Received：2020-10-09；Revised：2020-10-24 doi：10.3969/j.issn.0253-2778.2020.10.006

Adversarial attack based countermeasures against deep learning side-channel attacks

GU Ruizhe，WANG Ping，ZHENG Mengce，HU Honggang，YU Nenghai

CAS Key Laboratory of Electromagnetic Space Information，University of Science and Technology of China，Hefei 230027，*China*

Abstract：Numerous previous works have studied deep learning algorithms applied in the context of side-channel attacks，which demonstrated the ability to perform successful key recoveries. These studies show that modern cryptographic devices are increasingly threatened by side-channel attacks with the help of deep learning. However，the existing countermeasures are designed to resist classical side-channel attacks，and cannot protect cryptographic devices from deep learning based side-channel attacks. Thus，there arises a strong need for countermeasures against deep learning based side-channel attacks. Although deep learning has the high potential in solving complex problems，it is vulnerable to adversarial attacks in the form of subtle perturbations to inputs that lead a model to give wrong pedictions. In this paper，a kind of novel countermeasures is proposed based on adversarial attacks that is specifically designed against deep learning based side-channel attacks. We estimate several models commonly used in deep learning based side-channel attacks to evaluate the proposed countermeasures. It is shown that our approach can effectively protect cryptographic devices from deep learning based side-channel attacks in practice. In addition，our experiments show that the new countermeasures can also resist classical side-channel attacks.

Key words：side-channel attacks；countermeasures；adversarial attack；deep learning

CLC number：TP309.7 **Document code**：A

0 Introduction

Side-channel attacks (SCA) are a major threat to embedded devices[1]. They can use only a limited budget to recover the keys of cryptographic devices. The side-channel attacks exploit the side-channel information of a cryptographic computation to recover sensitive data. The side-channel information includes power consumption, electromagnetic radiations，and running-time，etc. They can recover sensitive data values in very few side-channel observations. The profiling attacks[2] are one of the most powerful side-channel attacks. In this scenario，the adversary may precisely tune all the parameters of the cryptographic device, and characterize the correlation between the physical leakage and sensitive data value. They can predict the sensitive value on a target device containing a secret they wish to retrieve by using multiple traces.

Very similar to profiling attacks，deep learning algorithms are also used in the context of side-channel attacks[3-7]. Some recent studies have demonstrated the robustness of deep learning techniques to the most common countermeasures[3,6,8]. Deep learning techniques are at least as effective as classical profiled attacks. Today，security components are embedded everywhere，so deep learning based side-channel attacks have become a major threat to many everyday life objects. Facing the application of deep learning techniques in the context of SCA，the classical security protections designed to thwart classical side-channel attacks can no longer protect modern security components. Therefore，there arises a strong need for new countermeasures that can protect cryptographic devices against deep learning attacks.

Citation：GU Ruizhe，WANG Ping，ZHENG Mengce，et al. Against deep learning side-channel attacks. J. Univ. Sci. Tech. China，2020，50(10)：1343-1358.

0.1 Related work

Singh et al[9] exploited random dynamic voltage and frequency scaling to thwart SCA. Courousse et al[10] presented a code morphing runtime framework to resist SCA. Boulet et al[11] described the protection of electronic devices against hidden-channel analysis. The protection converts the original codes to functionally equivalent codes by a modified compiler. Coron et al[12] mitigated side-channel attacks by the execution of dummy instructions. Ambrose et al[13] proposed to randomly insert a limited set of randomly selected instructions. They argued that such instructions could protect devices. As compared to our work, these previous works insert randomly selected instructions into the entire algorithm or the entire sensitive function. Our main contribution is to select the best suitable noise instructions and determine the exact insertion position.

Some recent studies have demonstrated the robustness of deep learning techniques against the most common countermeasures[3,6,8]. Therefore, there arises a strong need for new countermeasures that can protect cryptographic devices against deep learning attacks. In particular, to the best of our knowledge, the only former work that uses adversarial attacks to resist SCA is carried out by Picek et al[14]. However, different from our work, they just modified each side-channel trace into adversarial samples. The experiments in this paper show that turning each side-channel trace into an adversarial trace is not an effective countermeasure.

0.2 Our contributions

In this paper, we present a kind of countermeasures against the deep learning based side-channel attacks. The key idea of our approach is to add adversarial perturbations to the cryptographic algorithm implementation during compilation. We propose an approach to selecting adversarial perturbation instructions, and where to insert these instructions. Moreover, we also evaluate the security of our countermeasures by experiments.

In our experiments, we use two different deep learning techniques to assess the security level of our countermeasures: Multilayer Perceptron (MLP) and Convolutional Neural Network (CNN). The experimental results show that our countermeasures can reach a high level of security under deep learning based SCA. We also evaluate the performance of our approach under the classical side-channel attacks. Template attacks (TA) are exploited to attack our countermeasures, and the experiment shows that our method can thwart such attacks.

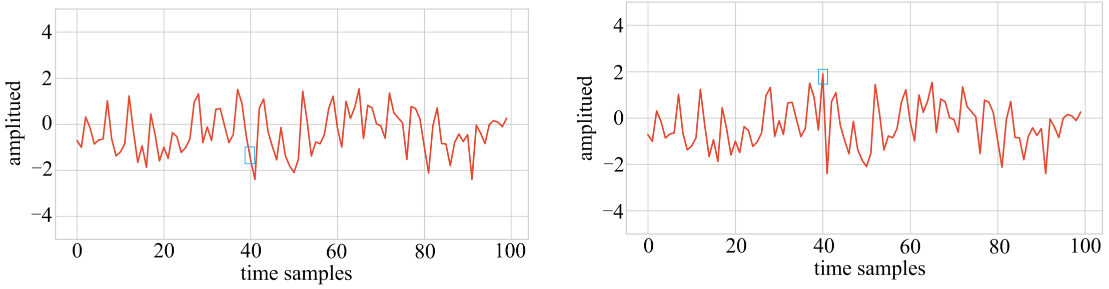
0.3 Organization

The paper is organized as follows. After describing notations and terminology in Section 1.1, Sections 1.2 and 1.3 give some background on side-channel techniques and adversarial attacks. The threat model is described in Section 2. In Section 3, we describe our countermeasures in detail. Some experiments are implemented in Section 4. Section 5 concludes this paper.

1 Preliminaries

1.1 Notations and terminology subsection-nat

In this paper, k^* denotes secret keys, $\mathfrak{K}$ denotes the set of all possible keys, and $\mathfrak{D}_{\text{profiling}}$ denotes the profiling dataset which contains the profiling traces dataset $\mathfrak{T}_{\text{profiling}}$ and the profiling labels dataset $\mathfrak{L}_{\text{profiling}}$. The profiling traces dataset contains N profiling traces, and each trace is composed of n time samples. The profiling labels



(a) Original trace

(b) Trace with one pixel perturbation added

Fig. 1 One-pixel attack on side-channel traces. The label of (a) is 0. The prediction vector of (a) is [0.8141893, 0.18581069], It is correctly classified as 0. (b) is obtained by adding one pixel perturbation in the 40 th time sample, and (b) is incorrectly classified as class 1. The modified time sample is highlighted with blue rectangle.

dataset contains the classes/labels which follows the ML classification meaning for each profiling trace. $\mathfrak{D}_{\text{attack}}$ denotes the attack dataset, and it contains $\mathfrak{T}_{\text{attack}}$ and $\mathfrak{L}_{\text{attack}}$, where $\mathfrak{T}_{\text{attack}}$ contains M attack traces. We train neural network using $\mathfrak{D}_{\text{profiling}}$ and obtain a deep learning model. Model() denotes the model we trained.

Given an input trace, Model() aims to compute an output called a prediction vector $d \in \mathbb{R}^m$, where m represents the number of possible classes/labels corresponding to the input trace. Each component in d represents the confidence of corresponding possible class/label. For example, in Fig. 1, the label of the traces is the least significant bit of the third key byte corresponding to the traces, [0.4294791, 0.57052094] indicates that the confidence of label 0 is 0.4294791, and the confidence of label 1 is 0.57052094. The trace is classified as class 1, because the confidence of class 1 is greater than the confidence of class 0.

