Hindawi
IET Information Security
Volume 2023, Article ID 6653956, 17 pages
https://doi.org/10.1049/2023/6653956

Research Article

Improving the Performance of CPA Attacks for Ciphers Using Parallel Implementation of S-Boxes

Fu Yao,[1,2] **Yongzhuang Wei**,[3] **Hua Chen**,[2] **and Enes Pasalic**[4]

[1]*University of Chinese Academy of Sciences, Beijing, China*
[2]*Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[3]*Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, China*
[4]*FAMNIT, University of Primorska, Koper, Slovenia*

Correspondence should be addressed to Yongzhuang Wei; walker_wyz@guet.edu.cn

Received 18 May 2023; Revised 29 July 2023; Accepted 8 November 2023; Published 12 December 2023

Academic Editor: Youwen Zhu

Since their introduction in early 2000, CPA (correlation power analysis), as a cryptographic tool, has been widely used in the cryptanalysis of cryptographic algorithms (being applicable to both symmetric key ciphers as well as to public key encryption schemes). An application of the classical CPA method, along with its variants, to cryptographic algorithms that use parallel implementation of its substitution boxes (S-boxes) commonly requires more power traces to extract the secret key compared to the case when serial implementation of S-boxes is employed. To reduce the amount of power traces in this scenario, we propose a modification of the standard CPA approaches and demonstrate practically that our method performs better than the existing ones in this respect. To verify the efficiency of our improved CPA method, we apply it to the public databases of DPA Contest V2. In particular, the experimental results show that only 495 power traces are required to recover the secret key of AES. We also compare the performance of our attack to the relevant methods whose parameters are available at DPA Contest V2. The results show that compared to the best nonprofiling side-channel attack (SCA) attack, our method reduces the number of power traces required to recover the secret key by 6,566. Also, our new method performs almost similarly as the best profiling SCA attack of Benoit Gerard (in terms of the required number of power traces), thus reducing the gap in the performance of profiling and nonprofiling SCA attacks.

1. Introduction

A cryptographic device running will produce some physical characteristics, such as execution time [1], power consumption [2], electromagnetic radiation [3], and so on. Essentially, the size of these physical characteristics are connected to a particular secret key used in the cryptographic device, thus in many cases enabling its efficient recovery. Therefore, an attacker can employ relevant statistical models to analyze these characteristics generated by the cryptographic devices and eventually to recover the secret key, which is in general referred to as the side-channel attack (SCA). Depending on whether there is a profiling phase in the attack, SCA methods can be classified into two classes, namely nonprofiling and profiling.

(1) *Nonprofiling SCA Methods*. During the attack, the adversary collects physical characteristics generated by the targeted cryptographic device that uses a fixed secret key k^*. The attacker then uses statistical analysis, serving as a distinguisher, to calculate the correlation between the collected physical traces for the purpose of identifying the correct secret key k^* among a (sub) set of suitable key candidates $\{k\}$. Depending on the distinguisher, the nonprofiling SCA methods include simple power analysis (SPA) [1], differential power analysis (DPA) [2], correlation power analysis (CPA) [4], and mutual information analysis (MIA) [5] among others.

(2) *Profiling SCA Methods*. In this case, the attacker has access to a pair of identical devices, which are commonly called the profiling and targeted devices. The

attack is performed in two steps, known as the profiling and attack phases. During the first phase, the attacker uses the profiling device to model physical leakage of the targeted device for all possible guessed keys k. In the second phase, the attacker collects the physical characteristics of the targeted device and retrieves the correct secret key k^* through the specific leakage model. This kind of attack includes Templates Attacks [6], Stochastic Attacks [7], and Machine Learning-based Attacks [8] among others. Due to the profiling phase, this kind of attack generally performs better than the nonprofiling SCAs.

Since their introduction in the early 2000, a lot of efforts have been put in improving the performance of SCA methods. When preprocessing of the physical traces is of concern, different approaches such as linear discriminant analysis (LDA) [9], elastic alignment [10], principal component analysis (PCA) [11], singular spectrum analysis (SSA) [12], and lowpass filtering [13] are often used to improve the signal-to-noise ratio (SNR). For nonprofiling SCA attacks, the cryptography community has studied the effect of the different distinguishers [14–16] on their success rate. In addition, an effective enumeration of the guessed keys [17–19] can be useful in improving the performance of these nonprofiling attacks and rank evaluation methods [17, 20, 21] can be used to enumerate candidate keys in decreasing order of likelihood or estimate rank of the correct key. Moreover, the application of artificial intelligence and machine learning in the nonprofiling SCA attacks seems to be under rapid development recently [22]. For profiling SCA attacks, the quality of the profiling phase is one of the key factors that affects its performance. Due to the possibility of accurately extracting important properties of the available data, the methods such as deep learning [8] (e.g., deep neural networks (DNN) [23]), convolutional neural networks (CNN), and multilayer perception networks [24] are often used in the profiling phase. We also mention the efforts of the cryptographic community toward various improvements when a recovery of the secret key of cryptographic devices with a protective scheme is considered, refer to [25].

In this article, we propose certain improvements concerning the family of nonprofiling SCA attacks, particularly CPA, applied to unprotected cryptographic implementations. A more detailed overview of these improvements is given in the next subsection.

1.1. Related Work. In CHES 2004, Brier et al. [4] proposed the use of Pearson correlation coefficients for calculating the correlation between power traces of the cryptographic algorithm and the data processed by the cryptographic device. These coefficients are then efficiently used to recover (a portion of) the secret key k^* and the attack is called CPA [4]. These original ideas have been further developed for the purpose of improving the performance of the CPA method. In CHES 2006, Le et al. [26] proposed partitioning power analysis (PPA) that employs the Hamming distance as a measure of correlation. In addition, a method of improving the performance of PPA and CPA by restricting the normalization factor was also suggested in [26]. The main benefit of this approach is a reduced number of power traces required for a successful attack (approximately by half compared to the classical CPA method). In COSADE 2010, Yongdae et al. [27] demonstrated that the distribution of power traces can be used in a more sophisticated manner, which resulted in an attack on AES (implemented in Xilinx FPGA) that uses (biased) power traces whose amount is eight times less compared to the classical CPA. In the same year, Komano et al. [28] proposed another version of CPA, the so-called built-in determined subkey CPA (BS-CPA), which uses the knowledge of previously retrieved m subkeys when recovering the $(m+1)$-th subkey and thereby increasing the SNR. For the public databases of DPA Contest V1, the number of power traces required for BS-CPA to recover the secret key is 65, whereas this number equals 280 for a classical CPA.

In CARDIS 2012, Oswald and Paar [29] presented an algorithm for efficient computation of optimal filter coefficient values, which further improves the performance of CPA when preprocessing of power traces through a filter is performed. For the public databases of DPA Contest V2 (containing the power traces for different encryption algorithms and the involved secret keys), a template attack based on this method can recover the correct guessed key of the first S-box of AES with 3,000 power traces, whereas the classical CPA requires about 8,000 power traces. In ICISC 2013, Kim and Ko [30] used the so-called PCA to further improve the performance of CPA. The effect of increased correlation resulted in an even smaller number of power traces (requiring only 1,500) for recovering a portion of the secret key that affects the first S-box of AES. In SecureComm 2015, Zhang et al. [31] proposed a novel leakage model (simple-genetic-algorithm-based CPA) based on the power consumption of multiple S-boxes to turn the key-searching problem into a correlation coefficient optimization problem. The experiment shows that this method can reduce the power traces by 52% for DES and 32% for SM4 than the classical CPA.

