b) shows a zoomed-in view of the traces curve for the first round of SubBytes operations, and Figure 3(c) shows the POIs corresponding to the power consumption traces, where the range marked by the dashed line is the interval of interest of the target byte.

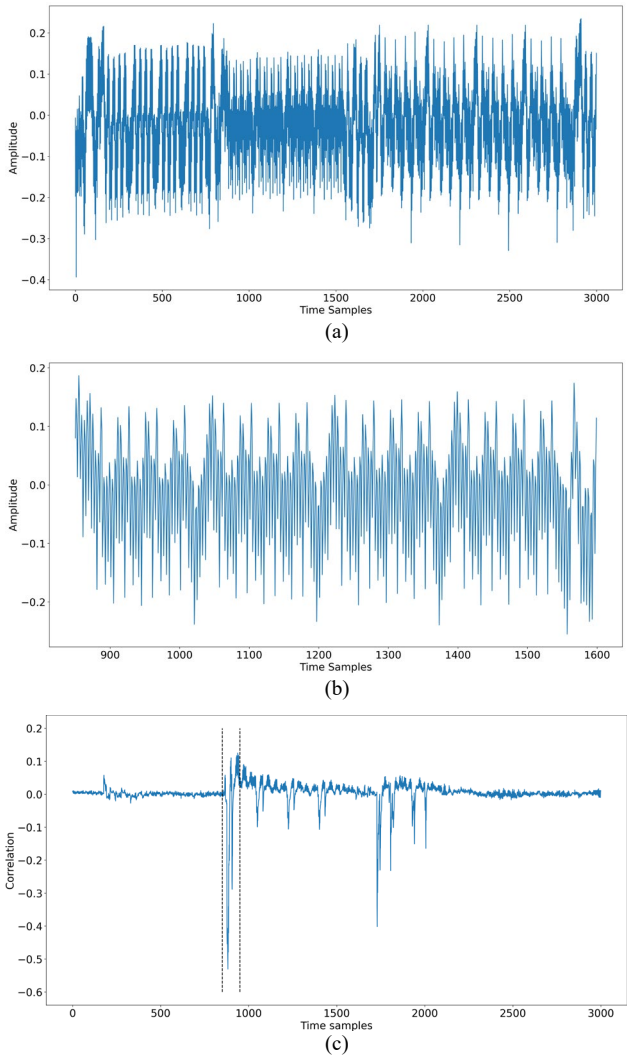


Fig. 3. (a) Single energy trace; (b) Zoomed in S-box energy trace; (c) The result of CPA.

3.4 ASCAD Dataset

The dataset provided in the ASCAD database is extracted from the ATMega8515_raw_traces.h5 file containing the raw traces. In order to avoid useless and extensive data processing, it only retains the 700 feature points from 45,400 to 46,100, which correspond to the third byte of the output of the S-box operation of the first round of AES-128 encryption. The ASCAD database contains mainly the following three datasets in HDF5 file format:

1) ASCAD.h5, which contains traces that are synchronized and without jitter.

2) ASCAD_desync50.h5, which contains traces with a 50 samples window maximum jitter.

3) ASCAD_desync100.h5, which contains traces with a 100 samples window maximum jitter.

The datasets contained in the above three files are divided into two groups, one is the profiling datasets, which contains a total of 50,000 energy traces, and the other is the attack datasets, which contains a total of 10,000 energy traces. For the ASCAD dataset, the targeted attack point is the third byte state of the S-box output of the first round of encryption, and the corresponding label is set as $\mathrm{Sbox}(\mathbf{P}_3 \oplus \mathbf{K}_3)$.

After extracting all the data sets from ASCAD dataset, the same pre-processing operation is done for each set of data, and the main steps are as follows:

1) Making unbalanced small samples. All data are classified according to HW labels, and then a few energy traces are randomly extracted from each class and combined to obtain the unbalanced small sample dataset, which is a total of 2000 data in the experiments of this paper.

2) Normalizing the unbalanced small samples.

4. The Experimental Results and Analysis

4.1 Experimental Protocols

In this paper, we design the same comparison experiments for both CW data and ASCAD dataset with defense strategies, and the specific experimental protocols are shown in Tab. 2.

All experiments use first-order CPA to find the POIs in the corresponding dataset. In addition, for the ASCAD dataset, the role of mask is not considered in the experimental profiling phase, which is because the attacker does not know the existence of mask in the actual SCA.