Below we introduce some definitions, which are related to adversarial attacks against deep learning based side-channel attacks. The definitions of these terms are similar to those of adversarial attacks in computer vision area[15]. The rest of this paper follows these definitions.

- Adversarial example/trace is a trace obtained by adding noise on the cryptographic devices to obfuscate the deep learning classifier.
- Adversarial perturbation is the noise added to the cryptographic devices when generating adversarial trace.
- Black-box attacks mean that the adversary attacks a deep learning model without the structure and parameters knowledge. When the adversary may have information about the structure and parameters of the model, we call it white-box attacks.
- Targeted attacks fool the deep learning models to make it misclassify adversarial traces into specified target classes/labels. They are the opposite of non-targeted attacks. The goal of the non-targeted attack is to slightly modify the cryptographic devices in a way that the side-channel trace will be classified incorrectly by generally unknown deep learning classifier.
- Universal perturbation means that the same perturbation is added to different power traces, which can make the traces misclassified by the classifier.

1.2 Side-channel attacks

In the real world, cryptographic algorithms always rely on a physical carrier, such as a PC, smart card, or embedded processor. When a cryptographic algorithm is running on a physical carrier, execution time[16], power consumption[17], electromagnetic emissions[18], and other side-channel information of a cryptographic computation are leaked. These side-channel leakages of a cryptographic computation depend on some small part of the internally used sensitive data or sensitive operations in the cryptographic devices, and can be exploited to recover keys. A key-recovery attack based on side-channel leakage analysis is called a side-channel attack for simplicity.

1.2.1 Template attack

TA can be considered as the most successful and in-depth research method in classical SCA. In this paper, we use TA to evaluate the security level of our countermeasures. Let us consider the target device executing a cryptographic algorithm with the secret key k^*. The adversary may control a copy of the target device called profiling device and priorly use it to precisely tune all the parameters of the cryptographic computation. For each possible key k the adversary observes $N^{(k)}$ time over a time interval of n time samples' power consumption of the profiling device and we denote by trace the series of observations $T_{(i)}^{(k)} = \{T_{(i)(t)}^{(k)} \in \mathbb{R} \mid t \in [1 ; n]\}$, $i = 1, \cdots, N^{(k)}$. The most common TA model modelizes the stochastic dependency between k and trace by means of a multivariate normal conditional density:

$$P(T_{(i)}^{(k)} \mid k) = \frac{1}{\sqrt{(2\pi)^n \mid \Sigma_k \mid}} e^{-\frac{1}{2}(T_{(i)}^{(k)} - \mu_k)\Sigma_k^{-1}(T_{(i)}^{(k)} - \mu_k)^*} \tag{1}$$

where $\mu_k \in \mathbb{R}^n$ and $\Sigma_k \in \mathbb{R}^{n \times n}$ are the expected value and the covariance of the n variate traces respectively.

In the context of TA, two phases may be distinguished:

Profiling Phase For each possible key k , the adversary captures $N^{(k)}$ traces $T_{(i)}^{(k)}$ over a time interval of length n . TA estimates the expected value μ_k and the covariance Σ_k by

$$\hat{\mu}_k = \frac{1}{N^{(k)}} \sum_{i=1}^{N^{(k)}} T_{(i)}^{(k)} \tag{2}$$

$$\hat{\Sigma}_k = \frac{1}{N^{(k)}-1}\sum_{i=1}^{N^{(k)}}(T_{(i)}^{(k)}-\hat{\mu}_k)^{\mathrm{T}}(T_{(i)}^{(k)}-\hat{\mu}_k) \tag{3}$$

Attack phase The attacker captures a trace T when the target device executes a cryptographic algorithm. The adversary estimates the secret key which maximizes the likelihood

$$\hat{k} = \underset{k}{\operatorname{argmax}} P(T \mid k) \tag{4}$$

1.2.2 Deep learning based SCA

Deep learning based side-channel attacks focus mainly on two techniques: multi-layer perceptrons (MLP) and convolutional neural networks (CNN)[31]. Martinasek et al[19-21] compared MLP-based methods with other classical attack such as template attacks. Cagli et al[3] have shown that MLP-based attack is far more effective than other classical methods. Prouff et al[6] have demonstrated that CNN can obtain a great success in attacking cryptographic implementations with jitter.

Deep learning based SCA[6] is similar to TA, but uses deep learning techniques as a profiling method instead of using multivariate Gaussian profiling as in TA. To train a deep learning model, the typical leakage models used for the power consumption are the Hamming Weight (HW) model (9-class classification), and the Least Significant Bit (LSB) model (2-class classification)[22]. In this paper, we also uses these two leakage models.

MLP MLP is also called artificial neural networks. It contains at least three layers: In addition to the input and output layers. There can be multiple dense layers between them. The number of neurons in the input layer is determined by the number of time samples n in the input data. The MLP layer is fully connected (fully connected means that each neuron in the upper layer is connected to all neurons in lower layer). If the output of the lower layer is represented by a vector X, the output of the higher layer is $f(w_{ij}x+b_{ij})$, where w_{ij} and b_{ij} are the weight and bias of the jth neuron in the ith layer respectively, and $x \in X$. Generally, the function f is sigmoid or tanh. Finally, the output layer can be viewed as multi-class logistic regression, i. e. softmax regression. Thus, the output of the output layer is softmax($wx+b$). The parameters of the MLP are all the connection weight w and the bias b between the layers. The process of training these deep learning models is the process of finding the optimal parameters. How to set the optimal parameters is an optimization problem. To solve the optimization problem, the easiest method is the gradient descent method.

CNN CNN can be regarded as a variant of MLP. In addition to the input layer, dense layer and output layer, it also uses one or more convolutional layers and pooling layer. A convolution layer includes a convolution operation, followed by an activation function (such as ReLU) and a pooling layer. The pooling layer is used to reduce the dimensions. The convolutional layer performs a series of convolutional operations on its inputs (each input is convoluted with a filter).

1.3 Adversarial attacks

The adversary may design a targeted machine learning sample (adversarial example/trace) to make the machine learning model misjudge. This is called an adversarial attack. Szegedy et al[23] first discovered an interesting weakness of deep neural networks in image classification. Their study shows that, despite the high accuracy of deep learning, it is surprising that they are susceptible to adversarial attacks in the form of applying small adversarial perturbations on source images. Mohse et al[24] demonstrated the existence of universal perturbations, which can be used to fool deep learning classifiers by adding it to arbitrary images. This work inspires us to protect cryptographic devices by adding universal perturbations.

In fact, the deep learning based side-channel attacks are also classification problems. They use deep learning techniques to classify side-channel traces. The labels for the side-channel traces are key-related values. Adversarial attacks can be seen as the process of seeking a vector v such that

$$\mathrm{Model}(T+v) \neq \mathrm{Model}(T)$$

side-channel tarce $T \in \mathfrak{T}$. For each trace T, Model() outputs an estimated label Model(T). v is an adversarial perturbation. In the context of image classification, in order to make the adversarial perturbation less perceptible, v is often restricted to satisfy certain restrictions. If there is a v such that

$$\mathrm{Model}(T+v) \neq \mathrm{Model}(T) \text{ for “most” } T \in \mathfrak{T},$$

then v is universal perturbation.

These universal perturbations are not only universal across side-channel traces, but also generalize well across deep learning models[23-25]. The deep learning models find the decision boundaries of the data in the high-dimensional space. In order to make v as small as possible, the adversarial perturbations are all in the neighbourhood of decision boundaries. Even if different models are used to classify side-channel traces, as long as the models are efficient, the decision boundaries they find are similar. Such perturbations are therefore doubly universal, both with respect to the data and the models. That is, if we use one model to generate a set of universal perturbations, we can find that these perturbations are still effective for another model even thought it was trained with different hyperparameters or it is trained on a different set of traces.

One-pixel attack, a type of adversarial attack techniques, which fools the deep learning classifier by changing only one pixel in the image. In order to reduce the number of inserted noise instructions, we hope our countermeasures to modify as few pixels as possible. Thus, in this paper we use one-pixel attack to calculate universal perturbations.