To increase the efficiency of CPA, in CHES 2019, Timon [22] proposed a CPA variant called CNN-DDLA, which employs the deep learning and neural networks in a nonprofiling scenario. The experimental results showed that the CNN-DDLA method has the ability of recovering the secret key using 3,000 software desynchronized traces, whereas the classical CPA would fail in achieving this. Recently, in CHES 2019, Robyns et al. [32] improved the performance of CPA by encoding the leaked information for the purpose of maximizing the correlation coefficient for the EM traces. In 2020, Kwon et al. [33] introduced a novel approach that uses deep learning techniques in the nonprofiling SCA scenario. For a software AES implementation on the chip Whisperer-Lite platform, this approach improves the SNR from 5.18 to 20.49 when deep learning techniques are employed. In FGCS 2020, Ding et al. [34] pointed out that there are bitwise linear leakages in the implemenation of the block ciphers which involve keys with XOR operation and proposed a genetic-algorithm-based approach to conduct attacks on bitwise lineak leakages.

1.2. Motivation and Contributions. When classical CPA methods are used in cryptanalysis of cryptographic devices such as single chip microcomputer, RFID tags, wireless sensors, and only dozens (possibly hundreds) of power traces are needed for recovering the secret key of the device. However, applying a classical CPA to devices implemented on FPGA and ASIC platforms requires thousands of power traces for the key recovery. The reason for this phenomenon is that different implementations of the cryptographic algorithm can make the classical leakage model of CPA, based on a single S-box, more or less accurate.

To be more specific, a serial implementation of the cryptographic algorithm's S-boxes makes its power consumption to be consistent with the operation of adding round subkeys, whereas in the case of a parallel implementation the power consumption of the device is rather consistent with the sum of the operations performed on each subkey. In the latter case, when the attacker uses a classical CPA to calculate the correlation between available power traces of a cryptographic device and the affected subkey, the power consumption of other subkeys is considered as noise which results in an increased number of power traces needed for a key recovery. It is worth noting that although there have been a plenty of works on improving the performance of nonprofiling SCA attacks, these approaches almost exclusively focus on the scenarios where S-boxes are implemented serially. Hence, the problem of reducing the noise and an optimal utilization of the leakage information (for the purpose of improving the performance of standard CPA methods) when S-box primitives are implemented in parallel appears to be a challenging task.

To address the above problem, we propose a subtle modification of the existing techniques toward their application to the case when S-box primitives are implemented in parallel. To achieve this, we give a detailed overview of the relevant methods and discuss the suggested modifications, which are summarized below.

(1) *An Overview of Three Related Works*. We review three works on improving the performance of CPA against block ciphers found in [26–28]. In order to fairly compare the efficiency of these methods to our approach, under the assumption that S-boxes are implemented in parallel, we have reimplemented these methods and applied them to the public databases of DPA Contest V2. Based on these simulation results, we deduce the following conclusions:
   (i) The methods of Le et al. [26] and Komano et al. [28] have limited ability to improve the performance of CPA in this setting.
   (ii) The method of Yongdae et al. [27] requires a selection of certain specific power traces from a larger pool of power traces, which essentially does not reduce their number.

(2) *Our Modified CPA Method.* We propose another strategy that improves the performance of CPA cryptanalysis in the above mentioned setting (having S-boxes implemented in parallel). The main refinement of our approach consists of suitable restrictions of the normalization factor and the use of previously recovered subkeys recursively when recovering the next subkey. When our method is applied to the public databases of DPA Contest V2, the simulation results indicate that it outperforms the other methods in terms of the required number of traces, see Table 3 and Figure 5.

1.3. Organization. The rest of this paper is organized as follows: Section 2 describes the common leakage model and the standard CPA method in more detail. In Section 3, we give an overview of the methods in [26–28] and describe our novel CPA approach. The performance of our improved CPA method is demonstrated experimentally through its application to the public databases of DPA Contest V2 in Section 4. Some concluding remarks are given in Section 5.

2. Preliminaries

We now provide a brief introduction to the relevant concepts such as the common leakage model and the correlation power analysis.

2.1. Common Leakage Model. A majority of power analysis attacks uses Hamming distance or Hamming weight model as the preferred leakage models. Define a data word $\beta = b_1 b_2 \ldots b_d$, where $b_i \in \{0, 1\}$. The Hamming weight of D is the number of bits whose logical value is 1, that is,

$$\mathrm{HW}(D) = \sum_{j=1}^{m} b_j. \tag{1}$$

The Hamming distance between v_0 and v_1 can be calculated as follows:

$$\mathrm{HD}(v_0, v_1) = \mathrm{HW}(v_0 \oplus v_1). \tag{2}$$

The main idea behind Hamming distance model is to calculate a total number of 0-to-1 and 1-to-0 conversions of digital circuits in a certain period of time. The simulation of power consumption in the Hamming distance model uses the following assumptions:

(1) All 0-to-1 conversions use the same amount of power as 1-to-0 conversions.
(2) All 0-to-0 conversions and 1-to-1 conversions have the same effect on power consumption.
(3) All omponents have the same effect on power consumption.
(4) The static power consumption of the components is neglected.

In this paper, the leakage model $W = aH(D \oplus R) + b$ is used to describe the relationship between the power consumption of a cryptographic device (denoted by W) and the intermediate state of cryptographic algorithm D, where D

depends both on the input data and the used key. Here, R represents a reference state (a constant machine code) and a is a constant that specifies the ratio between Hamming distance and power consumption W. Finally, b represents the remaining power consumption which is assumed to be independent of other variables.

2.2. Correlation Power Analysis. Correlation power analysis is the most widely used method when nonprofiling SCA attacks are considered. The specific steps of CPA are summarized below [4].

(1) *Select an Intermediate State.* A proper selection of the most suitable intermediate state (in either encryption or decryption direction) of a cryptographic algorithm is of great importance for the CPA. The intermediate state is usually generated by a function $f(p_{i,j}, k_j^*)$, where $p_{i,j}$ is part of the plaintext P_i and k_j^* is part of the secret key k^* corresponding to $p_{i,j}$. Those intermediate states that satisfy this condition can leak some information about the subkey k_j^*.

(2) *Collect Power Traces.* An experimental platform is built and the encryption/decryption process is performed N times, thus generating plaintext/ciphertext pairs (P_i, C_i), for $i = 1, \ldots, n$ (P_i or C_i being uniformly distributed). For each encryption or decryption, the power trace is collected with the oscilloscope and denoted as $W(P_i)$ or $W(C_i)$. Notice that the collected power traces need to be preprocessed (such as alignment operation [10], etc.) to ensure that the power consumption of different power traces at the same time refers to the same encryption operation.

(3) *Calculate Possible Intermediate States.* Enumerate all possible values $k_{j,l}$ of the subkey k_j^* and record them in the vector $\vec{k}_j = (k_{j,1}, \ldots, k_{j,N_k})$, where N_k represents the number of values that $k_{j,l}$ can take. For each possible value $k_{j,l}$ of the subkey k_j^*, an intermediate state $d = f(P_i/C_i, k_{j,l})$ is calculated using the plaintext/ciphertext P_i/C_i to form a vector $\vec{d}_{j,l} = \{d_{j,l,1}, \ldots, d_{j,l,N}\}$.