Experimental Protocols	Original traces	Generated traces	Augmented traces	Training set: Test set
A	1000	0	1000	4:1
B	1500	0	1500	4:1
C	1500	1500	3000	4:1
D	1500	1500 × 2	4500	4:1

Tab. 2. All experimental protocols.

Layer(type)	Output Shape	Param #
Dense	(None, 100)	10100
LeakyReLU	(None, 100)	0
Dense	(None, 256)	25856
LeakyReLU	(None, 256)	0
BatchNormalization	(None, 256)	1024
Dense	(None, 512)	131584
LeakyReLU	(None, 512)	0
BatchNormalization	(None, 512)	2048
Dense	(None, 512)	262656
LeakyReLU	(None, 512)	0
BatchNormalization	(None, 512)	2048
Dense	(None, 256)	131328
LeakyReLU	(None, 256)	0
BatchNormalization	(None, 256)	1024
Dense	(None, 100)	25700
Total params:		593,368

Tab. 3. Structure of the Generator.

Layer(type)	Output Shape	Param #
Dense	(None, 100)	10100
LeakyReLU	(None, 100)	0
Dense	(None, 512)	51712
LeakyReLU	(None, 512)	0
Dropout	(None, 512)	0
Dense	(None, 512)	262656
LeakyReLU	(None, 512)	0
Dropout	(None, 512)	0
Dense	(None, 512)	262656
LeakyReLU	(None, 512)	0
Dropout	(None, 512)	0
Dense	(None, 512)	262656
LeakyReLU	(None, 512)	0
Dropout	(None, 512)	0
Dense	(None, 1)	513
Dense	(None, 9)	4617

Tab. 4. Structure of the Discriminator.

Optimizer	Adam
Learning rate	0.0002
Loss function	Cross-Entropy
Mini_batch	256
Epochs	5000

Tab. 5. Hyperparameter setting.

4.2 The Experimental Results and Analysis for CW Dataset

4.2.1 Training

The structures and parameters of the SCA-CGAN model in the experiments for CW dataset are shown in Tab. 3 and Tab. 4, and the information on the hyperparameter settings for model training and testing is shown in Tab. 5. For the ASCAD dataset, the structure of the SCA-CGAN model is the same, except that the number of input data features.

4.2.2 Testing

After training the SCA-CGAN model, for experiments A, B, C and D, the corresponding number of simulated power consumption traces were generated with the generator in SCA-CGAN, and finally the simulated traces and the original traces were visualized, as shown in Fig. 4.

In the visualization results, the original trace curve (blue) and the simulated trace curve (orange) are very similar in appearance and shape, with only minor differences at individual peak vertices.

In addition, the leaked information correlation analysis of the simulated traces is performed using CPA technique, and the results are shown in Fig. 5. The experimental results show that the simulated traces also have leakage information in the leakage interval, and the correlation of the leakage information is high and very close to the results of the original traces, which indicates that the simulated traces have a positive effect on keys recovery. In addition, for CW dataset, we also find that the CPA results show that the generated traces have more jittered parts than the results of the original traces, which may be the reason that the noisy data input to the generator is not completely fitted to the distribution of the real data during the training of the SCA-CGAN model, and some noisy information still exists.

In order to further evaluate whether the generated simulated traces have obvious features between each class, this paper uses the feature clustering technique to cluster

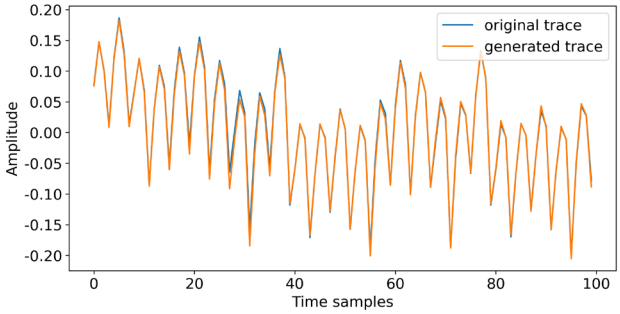


Fig. 4. The appearance of original trace and generated trace.

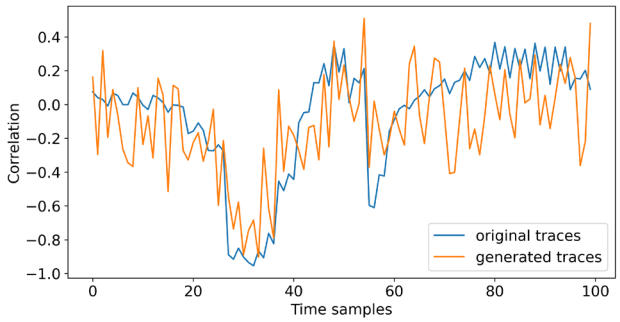


Fig. 5. The result of CPA for original traces and generated traces.