Su et al[26] claim to have archieved an extreme case in adversarial attacks, and they fool the deep learning classifier by changing only one pixel in the image. One-pixel attack is also effective on the side-channel traces, and we show in Fig. 1 an adversarial sample on the side-channel traces generated by one-pixel attack. Trace (a) is the original trace captured during the cryptographic computation. The label for the original trace (a) is 0 , and the prediction vector calculated by classifier is [0. 8141893, 0. 18581069]. The deep learning classifier can correctly classify the trace (a). We use one-pixel attack to generate an adversarial trace (b) based on the trace (a). Trace (b) is obtained by changing one time sample in trace (a). The modified time sample is highlighted with blue rectangle. We use classifier to calculate the prediction vector of (b): [0. 4294791, 0. 57052094]. Trace (b) is incorrectly classified as class 1 by the deep learning classifier.

One-pixel attack generates the adversarial samples using differential evolution algorithm[27]. Differential evolution (DE) is a population based optimization algorithm for solving complex multi-modal optimization problems[27,28]. The differential evolution algorithm is composed of three phases: mutation, crossover, and selection. Mutation is a method used to generate random solutions. Crossover is used to enhance the diversity of random solutions. Selection removes solutions that fail to evolve, and leaves solutions that succeed in evolution. The differential evolution algorithm flow is as follows: first, a sufficient number of random variables are generated as the initial possible solution. Then, the mutation, crossover, and selection are performed in order. After completing a round, a certain termination condition is checked. If the termination conditions have not been met, the differential evolution algorithm returns to mutation, crossover, and selection; otherwise, the algorithm terminates, and outputs the best solution of the last round.

When we use adversarial attacks to protect cryptographic devices, the less adversarial perturbations we insert, the easier our counter-measures can be implemented. For one-pixel attacks, we only need to add noise at one time sample. Besides, one-pixel attacks requires less network information as it is a black-box attack.

2 Threat model

The adversary targets the secret key k^* of a cryptographic device. We call this cryptographic device the target device. The adversary has the same device as the target device, called the profiling device. We consider that the cryptographic device has sufficient computational resources to compile the code before each encryption (our counter-measures insert noise instructions during compilation), and the adversary cannot get control over the code compilation.

We also assume that the adversary does not use any preprocessing techniques on power traces, but we argue that the preprocessing algorithm cannot break our countermeasures. The effect of pre-processing techniques on our countermeasures is discussed further in Section 5. The adversary uses these two devices to carry out deep learning based side-channel attacks. The deep learning based SCA is divided into two phases:

Profiling Phase The attacks are performed on the 3rd key byte of the AES-128 , which is the same as previous work[4]. In this case, we can refer to this previous work to obtain an effective deep learning model. For each key candidate value k , the adversary captures $N^{(k)}$ power traces. All these traces make up profiling traces dataset $\mathfrak{T}_{\text{profiling}}$. In order to analyze the effectiveness of our countermeasures on deep learning models with different output classes, the side-channel traces are labeled using two leakage models: LSB and HW. The adversary trains neural network using $\mathfrak{L}_{\text{profiling}}$ and $\mathfrak{T}_{\text{profiling}}$, and obtain a deep learning model Model().

Attack phase The adversary captures M side-channel traces on the target device. For each $T_{(i)} \in T_{\text{attack}}$, the adversary uses the deep learning model Model() to get a prediction vector $d_{(i)} \in \mathbb{R}^{|\mathfrak{K}|}$:

$$d_{(i)} = \text{Model}(T_{(i)}) \tag{5}$$

The adversary selects the key candidate with the highest sum confidence as the secret key k , i. e. $k = \text{argmax}_{k \in \mathfrak{K}} (\prod_{i=1}^{m} d_{(i)})[k]$. If $k = k^*$, the key recovery is successful.

3 Adversarial attack based security protections

3.1 Differences

One-pixel attacks were first proposed to attack deep-learning models in the image classification area. Although both deep learning based image classification and deep learning based side-channel attacks use deep learning models as classifiers, there are still many differences between them.

Different training sets In the context of image classification, we train the deep learning model on source images. However, for the deep learning based SCA, the profiling traces are captured on the profiling device. In our threat model, the profiling device is a copy of target device and it is a cryptographic implementation with our countermeasures. Therefore, if our countermeasures modify each trace to adversarial trace by adding adversarial perturbations, then the profiling traces captured on profiling device are all adversarial traces. Due to the robustness of deep learning technique, such countermeasures that insert perturbations to turn side-channel traces into adversarial traces cannot protect cryptographic devices. The attack results shown in Fig. 2 confirm this view.

We first collect 60000 power traces of unprotected AES, and we call these 60000 traces source traces. 50000 source traces are used as the training set, and 10000 source traces are used as the test set, and CNN is used to attack these source traces. The attack result is shown as unprotected AES in Fig. 2. We use one-pixel attack to generate 60000 adversarial traces, and we call them as one-pixel attacked AES traces. We use AES traces that have been attacked by 50000 one-pixel attack as training set and another 10000 as test set, and then use CNN to attack these traces. The attack result is shown as one-pixel attacked AES in Fig. 2. For unprotected AES, approximately 100 traces are required for a full success of the key recovery. For one-pixel attacked AES, performing successful key recoveries requires approximately 20 traces. Fig. 2 shows that converting the source traces to adversarial traces cannot protect the cryptographic devices, but makes the implementation more vulnerable. The reason is that these adversarial traces have high confidence in the wrong label, which is generally above 0. 95 . These adversarial traces can deceive models trained on source traces. However, when adversary trains models on these adversarial traces, these adversarial perturbations which fool the original model will instead become features exploited by the deep learning techniques.

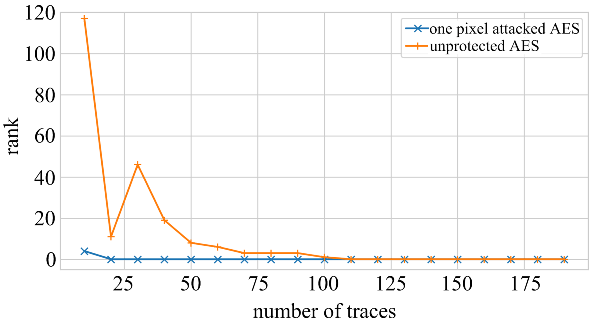


Fig. 2 The mean rank of the unprotected AES and the one-pixel attacked AES on CNN-based attack. The rank is a metric to evaluate the security level of countermeasures (described in Section 4. 2). For unprotected AES, approximately 100 traces are required for a full success of the key recovery. For one-pixel attacked AES, performing successful key recoveries requires approximately 20 traces.

Different attack dataset sizes In the image classification area, the purpose of the adversarial attack is to make a certain image misclassified by a deep learning model after adding perturbations that is not perceived by humans. We considered it a successful adversarial attack if the image was misclassified after adding perturbations. Therefore, for the image classification area, the size of the attack dataset can be regarded as 1. In SCA area, to perform successful key recoveries, the adversary captures M power traces on the target device and selects the key candidate with the highest sum confidence as the secret key, as mentioned in Section 2. The size of the attack set can be regarded as M . In order to thwart the deep learning SCA, our countermeasures need to modify all the power traces generated by cryptographic computations instead of modifying one trace.