(4) *Map Intermediate States to Power Consumption.* Select an appropriate leakage model. Usually, the Hamming weight model performs better in software implementations, whereas the Hamming distance model is more suitable for hardware implementations. For each possible value $k_{j,l}$ of the subkey k_j^*, the power consumption corresponding to all intermediate states in $\vec{d}_{j,l} = \{d_{j,l,1}, \ldots, d_{j,l,N}\}$ is calculated and these values form the vector $\vec{H}_{j,l} = \{h_{j,l,1}, \ldots, h_{j,l,N}\}$.

(5) *Calculate and Compare the Correlation Coefficients.* For each possible value $k_{j,l}$ of the subkey k_j^*, the correlation coefficient $r_{j,l}$ is calculated using the power traces $W(P_i)/W(C_i)$ and the intermediate state vector $\vec{H}_{j,l} = \{h_{j,l,1}, \ldots, h_{j,l,N}\}$. The value of $k_{j,l}$ that corresponds to the maximum correlation coefficient is regarded as a correct guess for the subkey k_j^*. The correlation coefficient $r_{j,l}$ in the encryption direction is given by Equation (3).

$$r_{j,l} = \frac{N\sum_{i=1}^{N} W(P_i)h_{j,l,i} - \sum_{i=1}^{N} W(P_i)\sum_{i=1}^{N} h_{j,l,i}}{\sqrt{N\sum_{i=1}^{N} W(P_i)^2 - \left(\sum_{i=1}^{N} W(P_i)\right)^2} \cdot \sqrt{N\sum_{i=1}^{N} h_{j,l,i}^2 - \left(\sum_{i=1}^{N} h_{j,l,i}\right)^2}}. \tag{3}$$

3. Related Works and the Description of Our Attack

In this section, we first review the three relevant works on improving the performance of CPA, considered by Le et al. [26] at CHES 2006, the method of Komano et al. [28], and the Yongdae et al. [27] approach. Then, we present the detailed steps of our improved CPA method. Later, in Section 4, we will estimate the performance of these three methods when applied to certain block ciphers whose S-boxes are implemented in parallel and give a comparison to our method.

3.1. An Overview of the Relevant Works

(1) Le et al. [26] Method.

The so-called PPA was originally proposed in [26]. PPA is a general concept that includes the mono-bit DPA (differential power analysis), the multibit DPA and CPA based on the Hamming distance. We now describe the PPA attack in more detail when it is applied during the encryption process of the considered block cipher. During the attack, we first need to determine the targeted d-bit set $\beta = b_1 \ldots b_d$ and the number of guessed keys $N_k = 2^d$. Then, for each guessed subkey $k_{j,l}$ (as a d-bit portion of the secret key), one computes the Hamming distance $m = \mathrm{HD}(W(P_i), R, D, k_{j,l})$ with each trace $W(P_i)$ and subsequently divides N power traces $W(P_i)$ into $(d+1)$ classes $G_{j,l,0}, \ldots, G_{j,l,d}$, where $G_{j,l,m} = \{W(P_i), i \in [1, N] | m = \mathrm{HD}(W(P_i), R, D, k_{j,l})\}$. Finally, one can calculate the correlation coefficient $r_{j,l}$ of the guessed key $k_{j,l}$ with respect to $W(P_i)$ using the following equation:

$$r_{j,l} = \frac{\sum_{m=0}^{d} \left(\alpha_{j,l,m} \frac{\sum_{G_{j,l,m}} W(P_i)}{N_{j,l,m}} \right)}{\sigma_W \sigma_H}, \tag{4}$$

where $N_{j,l,m} = \#(G_{j,l,m})$ refers to the number of power traces in $G_{j,l,m}$, and σ_W represents the standard deviation of N power traces and can be computed as $\sigma_W = \sqrt{\frac{1}{N}\sum_{i=1}^{N} W(P_i)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N} W(P_i)\right)^2}$. On the other hand, σ_H stands for the standard deviation of the Hamming distance of a guessed key $k_{j,l}$ when using N power traces. It can be computed using $\sigma_H = \sqrt{\frac{1}{N}\sum_{i=1}^{N} h_{j,l,i}^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N} h_{j,l,i}\right)^2}$, where

$\alpha_{j,l,m}$ denotes a specific weight of each class $G_{j,l,m}$, and it can be determined using $\alpha_{j,l,m} = \frac{N_{j,l,m}}{N}(m - \sum_{n=0}^{d} \frac{N_{j,l,m}}{N} \cdot n)$. However, if the targeted d-bit set $\beta = b_1 \ldots b_d$ follows a uniform distribution, then $\alpha_{j,l,m}$ is given by:

$$\alpha_{j,l,m} = \frac{C_d^m}{2^d}\left(m - \sum_{n=0}^{d} \frac{C_d^n}{2^d} \cdot n\right), \tag{5}$$

where $C_d^m = \frac{d!}{m!(d-m)!}$ (similarly for C_d^n). The main difference between DPA and CPA is essentially the value $\sigma_W \sigma_H$ which equals to 1 for the DPA, whereas for CPA, $\sigma_W \sigma_H$ is a product of the standard deviation of N power traces σ_W and the standard deviation σ_H of the Hamming distance of a guessed key $k_{j,l}$ with N power traces.

In [26], it was shown that the normalization of $\sigma_W \sigma_H$ negatively affects the SNR, which becomes low and can render the PPA attack impossible. In order to improve the performance of PPA, the normalization effect of $\sigma_W \sigma_H$ was reduced by adding to σ_W a positive constant ε [35]. The correlation coefficient $r_{j,l}$ is then computed as follows:

$$r_{j,l} = \frac{\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}} W(P_i)}{N_{j,l,m}}\right)}{(\sigma_W + \varepsilon)\sigma_H}. \tag{6}$$

Note that there are two aspects that need to be taken into account when applying PPA-like cryptanalytic techniques. The first one is that usually the plaintext blocks corresponding to power traces are random. In a chosen plaintext scenario, there is a possibility of reducing the noise of power traces which leads to a better SNR. The second aspect is that the value of a positive constant ε should be chosen carefully in which case the SNR can be improved without modifying any parameter of significance.

(2) Komano et al. [28] Approach.

The classical CPA method recovers the targeted d-bit of the secret key of the cryptographic algorithm individually. Such an approach is appropriate when cryptographic S-boxes are implemented serially since in this case the power consumption refers to a single targeted S-box. However, when the encryption algorithms use FPGA or ASIC as its implementation platform, then S-boxes are commonly implemented in parallel for the sake of performance. For this kind of cryptographic devices, power traces collected by the attacker correspond to all S-boxes used in a single encryption round (these S-boxes process the secret key k^* as a part of its input data). In this case, the attacker recovers a d-bit subkey k_j^* of the secret key k^* by calculating the correlation between a guessed subkey $k_{j,l}$ and the power consumption which refers to all S-boxes implemented in parallel. This means that the power consumption of the targeted S-box is only a suitable portion of the measured power trace, which is not consistent with the main ideas of classical CPA.

To remedy this issue, Komano et al. [28] proposed a more general power analysis, known as built-in determined subkey CPA (BS-CPA). Its main strategy is to use previously recovered subkeys recursively when computing the correlation between a guessed key $k_{j,l}$ and measured power traces, in the process of recovering a new subkey. The main advantage of BS-CPA over classical CPA is that a larger number of successfully recovered subkeys implies a greater correlation for the remaining subkeys and in general the number of power traces required is smaller compared to the CPA. A pseudocode below, describes the main steps of BS-CPA.