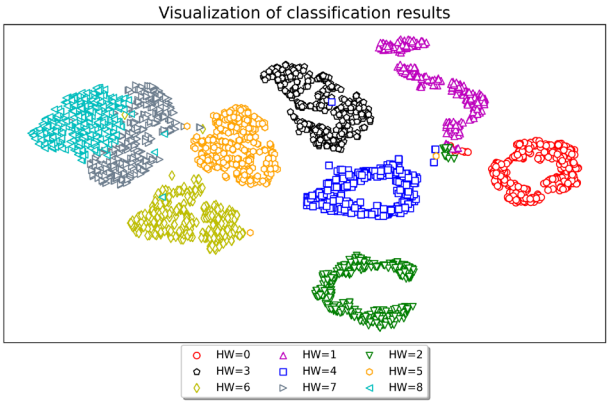


Fig. 6. The clustering result of generated traces.

the simulated traces, and the final results are shown in Fig. 6. The clustering results shows that there is no obvious crossover between each class of the generated traces, and the data points of each class are clustered relatively well, and the overall clustering effect is relatively good.

4.2.3 The Performance of Classifiers and GE

In this paper, the labels of traces selected in the profiling phase is the Hamming weight of the intermediate value of the target byte, so the number of classifications of the corresponding classifier is 9 classes in total. The specific structures and parameters of the classifier are shown in Tab. 6. The activation function of all hidden layers of the classifier is set to *ReLU*, and the activation function of the output layer is set to *Softmax*. Detailed classifier hyperparameters is shown in Tab. 7.

In the experiment, all classifiers from experiment A to D are of the same structure. The final model performance of each group of experiments is shown in Fig. 7. As can be seen from Fig. 7, the accuracy and loss value of each model gradually converge optimally in the four groups of experiments from experiment A to D, and all of them can reach

Layer(type)	Output Shape	Param #
Dense	(None, 100)	10100
Dense	(None, 200)	20200
Dropout	(None, 200)	0
Dense	(None, 400)	80400
Dropout	(None, 400)	0
Dense	(None, 400)	160400
Dropout	(None, 400)	0
Dense	(None, 400)	160400
Dropout	(None, 400)	0
Dense	(None, 200)	80200
Dropout	(None, 200)	0
Dense	(None, 9)	1809
Total params:		513,509

Tab. 6. Structure of the classifier.

Optimizer	Adam
Learning rate	0.0002
Loss function	Cross-Entropy
Mini_batch	256
Epochs	1000

Tab. 7. Hyperparameter setting of the classifier.

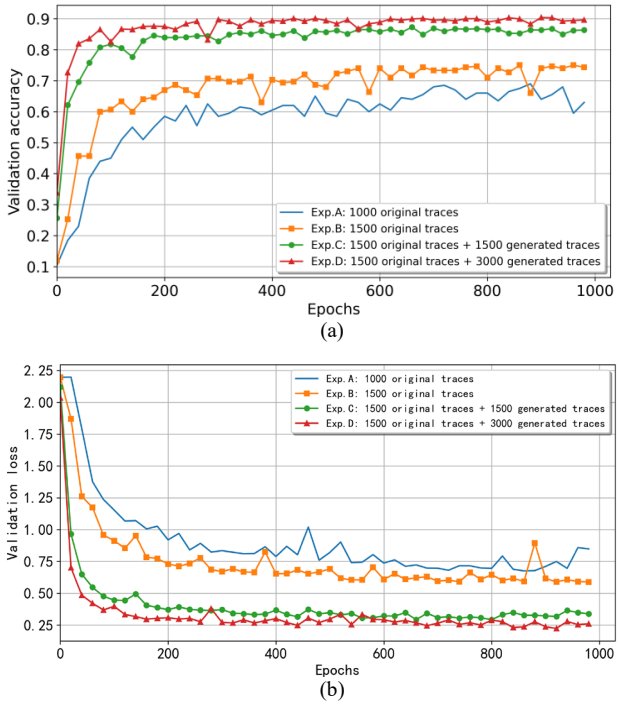


Fig. 7. (a) The validation accuracy of classifier; (b) The validation loss of classifier.

the optimal model after 1,000 iterations of training. The final results show that the model accuracy of experiment A finally reaches 70.50% and the corresponding loss value decreases to 0.6562, which indicates that the performance of this model is the worst.