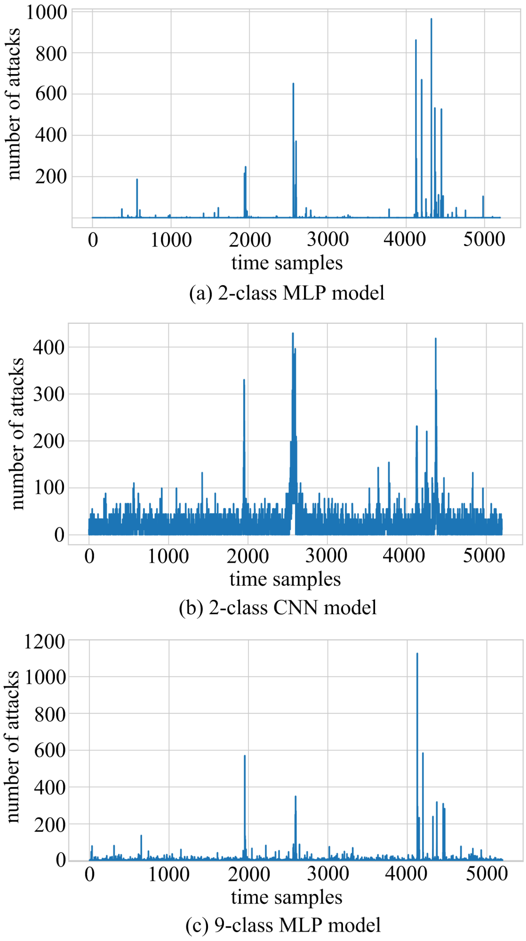
3.2 Our method

Our countermeasures insert noise instructions into the code. The power consumption of these noise instructions becomes universal perturbations. These universal perturbations make power traces misclassified by the deep learning models, and then thwart the deep learning SCA. In this process, we need to solve three problems: how to determine the position where the noise instruction is inserted, which instructions are inserted into the code as noise instructions, and how to insert the noise instruction at the selected position. We address these issues in the following subsections.

3.3 Locations of noise insertion

We want to insert noise instructions at the locations where the universal perturbations are located. Therefore, we generate universal perturbations, and observe their positions on power traces. Before calculating the universal perturbations, we need to determine what kind of universal perturbations we need to calculate. Different universal perturbations have different effects on the deep learning classifiers. Some universal perturbations make the confidence of a certain class very large, but some make it very small. These effects depend on the termination condition of adversary attacks. This subsection calculates the locations where the universal perturbations are located based on the 2-class model (the traces are labeled as LSB of sensitive value). We analyze at the end of this subsection that the positions calculated by 2-class model and 9-class model are close. We consider the formula $k = \mathrm{argmax}_{k\in\Re}(\prod_{i=1}^{m} d_{(i)})[k]$, used by the adversary to recover the secret key. The adversary selects the key candidate with the highest sum confidence as the secret key. To prevent the adversary from recovering the correct key, we can make the confidence corresponding to a certain class of all side-channel traces very large or make the confidence corresponding to each class the same. In this way, the adversary cannot recover the correct key. We test two termination conditions: the first termination condition is that the algorithm terminates when each trace is classified as 0 , and make the confidence of label 0 the as large as possible, i. e. $d_{k=0} \geqslant \tau$, where $d_{k=0}$ denotes the confidence corresponding to label 0 and τ is a constant close to 1 . Another termination condition: the algorithm terminates when the difference between the confidence of label 0 and label 1 within a small range, i. e. $|d_{k=0} - d_{k=1}| \leqslant \sigma$, where σ is a constant close to 0 .

Our experiments find that the second termination condition is computationally intensive when running differential evolution algorithms. Moreover, when the leakage model is not LSB, the labels of the traces are no longer only 0 and 1 , and the second termination condition is hard to implement. Thus, considering efficiency and versatility, our countermeasures use the first termination condition. The steps of calculating the insertion posi



(a) 2-class MLP model

(b) 2-class CNN model

(c) 9-class MLP model

Fig. 3 Distribution of adversarial perturbations. The horizontal axis represents 5200 time samples of the side-channel trace, and the vertical axis represents the number of adversarial perturbations falling on a certain time sample.

（Ⅰ）We capture 60000 labelled traces of the power consumption of unprotected AES implementation. The temporal acquisition window is set to record the first round of the AES only. Each trace is composed of 5200 time samples. We experimentally validate that the deep learning classifier trained on 50000 profiling traces can successfully recover the key in less than 1000 attack traces. Therefore, we select 50000 traces out of 60000 traces as profiling traces, and train deep learning models on these profiling traces.

（Ⅱ）Differential evolution algorithm is applied to generate adversarial perturbations based on the remaining 10000 traces. The termination condition is that the algorithm terminates when the trace is classified as 0 .

（Ⅲ）We generate 10000 adversarial perturbations based on MLP and CNN respectively. The distribution of the 10000 adversarial perturbations on 5200 time samples is shown in Fig. 3(a) and 3(b), which show that when attacking CNN and MLP, the distributions of adversarial perturbation are similar. They all have the largest distribution around the three time samples: 1900th time sample, 2560th time sample, and 4300th time sample. The perturbations near these three positions are universal perturbations.

（Ⅳ）We use these three points as the locations of the noise insertion. Noise instructions inserted near these three time samples can generate universal perturbations.

It can be observed in the power traces of AES that these three time samples are respectively included in three functions: AddRoundKey(), SubBytes() and MixColumns(). These functions may contain thousands of instructions, and we need to know where these time samples are in the cryptographic code more accurately. Only by knowing the specific location of the noise in the code can we accurately insert noise instructions into the code.

The C file is compiled to an assembly code file. We use the binary search algorithm to traverse the instructions of the assembly file, and insert the trigger_low() function after the instruction. This function is also the trigger signal used when we collect the power traces. Then we compile and run the file again, and observe whether the power trace becomes low level near the three time samples we selected earlier. If it becomes low level, then the position of trigger_low() is the position where we want to insert the noise instructions. If not, we continue to traverse the assembly file, and repeat the previous steps. In the process of determining the position of insertion, we find that even if the position the inserting the low-level signal are the same, the position of the low-level on the power trace is not the same, but their positions are very close on the power trace. The universal perturbations we generated are not all at a certain time sample but are concentrated around the three samples. Our experiments also find that if a perturbation with side-channel profile which is similar to the side-channel profile of universal perturbation is inserted near the three samples, the deep learning model can be deceived. Our purpose is only to find a fuzzy location, so that the side-channel leakage generated by the noise instructions is near to the three sampling points.

We use the 9-class MLP model to generate the adversarial perturbations, and observe their distribution on the time sample. The results are shown in Fig. 3(c). The distribution of these perturbations is similar to the distribution of perturbations generated by the 2-class model. Since universal perturbations are doubly universal, even if we use different deep learning models to generate universal perturbations, the positions of these universal perturbations are close.

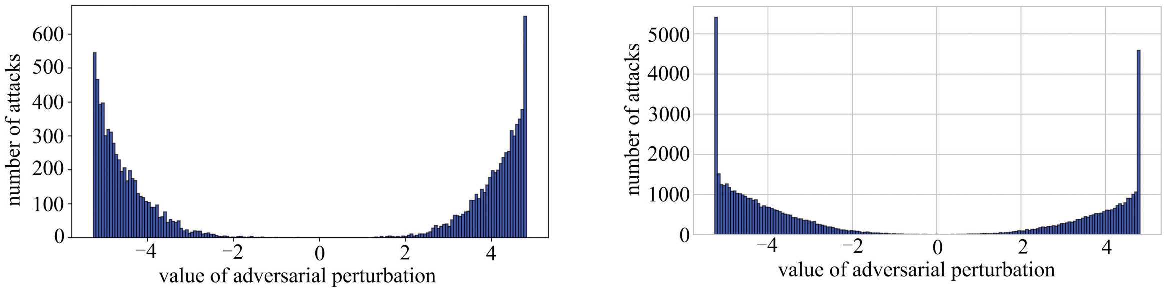
3.4 Choice of noise instructions

Previous works[10,29,30,31] insert noise instructions between each useful instruction in sensitive function Sbox. These inserted noise instructions are commonly used in cryptographic algorithms. However, the purpose of our noise insertion is different from these previous literatures. In previous works, the purpose of inserting noise is to move the point of information leakage in time and space, and to reduce side-channel leakage. Thus, in these previous literatures, the inserted noise instructions do not need to be carefully selected. The purpose of our noise instruction is to turn the captured side-channel traces into adversarial examples.

So the instructions need to meet the following requirements: ① the side-channel profile (i. e. power consumption or electromagnetic radiation) of noise instructions should be as close as possible to the profile of useful instructions, so that the adversary cannot distinguish them and filter them out from the side-channel traces[32]; ② the side-channel profile of the inserted instructions should be similar to the profile of the adversarial perturbations. The first requirement is easy to achieve, we only need to choose the instructions commonly used in cryptographic algorithms, such as addition, subtraction, exclusive or, and load. In order to meet the second requirement, the side-channel profile of adversarial perturbations should be taken into account. We analyze the distribution of adversarial perturbations over amplitude.