In the above algorithm, n_{k^*} denotes the number of subkeys of the secret key k^*, whereas $I_0 = \{j\}$ specifies indices of nonrecovered subkeys (initialized by $I_0 = \{1, \ldots, n_{k^*}\}$). The set of ordered pairs $I_1 = \{(j, k_j^*)\}$ collects the information about recovered subkeys, where j refers to the indices of recovered subkeys and k_j^* refers to the value of j-th recovered subkey. Max_j represents the threshold value of the j-th subkey. The correlation coefficients $r'_{j,l}$ are computed using Equation (7),

$$r'_{j,l} = \frac{N\sum_{i=1}^{N} W(P_i)\left(\sum_{k=1}^{p} h_{k,i}^* + h_{j,l,i}\right) - \sum_{i=1}^{N} W(P_i)\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h_{k,i}^* + h_{j,l,i}\right)}{\sqrt{N\sum_{i=1}^{N} W(P_i)^2 - \left(\sum_{i=1}^{N} W(P_i)\right)^2} \cdot \sqrt{N\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h_{k,i}^* + h_{j,l,i}\right)^2 - \left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h_{k,i}^* + h_{j,l,i}\right)\right)^2}}, \tag{7}$$

where p represents the number of recovered subkeys, $h_{k,i}^* = \mathrm{HD}(W(P_i), R, D, k_k^*)$ is the Hamming distance of the k-th recovered subkey k_k^* from the power trace $W(P_i)$, with $k \in [1, p]$. The remaining symbols have the same meaning as in Equation (3).

It is important to notice the following when applying the BS-CPA. First, for the calculation of correlation coefficients of a guessed key $k_{1,l}$ (thus retrieving the first subkey), in Equation (7), $\sum_{k=1}^{p} h_{k,i}^* = 0$. Also, the BS-CPA method is suitable for the scenarios where the cryptographic algorithm uses a parallel implementation of S-boxes, and it performs better than the higher level of parallelization is. Finally, the correctness of already recovered subkeys greatly affects the number of power traces required for a successful recovery of the subsequent subkeys.

(3) Yongdae et al. [27] Approach.

The power traces collected by the attacker always contain a noise component, which can increase a required number of power traces. Apparently, the lower the noise level the smaller is the number of power traces. The noise in a cryptographic device can be divided into two classes, namely

electronic and switching noise. Electronic noise is the power consumption generated by electronic components of the cryptographic device such as, e.g., transistors, and this noise is affected by the measurement setup. Switching noise is the power consumption generated by frequent switches of the logic gets in the cryptographic device, and its level depends on the level of parallelization. In order to reduce the impact of the electronic noise on the side-channel attack, cryptanalysts usually employ some preprocessing techniques in order to increase the SNR such as PCA [11], SSA [12], and so on.

Yongdae et al. [27] proposed an efficient preprocessing technique of selecting (biased) power traces for the purpose of increasing the SNR and therefore achieving a better performance of CPA attacks. For convenience, this method is referred to as BIAS-CPA in this paper. The BIAS-CPA method can be efficiently applied when power traces follow the normal distribution, in which case the power traces with large variance or standard deviation are selected in an SCA attack. The detailed steps of the attack process are given as follows:

(i) N_1 power traces are used to determine which sampling instance has the strongest correlation with the selected intermediate state, and this instance is denoted by t_{ct}.
(ii) These N_1 power traces are then used to calculate the mean $\mu_{t_{ct}}$ and the variance $\sigma^2_{t_{ct}}$ corresponding to t_{ct}.
(iii) Then, the probability density value of the sampling instance t_{ct} is calculated for each power trace, using the probability density function which is determined from $\mu_{t_{ct}}$ and $\sigma^2_{t_{ct}}$.
(iv) According to the probability density value in (3) above, the N_1 power traces are grouped in descending order.
(v) Select the first N_2 power traces out of N_1, to conduct an SCA attack on the targeted cipher.

Notice that the BIAS-CPA method is similar to a chosen-plaintext side-channel attack. The difference between them is that the attacker selects specific power traces when mounting a BIAS-CPA attack, whereas in the latter case specific plaintext blocks are chosen for which power traces are measured.

3.2. Our New Improved CPA Method. Before we describe our method, we briefly discuss the main properties of the above mentioned methods and the reasons behind they do not behave optimally when parallel implementation of S-boxes is the underlying setting. In Section 4, we will provide experimental results that indicate a significant reduction in the required number of power traces for our improved strategy, compared to the three methods described in the previous section. A performance comparison given in Table 3 in Section 4, leads to the following main conclusions:

(1) For classical CPA, 12,688 power traces are sufficient to recover the secret key k^* of the public databases of the DPA Contest V2, while 500 power traces can recover the key of the single chip microcomputer [36]. That is to say, the number of power traces required by classical CPA for recovering the secret key of the cryptographic devices is substantially larger when S-boxes are implemented in parallel.
(2) PPA can significantly reduce the number of power traces needed for each subkey recovery but its number, being typically around 3,650 (as shown in Table 3), is still quite large.
(3) For many subkeys (but not all), BS-CPA can reduce the number of power traces required for their recovery by more than 2,000, see also Table 3. However, BS-CPA cannot reduce the number of power traces required for a recovery of the first subkey.
(4) BIAS-CPA is a quite powerful approach, but it requires a large pool of power traces and the selection of those that possess certain special properties, which cannot always be efficiently applied (the required amount of power traces being not available). Therefore, a more effective attack method is needed for handling the case of parallel implementation of S-boxes in cryptographic algorithms.

```
1: Initialization: I_0 = {1, ..., n_{k*}}, I_1 = ∅;
2:     while (I_0 ≠ ∅) do
3:       for j ∈ I_0 do
4:         Set flag = 0;
5:         for each guessed key k_{j,l} do
6:           Compute the correlation coefficient r_{j,l};
7:           if r_{j,l} = = Max_j then
8:             Delete j from I_0;
9:             Add (j, k_{j,l}) to I_1;
10:              Set flag = 1;
11:            end if
12:            if flag = = 1 then
13:              break;
14:            end if
15:          end for
16:       end for
17:    end while
Output: I_1 = {(j, k_j^*)}
```

Algorithm 1: BS-CPA.

It is worth noting that both PPA and BS-CPA aim at improving the SNR (compared to the classical methods), but the nature of these improvements differs quite substantially. Hence, there is a possibility of combining these two methods for the purpose of maximizing the SNR, especially when parallel implementation of S-boxes is considered. This leads to a significant reduction of the number of power traces needed for a secret key recovery. We notice that both PPA

and BIAS-CPA improve the performance of CPA by increasing the variance of power traces. However, this is done in a different way and whereas PPA adds a suitable constant ε to the variance of the power traces, BIAS-CPA simply selects the power traces with larger variance. The simulation results in Section 4.2, performed assuming that power traces follow a normal distribution, justify the addition of an appropriate constant ε that essentially also reduces the number of power traces used by PPA.

3.2.1. Description of CT-CPA. Our new improved CPA method, named CT-CPA in this paper (where CT stands for combined techniques), assumes that the power traces of the cryptographic device follow a normal distribution. It combines the ideas of BS-CPA and PPA with the appropriate constant ε which is selected in accordance to the standard deviation of the power traces of the cryptographic device.