However, the model accuracy of experiment D finally reaches 91.25%, which is 20.75% higher than the model accuracy of experiment A, and the corresponding loss value decreases to 0.2212, which is 0.3759 lower than the model loss value of experiment A, which indicates that the performance of this model is the best. In addition, the comparison with the experimental results of experiments C and D reveals that the results of experiments A and B are better, which indicates that the quality of the generated simulated traces is better and can enhance the original traces, which has a positive effect on the training of the classification model and can improve the accuracy of the model.

After training all the classifiers, the further operation of correct key recovery needs to be performed. In the experiment, based on the prediction results of each group of

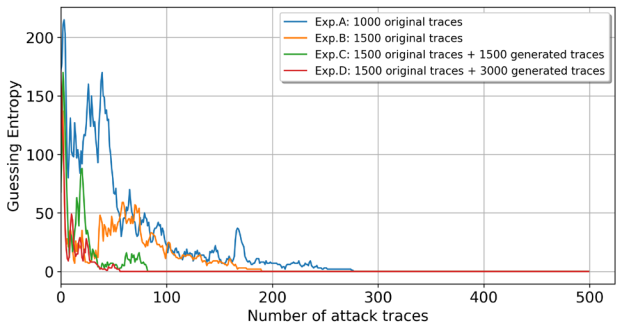


Fig. 8. The GE results of model with HW label.

experimental classifiers on the test set, the number of power traces needed to recover the correct key can be calculated by the guessing entropy algorithm, and the results of GE are visualized as shown in Fig. 8.

As can be seen from Fig. 8, when a total of 500 original power traces are used for key recovery, the GE results of all four groups of experiments can be reduced to 0, and all of them can successfully recover the correct key information. In addition, among the four groups of experiments, the number of power traces required to reduce GE to 0 is the largest in experiment D, which needs 278 traces, while it is the smallest in experiment A, which needs only 57 traces and reduces the number of power traces by about 79.5% compared with experiment D. The results of the comparative analysis prove that the generated simulated power traces have a positive effect on key recovery, which can effectively reduce the number of power traces required for correct key recovery and greatly improve the efficiency of SCA.

4.3 The Experimental Results and Analysis for ASCAD Dataset

To further investigate the performance of the SCA-CGAN model for SCA on datasets where first-order masking strategies exist, this paper performs an experimental study on each of the three jitter datasets from the ASCAD database.

4.3.1 Testing

Before the training of the SCA-CGAN model, the original energy trace data is first subjected to a preprocessing operation. All three jitter datasets in the ASCAD data contain 60,000 energy traces, each with 700 temporal sampling points. In the experiments, all the ASCAD data are firstly classified into energy traces with HW labels, and then the unbalanced traces are extracted separately and formed into a new unbalanced small sample, and then the SCA-CGAN model is trained.

After the SCA-CGAN model is trained, the quality of the generated simulated energy traces needs to be further evaluated. First, a visual comparison analysis between the original and simulated energy traces in terms of the appearance and shape of the energy traces and the correlation of the leakage information is performed, and the results are shown in Fig. 9. As can be seen from Fig. 9, the simulated energy traces generated by the SCA-CGAN model are very similar to the original traces in terms of appearance for three jittered ASCAD datasets, and there are only small deviations at some peak points. In addition, due to the presence of first-order mask in all jittered ASCAD datasets, that caused the results of all CPAs to show no significant correlation. In fact, in the actual SCA, the attacker is neither aware of the existence nor what the mask is, so the masking role of the mask is not considered in this experimental profiling phase.