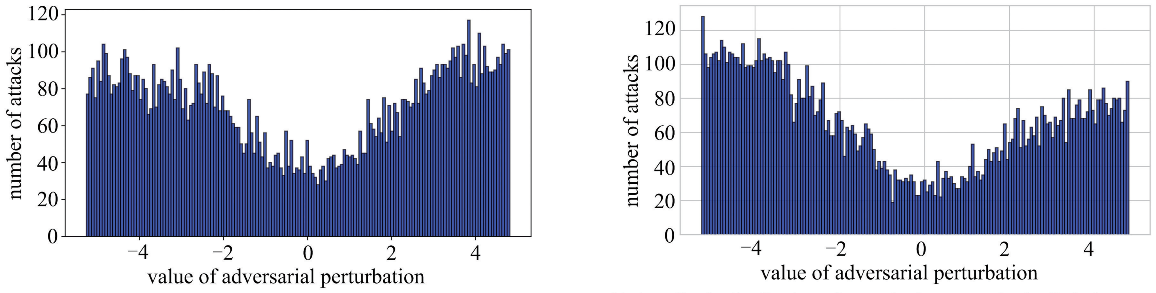
We perform one-pixel attacks on the 2-class MLP model and CNN model, and generate 10000 adversarial perturbations. Although we use the 2-class model to generate adversarial perturbations, we analyze later that the amplitude distribution of the perturbation generated by 9-class model is the same as the distribution of 2-class model. In the differential evolution algorithm, we limit the position of adversarial perturbation to the vicinity of three time samples we select in Section 3．3. We show the amplitude distribution of these adversarial perturbations near 1900th time sample and 2560th time sample in Figs. 4 and 5.

Fig. 4 shows the amplitude distribution of adversarial perturbations on MLP model. The most distributed amplitudes of adversarial perturbations are －5．2 and 4．8. Fig. 5 shows the amplitude distribution of the adversarial perturbations on the CNN model. In Fig. 5, the amplitudes of －5 and 3．8 have the most adversarial perturbations. In order to deceive both CNN and MLP, the amplitude that the noise instructions need to generate is within the interval [－5．2, －5] or [3．8, 4．8]. Such perturbations are more likely to become universal perturbation. The purpose of choosing two amplitude intervals as our criterion for selecting noise instructions is to be able to find more instructions that meet the requirements, and to make these noise instructions effective for various deep learning models.



(a) amplitude distribution of perturbations near the 1900th time sample　(b) Amplitude distribution of perturbations near the 2560th time sample

Fig. 4　Amplitude distribution of adversarial perturbations on MLP model. The values on the horizontal axis corresponds to the amplitude of power traces. The above chart shows the amplitude distribution of perturbations near the 1900th time sample, and the chart below shows the amplitude distribution of perturbations near the 2560 th time sample. We divide the interval of the amplitude [-5．2, 4．8] into 160 discrete intervals.



(a) Amplitude distribution of perturbations near the 1900 th time sample　(b) Amplitude distribution of perturbations near the 2560 th time sample

Fig. 5　Amplitude distribution of adversarial perturbations on CNN model. The values on the horizontal axis corresponds to the amplitude of power traces. The above chart shows the amplitude distribution of perturbations near the 1900th time samples, and the chart below shows the amplitude distribution of perturbations near the 2560 th time samples. We divide the interval of the amplitude [－5．2, 4．8] into 160 discrete intervals.

Tab. 1 Inserted noise instructions

mov r24, 0xff
ori r24, 0xff
ldi r24, 0xff
in r24, 0x3d

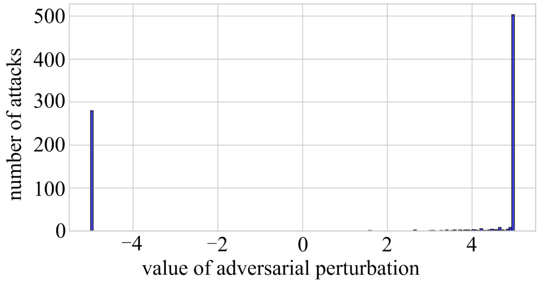


Fig. 6 Amplitude distribution of adversarial perturbation near the 1900th time sample on 2-class MLP model

In experiments, we target AES implementations running over an ARM Cortex-M3 processor. ARM Thumb1 and Thumb2 instructions are treated as a candidate noise instruction set. We capture the energy consumption traces of instructions candidate on the cryptographic device, and select the instruction that can generate a suitable perturbation size as the noise instruction.

The power consumption is not only related to instructions, but also related to operated constants. Generally speaking, 0xff causes greater power consumption. In this paper, we chose the four instructions listed in listing 1 as our noise instructions. r24 in the listing may be any free register determined by the compiler during compilation.

We use 9-class MLP to generate the adversarial perturbations, and observe their distribution. The results are shown in Fig. 6. As with the 2-class model, the perturbations generated by the 9-class model are concentrated in the largest and smallest amplitude. The noise instructions which we select using 2-class model can still generate universal perturbation in 9-class model.

3.5 Inserting noise instructions

The last step of our countermeasures is to insert the selected noise instructions into the selected positions. In this work, we insert noise instructions into the code at compile time. We start by annotating the assembly file at target positions. The annotated assembly file is recompiled before each invocation of the cryptosystem: when the compiler recognizes these annotations, it randomly picks ω noise instructions from listing 1, and inserts them to the code, where ω is an integer in $\{0, 1, 2\}$. The purpose of inserting different numbers of instructions is to increase the diversity of the code. In order to ensure that the side-channel leakage of each invocation of the cryptosystem becomes an adversarial sample, our approach requires that the cryptographic device recompiles the code at each invocation.

4 Experimental evaluation

We evaluate our countermeasures as a defense against deep learning based SCA. In order to demonstrate that our countermeasures are also effective for the classical side-channel attacks, we perform template attacks on our countermeasures. For convenience, we refer to our countermeasures as one-pixel protection AES in the following content.

4.1 Experimental setup

We use an STM32F3 board fitted with an Arm Cortex-M3 core running at 32 MHz, 16 kB of RAM, and 128 kB of flash memory. It does not provide any hardware security mechanisms against SCA. Our AES implementation is an unprotected 8-bit implementation that follows the NIST specification.

The side-channel traces are obtained with a Pico5444B PicoScope. The sampling acquisition is performed at 96 Msample/s. In this scenario, the length of one processor cycle on the side channel trace is three time samples. To ease the temporal alignment of the side-channel traces, a trigger signal is set, and held high during the execution of the first AES round. Using this setup, the security evaluation is performed under stricter conditions than it would be in practice for an adversary.

4.2 Evaluation metrics

We use two metrics to evaluate the performance of different AES implementations against attacks, which are the rank function and the accuracy.

With the same previous notations in Section 1.1, we define the score function $S_M[k]$ of the key candidates k

$$S_M[k]=\prod_{i=1}^{m} d_{(i)}[k] \tag{6}$$

According to Eq. (6) we can define the rank function:

$$\text{rank}(\text{Model}, \mathfrak{D}_{\text{profiling}}, \mathfrak{D}_{\text{attack}}, M) = |\{k \in \mathfrak{K} \,|> S_M[k] > S_M[k^*]\}| \qquad (7)$$

When the rank of k^* is 0 , we perform a successful key recovery. The larger the M required to recover the correct key, the better the implementation performs against side-channel attacks. To get a better measure of the rank, it is more suitable to estimate its mean value over several pairs of datasets.