When mounting CT-CPA attack on a block cipher, we consider d-bit set $\beta = b_1, \ldots, b_d$ as the attack target (any subkey being of length d bits), and divide the secret key k^* into n_{k^*} subkeys. Then, we use the BS-CPA method to recover each subkey one by one. When attacking the $(p+1)$-th subkey, the calculated hypothetical intermeidate value is the sum of the hypothetical intermediate values $(\sum_{k=1}^{p} h^*_{k,i} + h_{j,l,i})$ corresponding to p recovered correct subkeys and the guessed subkey $k_{j,l}$ (corresponding to the $(p+1)$-th subkey), rather than the one $h_{j,l,i}$ corresponding to the guessed key $k_{j,l}$. Thus, the power consumption generated by the p recovered correct subkeys cannot be treated as noise, improving the SNR during the attack process.

To compute the correlation between each possible guessed subkey $k_{j,l}$ with the p recovered correct subkey k^*_k $(k \in [1,p])$ and N power traces, we adopt the PPA method based on the Hamming distance. Specifically, we divide N power traces $W(P_i)$ into $(d+1)$ partitions classes $G_{j,l,0}, \ldots, G_{j,l,d}$ (where $G_{j,l,m} = \{W(P_i), i \in [1,N] | m = HD(W(P_i), R, D, k_{j,l})\}$) with the guessed subkey $k_{j,l}$ (corresponding to the $(p+1)$-th subkey), and $G^*_{k,0}, \ldots, G^*_{k,d}$ (there are a total of p similar power trace sets) (where $G^*_{k,m} = \{W(P_i), i \in [1, N] | m = HD(W(P_i), R, D, k^*_k)\}$ and $k \in [1,p]$) with each correct subkey k^*_k. Then, we use the decision signal Equation (8) to calculate the correlation coefficient $\vec{r}_{j,\text{new}}$. The detailed derivation of Equation (8) is shown in Appendix. Note that Equation (8) is a combiantion of the Equations (6) and (7). Here, we combine PPA and BS-CPA. On the one hand, we use BS-CPA to improve the SNR of the correlation coefficient $\vec{r}_{j,\text{new}}$. On the other hand, we use PPA to enhance each subbyte's attack capability, which can compensate for the shortcomings of BS-CPA.

During calculating correlation $\vec{r}_{j,\text{new}}$ using Equation (8), we need to determine the specific value of the positive constant ε. The specific value of the positive constant ε would affect the improved attack ability. According to the reimplementation of BIAS-CPA, we found that when the power traces belongs to the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$, the improved attack ability is better. Thus, we set the positive constant ε to an appropriate constant, so that $\sigma_W + \varepsilon$ in the correlation coefficient $\vec{r}_{j,new}$ (i.e., Equation (8)) immediacy approaches the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$. Specially, we use SPA to determine the sampling point t_{ct}, which has the strongest correlation with the selected intermediate state, with the available power traces. Then, we calculate the mean value $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ of the sampling point t_{ct} with these available power traces. Finally, we set the positive constant ε to ensure $\sigma_W + \varepsilon$ is in the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$. Note that the method of determining the specific value of the positive constant ε in this article is beneficial for improving the attack capability of the improved CPA, but it may not necessarily be the most effective. This forms a CPA improvement method suitable for S-boxes parallel computing scenarios.

In short, we use BS-CPA to improve SNR of the attack process, and use PPA to calculate the correlation more accurately, especially the correlation for the first subkey. In addition, we set the specific value of the positive constant ε based on the core idea of BIAS-CPA, in order to further enhance the attack capability of the improved CPA. The detailed steps of CT-CPA are given as follows:

(1) Select the sampling point which gives the largest correlation for the available power traces and denote it as t_{ct}.
(2) Calculate the mean value $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ at the sampling point t_{ct} of the power traces using a relatively small number of traces (according to simulations in Figure 3 it is sufficient to use 150 power traces to estimate the mean value and standard deviation).
(3) Initialize $I_0 = \{1, \ldots, n_{k^*}\}$, $I_1 = \{(0,0,0)\}$ and set ε to an appropriate constant value according to the mean $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ for the given power traces.
(4) Set $N = 0$. For $j \in I_0$, the following attack is performed as the number of power traces N increases:
   (i) For each guessed subkey $k_{j,l}$, the Hamming distance $m = \text{HD}(W(P_i), R, D, k_{j,l})$ is calculated using N power traces.
   (ii) Divide N power traces $W(P_i)$ into $(d+1)$ classes $G_{j,l,0}, \ldots, G_{j,l,d}$, where $G_{j,l,m} = \{W(P_i), i \in \{1, \ldots, N\} | m = \text{HD}(W(P_i), R, D, k_{j,l})\}$.
   (iii) Calculate the correlation coefficient $r_{j,l,\text{new}}$ for the guessed key $k_{j,l}$ according to the Equation (8), for $j = 1, \ldots, 2^d$, thus forming a correlation vector $\vec{r}_{j,\text{new}} = \{r_{j,1,\text{new}}, \ldots, r_{j,2^d,\text{new}}\}$.
   (iv) Sort the correlation vector $\vec{r}_{j,\text{new}}$ in descending order and record the guessed key $k_{j,l}$ corresponding to the first element in $\vec{r}_{j,\text{new}}$.
   (v) Delete j from the set I_0 and add $(j, k_{j,l}, N_j)$ to the set I_1 whenever the correlation coefficient of the guessed key $k_{j,l}$ appears at the first position in the correlation vector $\vec{r}_{j,\text{new}}$ at least T_j times, where T_j is a threshold value indicating that the guessed subkey value always appears at the first position in the correlation vector with further increase of power traces N.

(5) Check whether I_0 is empty. If I_0 is empty, terminate the attack. Otherwise, return to Step 4.

$$r_{j,l,\text{new}} = \frac{\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}} W(P_i)}{N^*_{k,m}}\right)\right] + \left[\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}} W(P_i)}{N_{j,l,m}}\right)\right]}{(\sigma_W + \varepsilon)\cdot\sigma'_H}. \tag{8}$$

Remark 1. The purpose of Steps 1, 2, and 3 is to determine an appropriate value of the positive constant ε, which is derived from BIAS-CPA. The Step 4 is a combination of PPA and BS-CPA. In Equation (8), used to compute the correlation coefficients $r_{j,l,\text{new}}$, the set $I_1 = \{(j, k^*_j, N_j)\}$ specifies the information about recovered subkeys. More precisely, j is the index of the recovered subkey k^*_j and N_j refers to the number of power traces required to recover the j-th subkey. Moreover, $m = \text{HD}(W(P_i), R, D, k_{j,l})$ represents the Hamming distance between the guessed key $k_{j,l}$ and the power trace $W(P_i)$, where $j \in [1, n_{k^*}]$ and $l \in [1, 2^d]$. Also, $G^*_{k,m} = \{W(P_i), i \in \{1, \ldots, N\} | m = \text{HD}(W(P_i), R, D, k^*_k)\}$ represents the class of power traces $W(P_i)$ related to the k-th recovered subkey k^*_k, whose is $N^*_{k,m} = \#(G^*_{k,m})$. Furthermore, $\alpha^*_{k,m}$ represents the specific weight of the class $G^*_{k,m}$. The specific formula for its calculation is: $\alpha^*_{k,m} = \frac{N^*_{k,m}}{N}(m - \sum_{n=0}^{d}\frac{N^*_{k,m}}{N}\cdot n)$. Finally, σ'_H represents the standard deviation of the Hamming distance between the guessed key $k_{j,l}$ and N power traces taking into account p recovered subkeys. The parameter σ'_H can be calculated as follows:

$$\sigma'_H = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h^*_{k,i} + h_{j,l,i}\right)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h^*_{k,i} + h_{j,l,i}\right)\right)^2}, \tag{9}$$

where $h^*_{k,i} = \text{HD}(W(P_i), R, D, k^*_k)$ represents the Hamming distance between the k-th recovered subkey k^*_k and the power trace $W(P_i)$. Here, $h_{j,l,i} = \text{HD}(W(P_i), R, D, k_{j,l})$ denotes the Hamming distance of the guessed subkey $k_{j,l}$ and the power trace $W(P_i)$. The remaining symbols have the same meaning as in the Equations (3) and (4). Note that we have determined the constant ε using the mean and standard deviation of those power traces used to recover the secret key in BIAS-CPA. However, we believe that there are other methods for specifying the constant ε, which deserve further analysis. In addition, if the available power traces are insufficient to ensure the sampling point t_{ct} which has the strongest correlation, we can perform the CT-CPA on each samping point to recover the key of the cryptographic algorithm. This would result in a longer time required for CT-CPA.

4. Experiments and Performance Evaluation

In this section, we first discuss DPA Contest V2. Then, we give the detailed parameters of the reimplementation of PPA, BS-CPA, BIAS-CPA, and CT-CPA. Finally, we give a performance comparison that clearly illustrates the benefits of using CT-CPA.

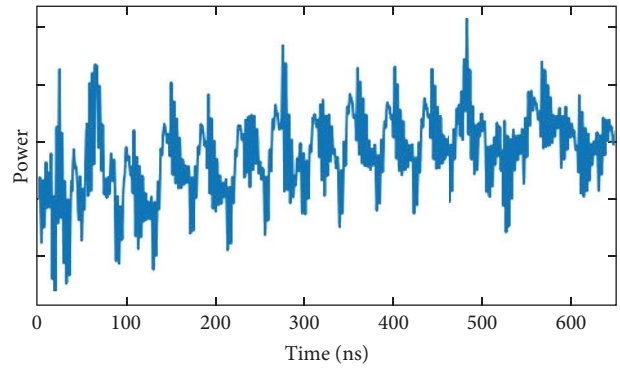

FIGURE 1: Power traces of AES on the SASEBO-GII board for DPA Contests V2. The horizontal axis represents the time samples proportional to clock cycles. The vertical axis represents the relative voltage values.

4.1. DPA Contest V2. The DPA contests specifies a series of challenges aiming at comparing the different attack methods in an objective manner [37]. Among them, the target of DPA Contest V2 is the unprotected parallel implementation of AES on FPGA. The power traces of DPA Contest V2 are collected on a SASEBO-GII board, which is a standard evaluation board for side-channel attacks. The SASEBO-GII board runs at 24 Mhz, whereas for the measurement of power traces the oscilloscope's sampling rate is 5 Gsample/s.

The DPA Contest V2 contains two public databases. The first database has 1,000,000 power traces, where each power trace is the power consumption of AES with a random plaintext and a random secret key. Hence, one can obtain the plaintext, secret key and ciphertext for each power trace with the name of the power trace file. This database is used for profiling SCA attacks. The second database has 640,000 power traces and uses only 32 random secret keys to generate 20,000 power traces for each key. Similarly, one can also obtain the plaintext, secret key and ciphertext for each power trace in this database.

The target of our attack is the second database. Figure 1 shows the power traces of of DPA Contest V2 for AES on the SASEBO-GII board. One round of AES encryption is completed within one clock cycle on the SASEBO-GII board. From Figure 1, we can see that the number of sample points for a single power trace is 3,253. Therefore, there are approximately 208 samples per clock and each power trace has a duration of 15.6 clocks, which is more than 10 rounds of AES.

4.2. Simulation Results. During the implementation of PPA, BS-CPA, BIAS-CPA, and CT-CPA, we have set sixteen 8-bit registers corresponding to the 10th encryption round of AES as the attack target. We choose the Hamming distance model as the leakage model, which is the power consumption that reflects transitions of the registers from 9th round to 10th round. In [27], three methods for determining the sample point with the greatest correlation were proposed. In this article, we use SPA to determine the 2,394-th sample point as a

Table 1: The specific value of the coefficients $\alpha_{j,l,m}$.

$\alpha_{j,l,0}$	$\alpha_{j,l,1}$	$\alpha_{j,l,2}$	$\alpha_{j,l,3}$	$\alpha_{j,l,4}$	$\alpha_{j,l,5}$	$\alpha_{j,l,6}$	$\alpha_{j,l,7}$	$\alpha_{j,l,8}$
$-1/64$	$-3/32$	$-7/32$	$-7/32$	0	$7/32$	$7/32$	$3/32$	$1/64$

sample point with the greatest correlation, hence $t_{ct} = 2,394$. The attack range is between 2,350 and 2,450. The parameters for each improved CPA method are given below.

(1) The PPA Method of Le et al. [26].

In this case, one needs to determine the specific weights $\alpha_{j,l,m}$ and the constant ε. As the number of power traces is large and the bits of β are uniformly distributed, the coefficients $\alpha_{j,l,m}$ are calculated using Equation (5). The specific values of the coefficients $\alpha_{j,l,m}$ are shown in Table 1. The value of positive constant ε is chosen to be 33, which is 10% of the standard deviation of the power traces σ_W (following the same reasoning as in [26]).

(2) The BS-CPA Method of Komano et al. [28].

Komano et al. [28] introduced the parameter Max_i to denote the maximum correlation coefficient of all guessed subkeys $k_{j,l}$, when a recovery of the j-th subkey is considered. Instead of specifying Max_i, in order to estimate the performance of BS-CPA, we have reimplemented this method assuming the knowledge of a certain portion of the subkeys. To this end, suppose we have correctly recovered p subkeys. When recovering the $(p+1)$ subkey, we use Equation (7) to calculate the correlation $r'_{j,l}$ for each guessed subkey $k_{j,l}$ using the values of p correct subkeys. The decision strategy for determining the number of power traces required to recover each subkey is that the correctly guessed subkey preserves the maximum correlation value as the number of power traces increases.

(3) The BIAS-CPA Method of Yongdae et al. [27].

The main assumption of BIAS-CPA is that the power traces follow a normal distribution. Therefore, we first verify whether the power traces of DPA Contest V2 satisfy this assumption. With $N = 20,000$ power traces, generated using the same random secret key, the mean and standard deviation of the 2,394-th sample point of the power traces are 3314.73 and 329.36, respectively. The black curve in Figure 2 represents the probability density value of the normal distribution $N(3314.73, 329.36^2)$, whereas the blue points in Figure 2 approximate the frequency of the 2,394-th sample point for $N = 20,000$ power traces. Notice that the frequency of each value at the 2,394-th sample point is obtained by dividing the number of its occurrence by the number of total power traces N. From Figure 2, we draw the conclusion that the 2,394-th sample point of $N = 20,000$ power traces of DPA Contest V2 indeed follows the normal distribution $N(3314.73, 329.36^2)$, thus applying the BIAS-CPA method to the public databases of DPA Contest V2 is therefore justified.