Additionally, the generated simulated energy traces

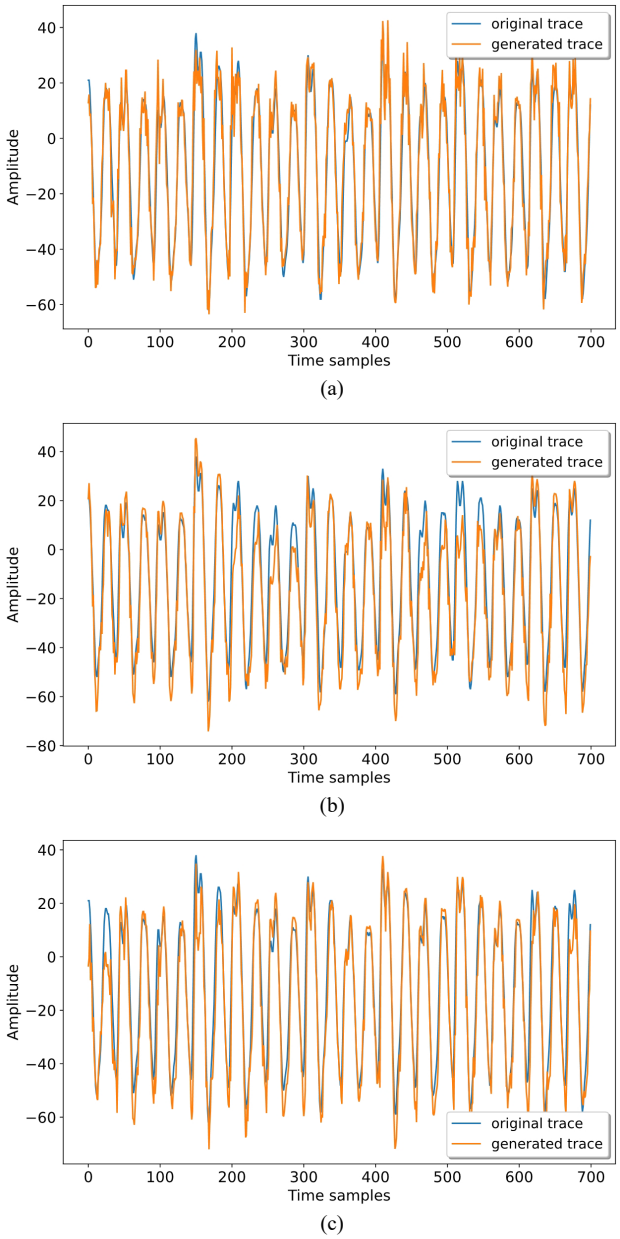


Fig. 9. Appearance of the original trace and the generated trace for: (a) ASCAD; (b) ASCAD_desync50; (c) ASCAD_desync100.

are further visualized and analyzed by feature clustering, and the experimental results are shown in Fig. 10. From the clustering effect, it can be seen that the simulated traces under all jittered situations can be successfully clustered and clustered into 9 classes at one time. In addition, there is no serious cross-mixture among all classes, which indicates that the generated simulated traces have clear class characteristics and the data quality is good.

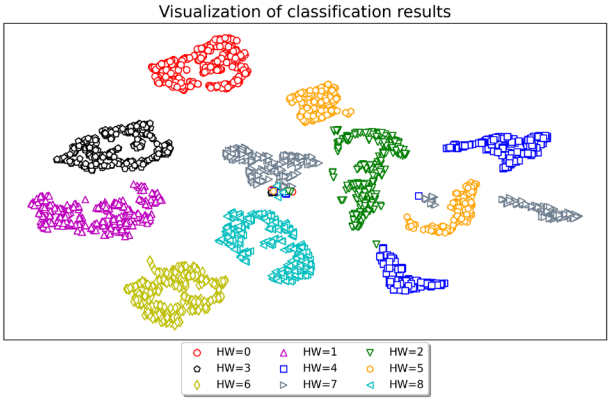
4.3.2 The Performance of GE

After evaluating the quality of all the simulated traces, we have to go further to investigate the effect of SCA on the simulated traces. In this experiment, the same experiments are performed for all jittered ASCAD datasets, and all classifiers have the same structure, which is basically

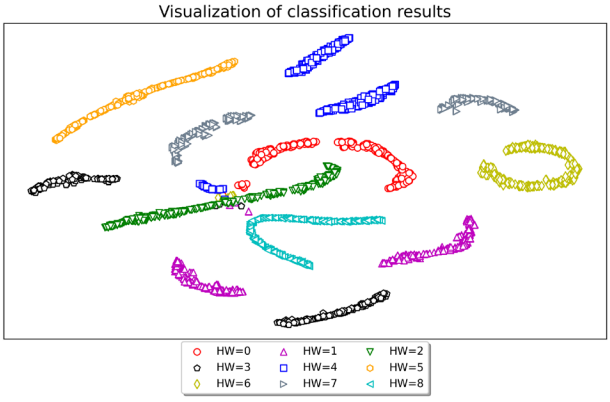
the same as the classifier structure of the CW dataset, only the number of features of the input data is different.

After training and testing all classifiers, the GE of the correct key under each jitter case is calculated, and the results are shown in Fig. 11.