The second metric is the accuracy which is commonly used in machine learning. We define it as：

$$\text{acc}(\text{Model}, \mathfrak{D}_{\text{profiling}}, \mathfrak{D}_{\text{attack}}) = \frac{|\{k^* \in \mathfrak{K} \mid k^* = \text{argmax}_{k \in \mathfrak{K}} d_{(i)}[k]\}|}{|\mathfrak{D}_{\text{attack}}|} \qquad (8)$$

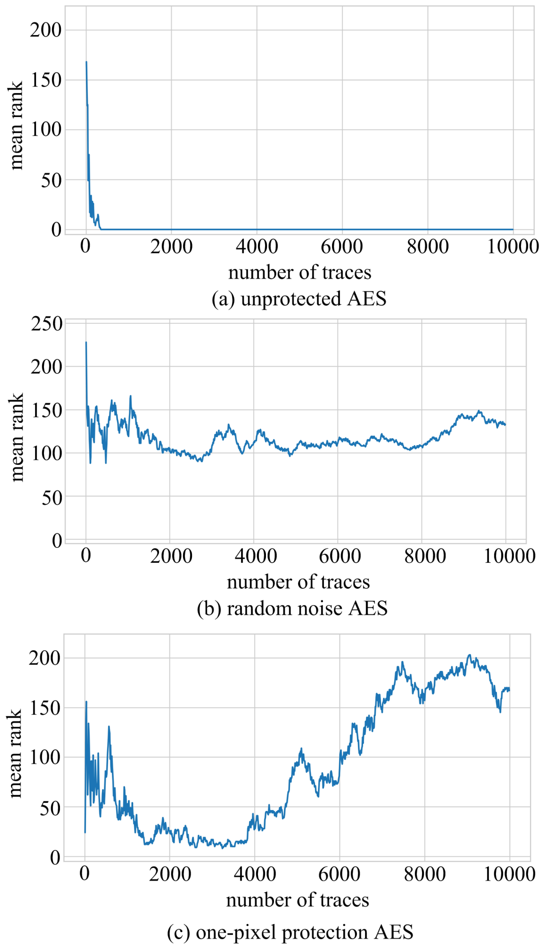
In this paper, accuracy is used to evaluate the performance of 2-class models. The numbers of elements of each class are equal. Thus, it is adequate to use the accuracy as a metric. The lower the accuracy, the better the security of the countermeasure.

4.3 Resistance to practical attacks

In this section, we use MLP and CNN to attack three different AES implementations, including unprotected AES implementation, random noise AES and one-pixel protection AES. In order to demonstrate that the effectiveness of one-pixel protection AES is due to the carefully selected insertion position and noise instructions, we implement random noise AES: randomly inserting noise instructions at three random locations in the first round of AES. The inserted noise instructions are randomly selected among instructions that are commonly used in AES programs.

We capture 60000 power traces for each AES implementation, and each trace is composed of 5200 time samples. To get a better measure of the rank function and the accuracy, we therefore need to calculate their mean value on several pairs of data sets. Among the 60000 traces, 10000 traces are randomly selected as the test set, and the remaining 50000 traces are used as the training set. Repeating this process 10 times, we get 10 different data sets.

In order to analyze the effectiveness of our method on deep learning models with different output classes, we train four deep learning models: 9-class CNN model (HW leakage model), 2-class CNN model (LSB leakage model), 9-class MLP model and 2-class MLP model. The CNN and MLP architecture used in this paper refers to Ref. [4].



(a) unprotected AES

(b) random noise AES

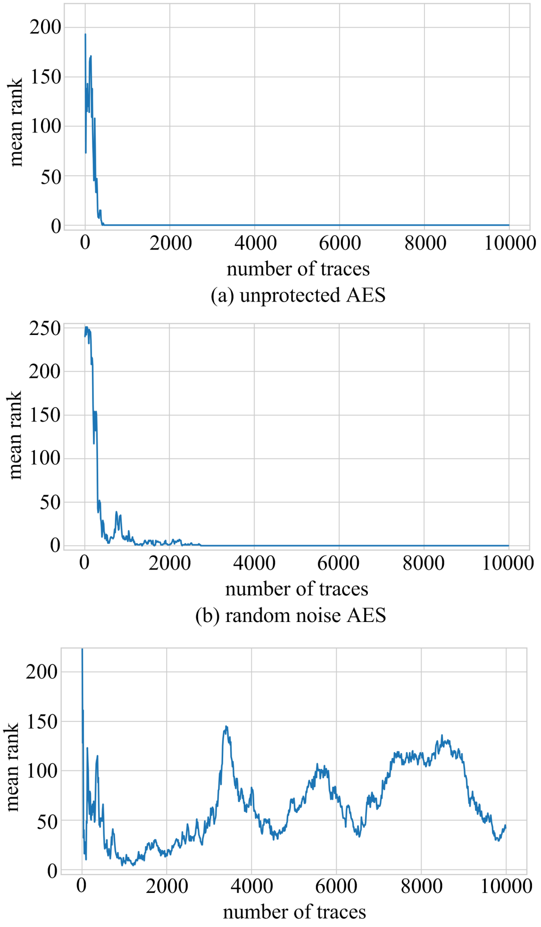
(c) one-pixel protection AES

Fig. 7 The mean rank of different AES implementations on MLP-based attacks. (a) unprotected AES, 340 traces are required for a successful key recovery. (b) Random noise AES, MLP cannot retrieve the secret key in 10000 traces. (c) One-pixel protection AES, MLP cannot retrieve the secret key in 10000 traces.

Since our targets are unmasked AES, we reduce the number of epochs. For MLP, the activation function is Relu and softmax, the optimizer is RMSprop, the learning rate is 0.00001, batch size is 256 , and the number of epochs is 100 . For CNN, we use the Softmax activation function in the classification layer combined with the Categorical/Binary Cross Entropy loss function. The learning rate is 0.0001, the optimizer is RMSprop, batch size is 256, and the number of epochs is 10.

4.3.1 2-Class model

Fig. 7 shows the results of MLP-based attacks on three AES implementations. (a), (b), and (c) respectively represent the unprotected AES implementation, the random noise AES and the one-pixel protection AES. Fig. 8 shows the results of CNN-based attacks on these three AES



(a) unprotected AES

(b) random noise AES

(c) one-pixel protection AES

Fig. 8 The mean rank of different AES implementations on CNN-based attacks. （a） unprotected AES，300 traces are required for a successful key recovery. （b） Random-noise AES，2650 traces are required for a successful key recovery. （c） One-pixel protection AES，CNN cannot retrieve the secret key in less than 10000 traces.

implementations. For unprotected AES implementation，MLP and CNN recover the secret key in 340 and 300 traces respectively. For random noise AES，MLP cannot retrieve the key in less than 10000 traces， and CNN needs about 2650 traces to recovery keys. For one-pixel protection AES，MLP and CNN cannot retrieve the secret key in less than 10000 traces.

The inserted noise instructions make the power traces of random noise AES into desynchronized traces. Therefore，MLP cannot recover the key of random noise AES. The convolution layer is the main difference between CNN and MLP，and it allows the former to have the property of shift-invariant[3]. Because of this，CNN can still recover the key of random noise AES in presence of desynchronization.

Although CNN can learn shift-invariant features，our countermeasures can still thwart CNN attacks by inserting noise. The noise instructions and insertion positions are carefully selected，and they can generate universal perturbations. Such universal perturbations are doubly universal，both with respect to the data and the network architecture. Therefore，we use a model trained on unprotected AES traces to generate a set of universal perturbations，which can still deceive other models，even when it was trained with different hyperparameters or when it was trained on a different set of traces.

We calculate correlation factor for the captured side-channel traces （see Fig. 9）. The adversary usually selects the points that leak the most information by calculating the correlation factors of the traces. In general，samples with a large correlation leak more side-channel information. Fig. 9 shows that the distribution of correlation factors is very similar to that of adversarial perturbations. Therefore，the positions where we insert the noise are the positions with larger correlation. This makes it more difficult for the adversary to recover the key.

Tab. 1 shows the mean accuracy of three AES implementations attacked by deep learning based side-channel attacks. In the experiments in this subsection，the trace is labeled with the LSB of the output of the third Sbox during the first round. So，among the side-channel traces，two classes may be distinguished：0 and 1. When the accuracy is closer to 0.5，it means that the corresponding AES achieves better resistance to side-channel attacks. The table shows that our security protection is very effective，making the accuracy very close to 0.5. In this situation，the deep learning model can hardly correctly classify the side-channel traces.