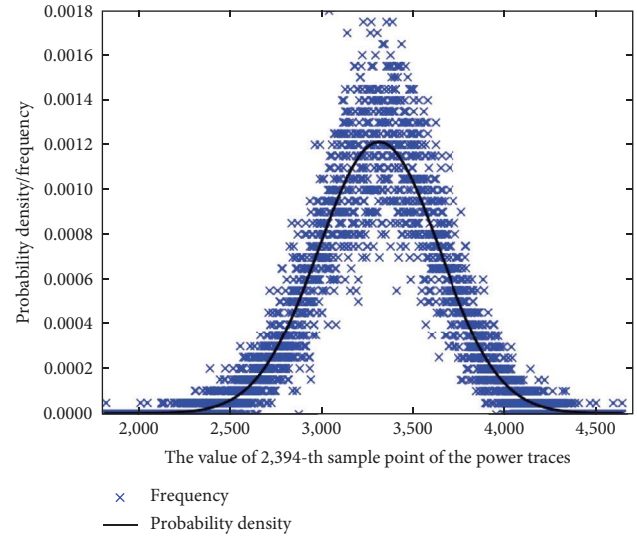

Figure 2: Probability density value/frequency of the 2,394-th sample point of the power traces. The horizontal axis represents the value of the 2,394-th sample point for each power trace. The vertical axis represents the probability density value/frequency of each value of the 2,394-th sample point of the power traces.

(4) Our New Improved CPA Method (CT-CPA).

First, CT-CPA also requires that the power traces follow a normal distribution, which has already been verified. Second, we need to calculate the mean $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ of the 2,394-th sampling point of DPA Contest V2 when using only a relatively small number of power traces. The variation of the mean $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ of the 2,394-th sampling point, as the number of power traces increases, is shown in Figure 3(a). Figure 3(b) is a close-up look at Figure 3(a), focusing on the first 250 power traces. According to the diagrams in Figure 3, after the number of power traces has increased to 150 the mean $\mu_{t_{ct}}$ and standard deviation $\sigma_{t_{ct}}$ will remain constant. This means that we can assume that the power traces follow a normal distribution even though their number is relatively small.

The next step is to set ε to an appropriate constant value. In accordance to the reimplementation results of BIAS-CPA, the power traces used to recover the secret key of the public databases of DPA Contest V2 belong to the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$. With respect to the fact that the standard deviation $\mu_{t_{ct}}$ of the 2,394-th sample point is 329.36 (for the power traces of DPA Contest V2), we have set the constant ε to be 500. In this way, when the number of power traces used to attack DPA Contest V2 exceeds 150 then $(\sigma_W + \varepsilon)$ approximately approaches the standard deviation 818.05 of the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$. The number of power traces for PPA with the

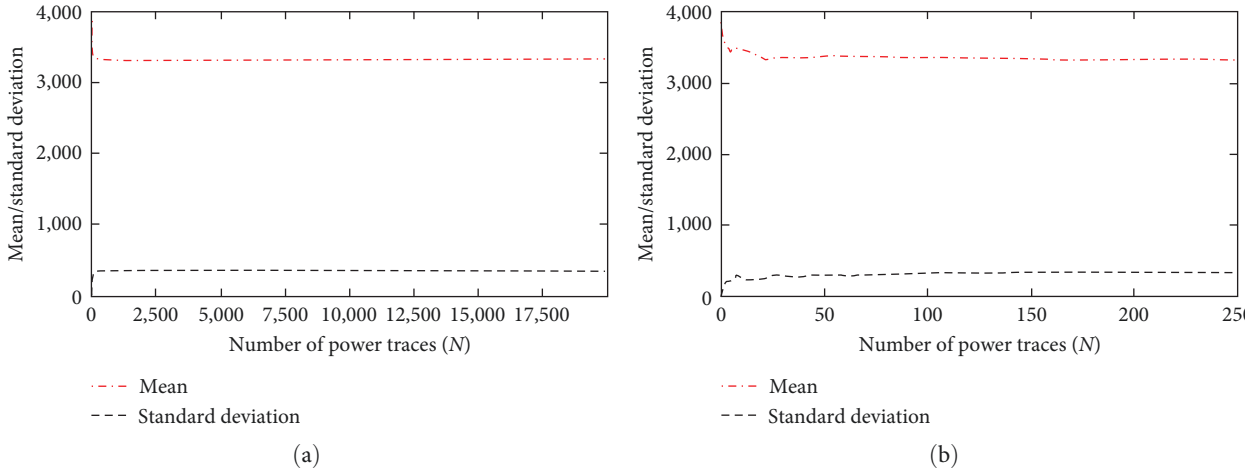

FIGURE 3: The variation of mean and standard deviation of 2,394-th sample point are represented by the red line and the black line, respectively. (a) Variation of mean/standard deviation with 20,000 power traces and (b) a close-up look at Figure 1(a). The horizontal axis represents the number of power traces. The vertical axis represents the value of mean and standard variation.

TABLE 2: The number of power traces for PPA with the constant $\varepsilon = 500$.

Byte	0	1	2	3	4	5	6	7
Traces	399	1,096	1,078	408	603	604	1,352	188
Byte	8	9	10	11	12	13	14	15
Traces	537	65	1,327	597	562	568	1,598	1,946

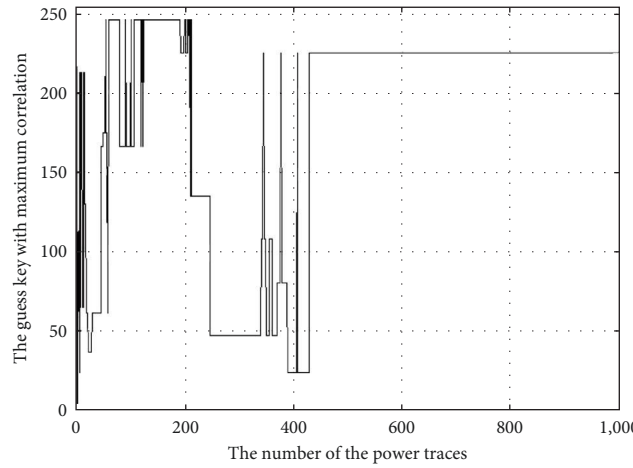

FIGURE 4: The maximum correlation for the 15-th subkey of CT-CPA for the public databases of DPA Contest V2. The horizontal axis represents the number of the power traces. The vertical axis represents the guessed key with maximum correlation.

constant $\varepsilon = 500$ is shown in Table 2. According to Table 2, setting the constant ε to be 500 implies that the number of power traces required for a recovery of AES' subkeys is reduced compared to PPA with the constant $\varepsilon = 33$. In particular, the number of power traces required to recover the 0th, 3th, 4th, 5th, 7th, 8th, 9th, 11th, 12th, and 13th is less than 610. To summarize, the constant ε is chosen so that $(\sigma_W + \varepsilon)$ approximates the standard deviation of the union of $(-\infty, \mu_{t_{ct}} - 2\sigma_{t_{ct}})$ and $(\mu_{t_{ct}} + 2\sigma_{t_{ct}}, +\infty)$ of the normal distribution $N(\mu_{t_{ct}}, \sigma^2_{t_{ct}})$, which can improve the performance of both PPA and CT-CPA.

Finally, we empirically set the threshold value T_j, introduced in Section 3.2.1, to be 100 due to the following. For $N = 20,000$ power traces generated using a random secret key in DPA Contest V2, the correct value of the 15-th subkey is $0xE1$. However, by increasing the number of power traces from 246 to 339, the correlation of incorrect subkey value $0x2F$ remains maximum, which implies that this key byte is incorrectly recovered.