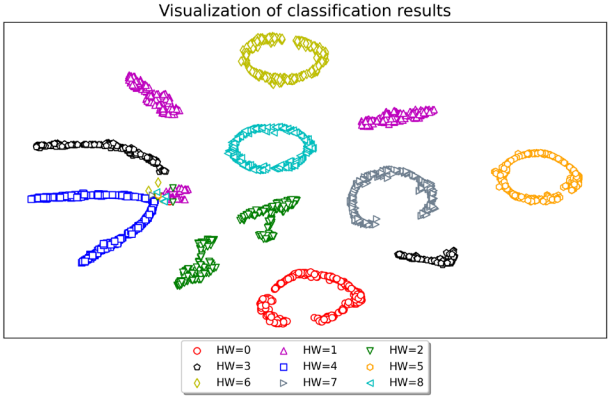
As can be seen from Fig. 11, all four groups of experiments use 1,000 original traces to calculate the GE of the correct key. Among all jittered ASCAD, only in the cases of jitter of 0 and 50, the corresponding GE can be reduced to 0, while in the case of jitter of 100, the GE cannot be reduced to 0.



(a)

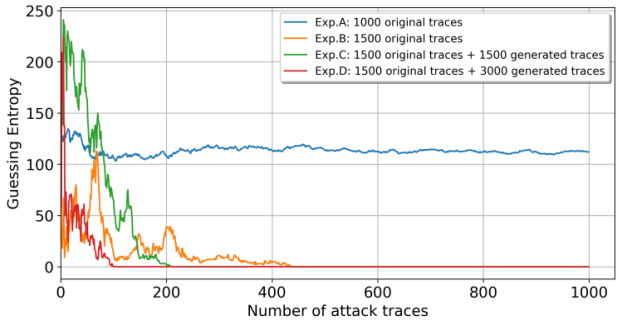


(b)

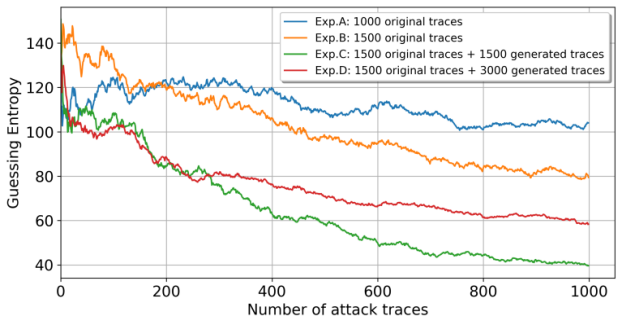


(c)

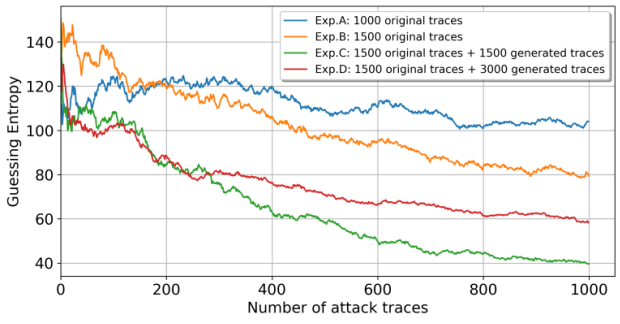
Fig. 10. The clustering result of generated traces for: (a) ASCAD; (b) ASCAD_desync50; (c) ASCAD_desync100.



(a)



(b)



(c)

Fig. 11. The results of GE for: (a) ASCAD; (b) ASCAD_desync50; (c) ASCAD_desync100.

When the jitter is 0, only the GE of experiment A cannot be reduced to 0, while the GE of the remaining three groups of experiments can be reduced to 0. In addition, experiment B requires a total of 436 original traces to reduce the GE of the correct key to 0, while experiment D requires only 101 original traces, which requires the least number of traces and is about 76.8% less than the result of experiment B.

When the jitter is 50, only experiment A still cannot reduce the GE to 0. In the remaining three groups of experiments, experiment B requires a total of 848 original traces to reduce the GE of the correct key to 0, while experiment D requires only 206 original traces, the least number of traces required, which is about 75.7% less than the result of experiment B.

When the jitter is 100, the GE of all four groups of experiments cannot be reduced to 0, and none of them can successfully recover the correct key.

In addition, the GE of experiment C and experiment D are better than those of experiment A and experiment B

in all cases of jitter, which indicates that the generated traces have a positive effect on the final key recovery and can improve the efficiency of the SCA. When the jitter is 0 and 50, all classifiers trained with the enhanced balanced dataset can successfully recover the correct key, with the results of experiment D outperforming those of experiment C, which further indicates that the enhanced dataset expanded with more generated traces can train a better classifier, which can recover the correct key faster and more efficiently.