4.3.2 9-Class model

In the experiments in this subsection，the trace

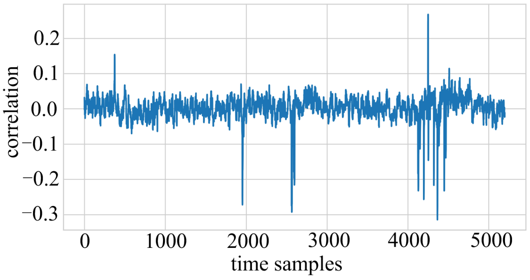
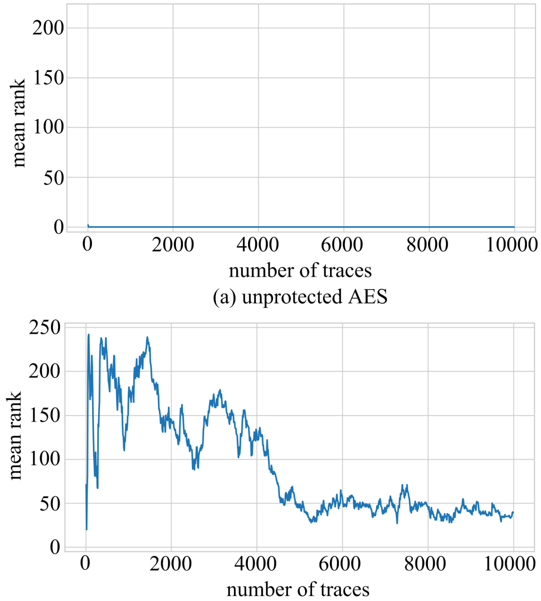


Fig. 9 Correlation of side-channel traces

Tab. 1　Mean accuracy of different AES implementations attacked by MLP and CNN

Model	AES mean accuracy		
	Unpro-tected	Rando-mnoise	One-pixel protection
CNN	0.7231	0.6296	0.5063
MLP	0.7083	0.5004	0.5023



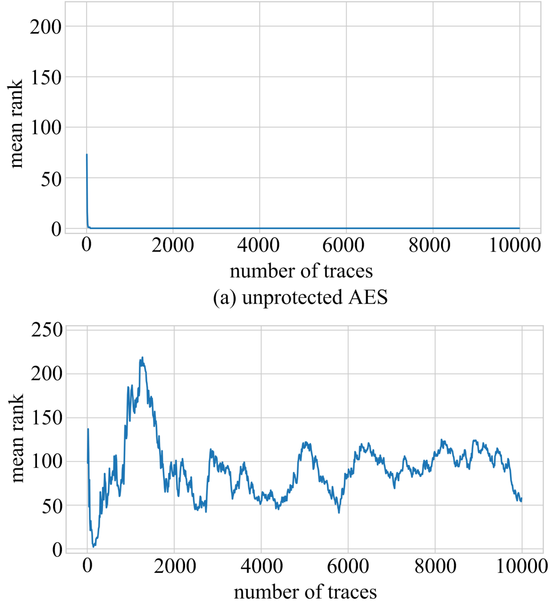
(a) unprotected AES

(b) one-pixel protection AES

Fig. 10　The mean rank of different AES implementations on 9-class MLP attacks. (a) unprotected AES, 10 traces are required for a successful key recovery. (b) One-pixel protection AES, MLP cannot retrieve the secret key in 10000 traces.

is labeled with the HW of the third output bytes of the Sbox during the first round. To compare with one-pixel protection AES, we also use 9-class model to attack unprotected AES. As in the previous sections, we have a set of 50000 power traces for the profiling phase and have a set of 10000 power traces for the attack phase.

Figs. 10 and 11 show respectively the mean rank of different AES implementations on 9-class MLP attack and CNN attack. The attack results are similar to those of the 2-class model. One-pixel protection AES can thwart 9-class CNN and MLP attacks. The reason is that the perturbations generated by inserted noise instructions are universal perturbations, and they can deceive models trained with different hyperparameters.



(a) unprotected AES

(b) one-pixel protection AES

Fig. 11　The mean rank of different AES implementations on 9-class CNN attacks. (a) unprotected AES, 50 traces are required for a successful key recovery. (b) One-pixel protection AES, CNN cannot retrieve the secret key in 10000 traces.

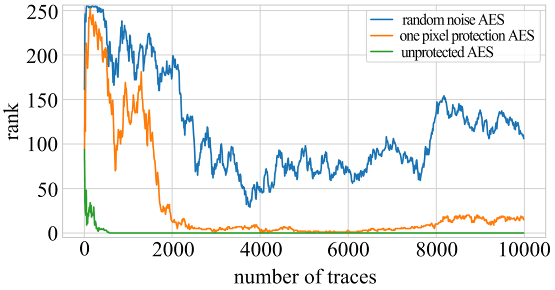


Fig. 12　Rank of the correct key attacked by a QDA (Template Attack). For unprotected AES, the key can be retrieved with 560 traces. For one-pixel protection AES and random noise AES, within 10000 traces, the key cannot be retrieved.

4.4　Resistance to classical side-channel attacks

TA are considered as the most successful method in classical SCA. We use TA to evaluate the security level of our countermeasures. We use the TA algorithm described in Section1.2.1. We perform template attacks exploiting quadratic discriminant analysis (QDA) which is a well-known generative strategy in the machine learning literature[33] to perform classification. We perform QDA on power traces composed of 5200 time samples, and do not perform any dimension reduction operation before the TA. As in previous sections, we have a set of 50000 power traces for the profiling phase and have a set of 10000 power traces for the attack phase. The attack results are illustrated in Fig. 12.

Fig. 12 illustrates that, TA only requires 560 traces to perform a successful key recovery on unprotected AES, but it cannot retrieve the secret key of one-pixel protection AES and random noise AES in less than 10000 traces. This demonstrates that, not only for deep learning based SCA, one-pixel protection AES is also effective for classical side-channel attacks. The performance of TA highly depends on some preliminary phases, such as the traces realignment or the selection of the points of interest. Our method can thwart TA for two reasons: ① we insert noise instructions, and these noise instructions are of variable length, which causes the power traces to be synchronized; ② the positions of the noise instructions we insert are also the points where the power traces have the greatest correlation, which reduce the correlation between the key k and the power traces.

4.5 Execution time overhead

Tab. 2 compares the execution time (in cycles, measured for 1000 executions of each AES implementation) of the unprotected AES and our countermeasures. The unprotected AES executes in 5482 processor cycles. The one pixel protection in 15952 to 21328 processor cycles (average 16418 cycles). Tab. 2 shows that our countermeasures lead to an increase in execution time overhead. The increased execution time overhead is mainly caused by the recompilation at each execution.

5 Conclusion

We argue that the current preprocessing algorithms cannot break our countermeasures. In order to break countermeasures of inserting noise instructions, some previous works use correlation analysis techniques (e. g. Hidden Markov Models[32]) to detect different types of patterns in leakage traces so that the adversary can distinguish noise instructions and filter them out from the side-channel traces. The reason for the effectiveness of such correlation analysis techniques is that the side-channel profile of noise instructions is different from the profile of useful instructions. Our countermeasures insert noise instructions that are often used in programs, and recompile the code at each invocation. Although this increases the execution time, it ensures that the inserted noise instructions do not have distinguishable headers and tails. Therefore, our countermeasures will not be broken by such techniques. In the context of image classification, there are some works that use dimension reduction techniques to thwart adversarial attacks (e. g. image compression[34], Principal Component Analysis[34]). Moreover, most of the existing dimension reductions are less effective. Their works demonstrate that dimension reduction techniques can reduce the interference of adversarial samples to the model, However, they can also reduces the accuracy of the model′s classification of normal examples.

Tab. 2 Execution time of the unprotected AES and one-pixel protection AES

	Unprotected			One-pixel protection		
	min.	avg.	max.	min.	avg.	max.
Execution time (cycles)	5479	5482	5486	15952	16418	21328

Existing methods for generating adversarial examples generally optimize the real examples or add perturbations to the real examples based on the gradient of the model. GAN can train a generator to generate adversarial examples without adding disturbances to specific samples. The samples generated by GAN have the advantage of being more diverse. However, GAN also has the problem that the generated samples change too much compared to the real samples. How to design constraint functions to ensure that GAN can generate adversarial examples of the target category without introducing huge perturbations is the direction of our future work.

In this paper, we present a new direction for achieving protection of cryptographic devices through one-pixel attack techniques. Based on the one-pixel attack techniques, we find the most vulnerable time samples on the side-channel observations, and find the noise instructions that may deceive the deep learning models. We implement our countermeasures and conduct experiments to evaluate the security level. Experiments show that our countermeasures can protect cryptographic devices against deep learning side-channel attacks. Our method is also effective enough against classical side-channel attacks, which makes it more competitive.

Acknowledgements

This work was supported by National Natural

Science Foundation of China（61972370，61632013），Fundamental Research Funds for the Central Universities（WK3480000007）.

References

[1] Yan M，Sprabery R，Gopireddy B，et al. Attack directories，not caches：Side channel attacks in a non-inclusive world. Symposium on Security and Privacy. San Francisco：IEEE，2019：888-904.

[2] Chari S，Rao J R，Rohatgi P. Template attacks. International Workshop on Cryptographic Hardware and Embedded Systems. Springer，2002：13-28.

[3] Cagli E，Dumas C，Prouff E. Convolutional neural networks with data augmentation against jitter-based countermeasures. International Conference on Cryptographic Hardware and Embedded Systems. Springer，2017：45-68.

[4] Prouff E，Strullu R，Benadjila R，et al. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. IACR Cryptology，ePrint，Archive，2018；53.

[5] Kim J，Picek S，Heuser A，et al. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems，2019，(3)：148-179.

[6] Maghrebi H，Portigliatti T，Prouff E. Breaking cryptographic implementations using deep learning techniques. International Conference on Security，Privacy，and Applied Cryptography Engineering. Springer，2016：3-26.

[7] Timon B. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems，2019：107-131.

[8] Masure L，Dumas C，Prouff E. A comprehensive study of deep learning for side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems，2020：348-375.

[9] Singh A，Kar M，Mathew S，et al. Exploiting on-chip power management for side-channel security. Design，Automation & Test in Europe Conference & Exhibition (DATE). IEEE，2018：401-406.

[10] Couroussé D，Barry T，Robisson B，Runtime code polymorphism as a protection against side channel attacks. IFIP International Conference on Information Security Theory and Practice. Springer，2016：136-152.

[11] Boulet F，Barthe M，Le T H. Protection of applets against hidden-channel analyses. November 21 2013，US Patent App. 13/997,136.

[12] Coron J S，Kizhvatov I. Analysis and improvement of the random delay countermeasure of CHES 2009. International Workshop on Cryptographic Hardware and Embedded Systems. Springer，2010：95-109.

[13] Ambrose J A，Ragel R G，Parameswaran S. Rijid：Random code injection to mask power analysis based side channel attacks. Proceedings of the 44th Annual Design Automation Conference. New York：IEEE，2007：489-492.

[14] Picek S，Jap D，Bhasin S. Poster：When adversary becomes the guardian-towards side-channel security with adversarial attacks. Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York：ACM，2019：2673-2675.

[15] Madry A，Makelov A，Schmidt L，et al. Towards deep learning models resistant to adversarial attacks. International Conference on Learning Representations Sydney：ACM，2018：1-10.

[16] Kocher P C. Timing attacks on implementations of diffie-hellman，rsa，dss，and other systems. Annual International Cryptology Conference. Springer，1996：104-113.

[17] Kocher P，Jaffe J，Jun B. Differential power analysis. Annual International Cryptology Conference. Springer，1999：388-397.

[18] Gandolfi K，Mourtel C，Olivier F. Electromagnetic analysis：Concrete results. International workshop on Cryptographic Hardware and Embedded Systems. Springer，2001：251-261.

[19] Martinasek Z，Dzurenda P，Malina L. Profiling power analysis attack based on mlp in dpa contest v4. 2. 39th International Conference on Telecommunications and Signal Processing. IEEE，2016：223-226.

[20] Martinasek Z，Hajny J，Malina L. Optimization of power analysis using neural network. International Conference on Smart Card Research and Advanced Applications. Springer，2013：94-107.

[21] Martinasek Z，Malina L，Trasy K. Profiling power analysis attack based on multi-layer perceptron network. Computational Problems in Science and Engineering. Springer，2015：317-339.

[22] Perin G，Ege B，van Woudenberg J. Lowering the bar：Deep learning for side channel analysis. Las Vegas：ACM，2018：163-188.

[23] Szegedy C，Zaremba W，Sutskever I，et al. Intriguing properties of neural networks，2013，arXiv preprint arXiv：1312. 6199.

[24] Moosavi-Dezfooli S M，Fawzi A，Fawzi O，et al. Universal adversarial perturbations. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii：IEEE，2017：1765-1773.

[25] Kurakin A，Goodfellow I，Bengio S. Adversarial examples in the physical world，2016，arXiv preprint，arXiv：1607. 02533.

[26] Su J，Vargas D V，Sakurai K. One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation，2019，23(5)：828-841.

[27] Das S，Suganthan P N. Differential evolution：A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation，2010，15(1)：4-31.

[28] Storn R，Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous

spaces. Journal of global optimization，1997，11(4)：341-359.

［29］Belleville N，Couroussé D，Heydemann K，et al. Automated software protection for the masses against side-channel attacks. ACM Transactions on Architecture and Code Optimization，2018，15(4)：No. 47.

［30］Amarilli A，Müller S，Naccache D，et al. Can code polymorphism limit information leakage? IFIP International Workshop on Information Security Theory and Practices. Springer，2011：1-21.

［31］Agosta G，Barenghi A，Pelosi G. Compiler-based techniques to secure cryptographic embedded software against side channel attacks. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems，2019，39(8)：1550-1554.

［32］Durvaux F，Renauld M，Standaert F X，et al. Efficient removal of random delays from embedded software implementations using hidden markov models. International Conference on Smart Card Research and Advanced Applications. Springer，2012：123-140.

［33］Fisher R A. The use of multiple measurements in taxonomic problems. Annals of Eugenics，1936，7(2)：179-188.

［34］Dziugaite G K，Ghahramani Z，Roy D M. A study of the effect of jpg compression on adversarial images，2016，arXiv preprint，arXiv:1608.00853.

［35］Hendrycks D，Gimpel K. Early methods for detecting adversarial images，2016，arXiv preprint，arXiv:1608.00530.

基于对抗攻击的侧信道防护方案

谷睿哲，汪　平，郑梦策，胡红钢，俞能海

中国科学技术大学中国科学院电磁空间信息重点实验室，安徽合肥 230027

摘要：随着深度学习技术在侧信道攻击领域的广泛应用，基于深度学习的侧信道攻击对现代密码设备的威胁越来越大. 现有的侧信道防护方案是针对经典的侧信道攻击而设计的，不能保护密码设备免受基于深度学习的侧信道攻击，因此亟需一个针对基于深度学习的侧信道攻击的防护对策. 尽管深度学习在解决复杂问题方面具有很高的潜力，但它很容易受到对输入添加轻微扰动形式的对抗攻击，从而导致模型误分类. 为此提出了一种基于对抗攻击的新颖侧信道防护对策，专门针对基于深度学习的侧信道攻击. 实验表明，该防护方案可以有效地保护密码设备免受基于深度学习的侧信道攻击和传统的侧信道攻击的威胁.

关键词：侧信道攻击；侧信道防护；对抗攻击；深度学习

GU Ruizhe：Master candidate. Research field：side-channel analysis. E-mail：zheruigu@mail.ustc.edu.cn

WANG Ping：Master candidate. Research field：Information safety. E-mail：wangpingwk@163.com

ZHENG Mengce：PhD candidate. Research field：Cryptanalysis of side channel.
E-mail：mczheng@ustc.edu.cn

HU Honggang：Corresponding author，PhD/professor. Research field：Cryptography，network security. E-mail：
hghu2005@ustc.edu.cn

YU Nenghai：PhD/professor. Research field：Video processing and multimedia communication.
E-mail：ynh@ustc.edu.cn