Experimental studies on ASCAD data further show that SCA-CGAN can be successfully implemented and recover the correct key information. Thus, for side-channel energy data with a defense strategy of first-order mask protection or a tiny range of timing data jitter, it is possible to successfully implement SCA and recover the key by using SCA-CGAN.

5. Conclusion

In this paper, we address the problem of unbalanced small-sample data in DLSCA where the original dataset is small and unbalanced, by using the generator of SCA-CGAN and external constraints to generate the specified number and class of data. In this paper, experimental studies related to unproducted CW data and protected ASCAD dataset with first-order mask are performed separately, and the same four experimental scenarios are designed for each dataset for comparative analysis. For the quality of the generated energy traces, the evaluated metrics include appearance shape, leakage information correlation, feature clustering, classifier performance, and the performance of each model. The efficiency of SCA is evaluated based on the accuracy and GE of the classifier model.

Finally, the results show that the SCA using the expanded and enhanced balanced dataset is more efficient and recover the correct key faster. In addition, the data enhancement method based on SCA-CGAN is still effective for ASCAD datasets with first-order mask masking and a certain range of timing sampling jitter, and the experimental results also show that the correct key can be successfully recovered.

In our future work, we will continue to study the work on SCA-CGAN, which mainly consists of the following:

1) Studying the impact of simulated data generated by different structures of discriminators and generators on SCAs.
2) Focusing on SCAs under protection strategies such as random time delay, second-order masking, etc.
3) Finding or improving corresponding side channel defense strategies.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No. 61973109).

References

[1] NGO, K., DUBROVA, E., GUO, Q., et al. A side-channel attack on a masked IND-CCA secure saber KEM implementation. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021, no. 4, p. 676–707. DOI: 10.46586/tches.v2021.i4.676-707

[2] WANG, R., WANG, H., DUBROVA, E., et al. Advanced far field EM side-channel attack on AES. In *Proceedings of the 7th ACM on Cyber-Physical System Security Workshop*. 2021, p. 29–39. DOI: 10.1145/3457339.3457982

[3] KOLHE, G., SHEAVES, T., GUBBI, K. I., et al. LOCK&ROLL: Deep-learning power side-channel attack mitigation using emerging reconfigurable devices and logic locking. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. San Francisco (CA, USA), 2022, p. 85–90. DOI: 10.1145/3489517.3530414

[4] BRUMLEY, D., BONEH, D. Remote timing attacks are practical. *Computer Networks*, 2005, vol. 48, no. 5, p. 701–716. DOI: 10.1016/j.comnet.2005.01.010

[5] SINGH, A., KAR, M., MATHEW, S. K., et al. Improved power/EM side-channel attack resistance of 128-bit AES engines with random fast voltage dithering. *IEEE Journal of Solid-State Circuits*, 2018, vol. 54, no. 2, p. 569–583. DOI: 10.1109/JSSC.2018.2875112

[6] WANG, H., BRISFORS, M., FORSMARK, S., et al. How diversity affects deep-learning side-channel attacks. In *2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*. Helsinki (Finland), 2019, p. 1–7. DOI: 10.1109/NORCHIP.2019.8906945

[7] ARSATH, M. K. F., GANESAN, V., BODDUNA, R., et al. PARAM: A microprocessor hardened for power side-channel attack resistance. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. San Jose (CA, USA), 2020, p. 23–34. DOI: 10.1109/HOST45689.2020.9300263

[8] HU, F., WANG, H., WANG, J. Multi-leak deep-learning side-channel analysis. *IEEE Access*, 2022, vol. 10, p. 22610–22621. DOI: 10.1109/ACCESS.2022.3152831

[9] KOCHER, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *The 16th Annual International Cryptology Conference (CRYPTO 1996)*. Santa Barbara (CA, USA), p. 104–113. DOI: 10.1007/3-540-68697-5_9

[10] SHAIKH, M., ARAIN, Q. A., SADDAR, S. Paradigm shift of machine learning to deep learning in side channel attacks-A survey. In *2021 6th International Multi-Topic ICT Conference (IMTIC)*. Jamshoro & Karachi (Pakistan), 2021, p. 1–6. DOI: 10.1109/IMTIC53841.2021.9719689

[11] DANIAL, J., DAS, D., GOLDER, A., et al. EM-X-DL: Efficient cross-device deep learning side-channel attack with noisy EM signatures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2021, vol. 18, no. 1, p. 1–17. DOI: 10.1145/3465380

[12] NOBLE, W. S. What is a support vector machine? *Nature Biotechnology*, 2006, vol. 24, no. 12, p. 1565–1567. DOI: 10.1038/nbt1206-1565

[13] BIAU, G., SCORNET, E. A random forest guided tour. *Test*, 2016, vol. 25, no. 2, p. 197–227. DOI: 10.1007/s11749-016-0481-7

[14] DONG, S., WANG, P., ABBAS, K. A survey on deep learning and its applications. *Computer Science Review*, 2021, vol. 40, p. 1–22. DOI: 10.1016/j.cosrev.2021.100379

[15] SHORTEN, C., KHOSHGOFTAAR, T. M., FURHT, B. Deep learning applications for COVID-19. *Journal of Big Data*, 2021, vol. 8, no. 1, p. 1–54. DOI: 10.1186/s40537-020-00392-9

[16] KUBOTA, T., YOSHIDA, K., SHIOZAKI, M., et al. Deep learning side-channel attack against hardware implementations of AES. *Microprocessors and Microsystems*, 2021, vol. 87, p. 1–13. DOI: 10.1016/j.micpro.2020.103383

[17] TIMON, B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019, p. 107–131. DOI: 10.13154/tches.v2019.i2.107-131

[18] KRČEK, M., LI, H., PAGUADA, S., et al. Deep learning on side-channel analysis. In BATINA, L., BÄCK, T., BUHAN, I., et al. (eds.) *Security and Artificial Intelligence. Lecture Notes in Computer Science,* vol. 13049. Springer, Cham, 2022, p. 48–71. DOI: 10.1007/978-3-030-98795-4_3

[19] KORDI, F., HOSSEINTALAEE, H., JAHANIAN, A. Cost-effective and practical countermeasure against the template side channel attack. In *2020 17th International ISC Conference on Information Security and Cryptology (ISCISC)*. Tehran (Iran), 2020, p. 22–27. DOI: 10.1109/ISCISC51277.2020.9261918

[20] PICEK, S., HEUSER, A., JOVIC, A., et al. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019, vol. 2019, no. 1, p. 1–29. DOI: 10.13154/tches.v2019.i1.209-237

[21] ITO, A., SAITO, K., UENO, R., et al. Imbalanced data problems in deep learning-based side-channel attacks: analysis and solution. *IEEE Transactions on Information Forensics and Security*, 2021, vol. 16, p. 3790–3802. DOI: 10.1109/TIFS.2021.3092050

[22] BARENGHI, A., CARRERA, D., MELLA, S., et al. Profiled side channel attacks against the RSA cryptosystem using neural networks. *Journal of Information Security and Applications*, 2022, vol. 66, p. 1–14. DOI: 10.1016/j.jisa.2022.103122

[23] BÉGUINOT, J., CHENG, W., GUILLEY, S., et al. Be my guess: Guessing entropy vs. success rate for evaluating side-channel attacks of secure chips. In *25th Euromicro Conference on Digital System Design (DSD 2022)*. Gran Canaria (Spain), 2022, p. 496 to 503. DOI: 10.1109/DSD57027.2022.00072

[24] O'FLYNN, C., CHEN, Z. D. Chipwhisperer: An open-source platform for hardware embedded security research. In PROUFF, E. (ed.) *Constructive Side-Channel Analysis and Secure Design. COSADE 2014. Lecture Notes in Computer Science*, vol. 8622. Springer, Cham, 2014, p. 243–260. DOI: 10.1007/978-3-319-10175-0_17

About the Authors …

Jun-Nian WANG received the bachelor's degree from the Department of Modern Physics, Lanzhou University, in 1991, the master's degree in Radio Physics from the School of Information Science and Engineering, Lanzhou University, in 2000, and the Ph.D. degree in Control Theory and Control Engineering from the School of Information Science and Engineering, Central South University, in 2006. He has undertaken four projects of the National Natural Science Foundation of China and more than ten other provincial and ministerial level research projects. He has published more than 50 scientific papers, including more than 20 SCI/EI papers. His research interests include deep learning, intelligent information processing, and fault diagnosis. He received the Second Prize of Hunan Provincial Science and Technology Progress Award.

Wan WANG received the bachelor's degree in Optoelectronic Information Science and Engineering from the Hunan University of Science and Technology, Hunan, China, in 2018, where he is currently pursuing the master's degree in Physics. His research interests include hardware encryption, side channel analysis, and network cloud computing security.