In the worst case, the number of power traces for which the correlation of a wrongly guessed subkey may be maximal equals 94, which justifies the use of $T_j = 100$ for identifying the correct subkey values. Figure 4 shows the worst case scenario, when the correlation of an incorrectly guessed subkey calculated by CT-CPA achieves the maximum as a function of the number power traces.

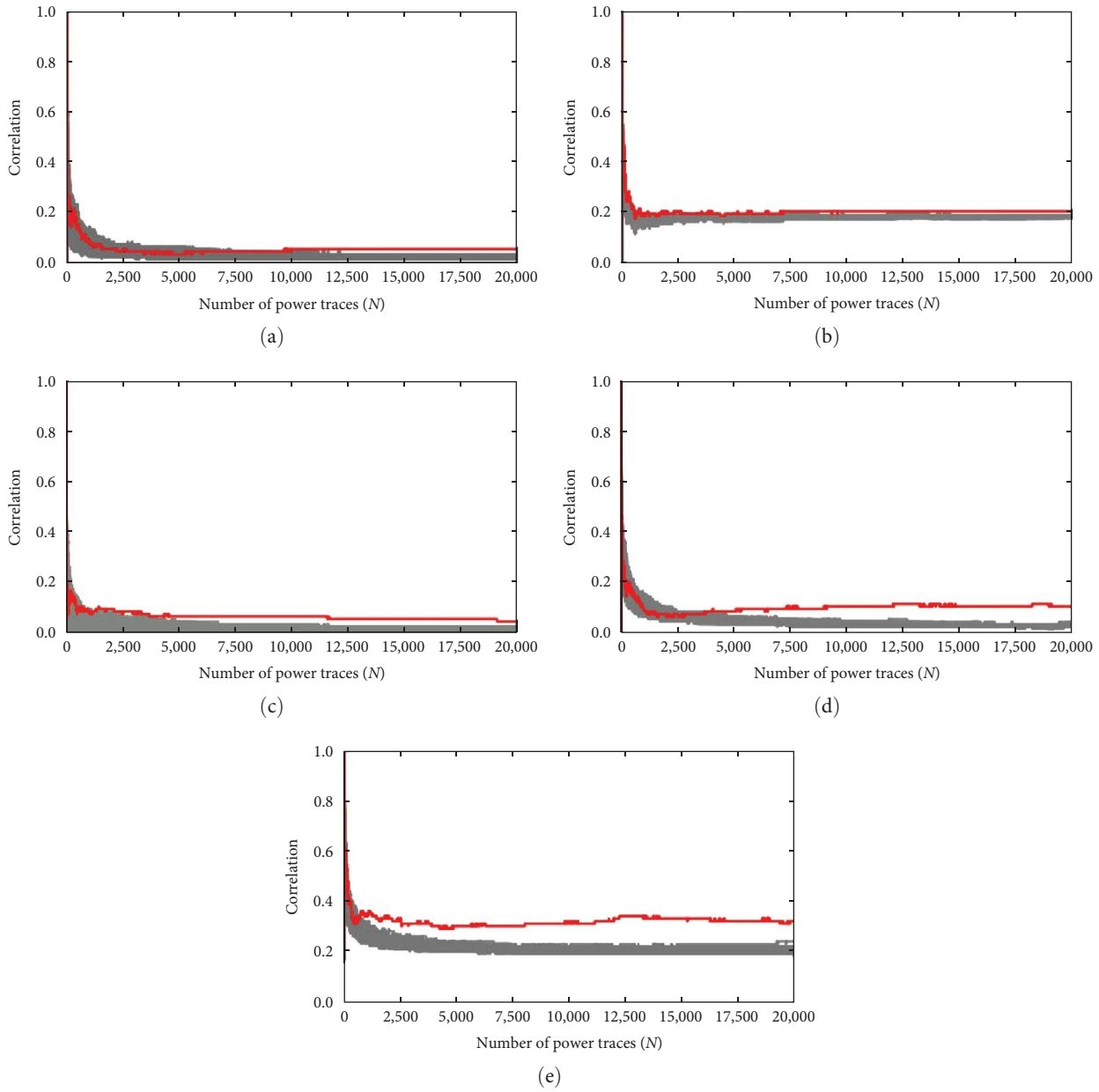

FIGURE 5: Variations of the correlations between guessed subkeys and the correct 15-th subkey for different methods. (a) CPA, (b) PPA ($\varepsilon = 33$), (c) BS-CPA, (d) BIAS-CPA, and (e) CT-CPA. The red and black lines represent the correlation to a correctly and wrongly guessed subkey, respectively.

Note that the value of $\alpha^*_{k,m}$ and $\alpha_{j,l,m}$ in Equation (8) are shown in Table 1.

4.3. Performance Evaluation. In this section, we analyze the performance of the mentioned attack methods in terms of the *required number of power traces* and *discrimination* which refers to the ratio between the correlation coefficient of the correct key candidate and of the wrongly guessed key whose correlation coefficient is maximal among the wrong guessed keys (see below for more details).

(1) Required Number of Power Traces.

This parameter refers to the minimum number of power traces required to successfully recover a subkey. Figure 5 shows the correlation variations for all guessed values for the 15-th subkey as the number of power traces increases, for different attack methods. According to Figure 5, to recover the 15-th subkey, CPA, PPA (with $\varepsilon = 33$), BS-CPA, and BIAS-CPA require 8,353, 3,771, 5,573, and 1,228 power traces, respectively. On the other hand, our CT-CPA method only needs 428 power traces for the same purpose, which is a substantial improvement over the other methods.

In addition, Table 3 shows a required number of power traces for recovering all the subkeys for these attack methods. According to Table 3, classical CPA, PPA (with $\varepsilon = 33$), BS-CPA, and BIAS-CPA require 12,688, 3,771, 10,808, and 3,472 power traces, respectively. There is a huge improvement of CT-CPA over the other methods since it only requires 495 power traces for recovering all the subkeys.

Table 3: The number of power traces required for recovering different subkeys for CPA [4], PPA ($\varepsilon = 33$) [26], BS-CPA [28], BIAS-CPA [27], and CT-CPA.

Bytes	CPA	PPA ($\varepsilon = 33$)	BS-CPA		BIAS-CPA	CT-CPA	
	Traces	Traces	Order	Traces	Traces	Order	Traces
0	7,157	1,129	9	8,524	359	6	197
1	5,667	1,265	4	3,244	970	3	120
2	12,688	1,091	14	10,631	2,748	12	104
3	2,394	734	0	2,394	328	1	59
4	10,869	663	15	10,808	3,472	15	495
5	3,423	3,414	5	3,433	333	13	315
6	6,934	2,819	6	3,380	2,296	7	310
7	9,223	2,024	12	7,367	140	2	77
8	6,410	1,596	2	2,174	423	4	215
9	3,676	1,980	3	3,183	34	0	65
10	9,697	1,570	13	8,009	415	8	295
11	4,560	1,649	1	2,356	720	5	234
12	4,699	3,650	8	5,583	647	10	405
13	10,946	1,083	10	4,930	319	11	411
14	5,540	2,894	7	4,577	601	9	361
15	8,353	3,771	11	5,573	1,228	14	428

Table 4: The number of power traces to recover the public databases of DPA Contest V2 submitted by the participants [38].

Participant/team	Number of traces
Nonprofiling SCA attacks	
Matthieu Walle	7,061
Victor Lomne	10,666
Mael Berthier,Yves Bocktaels	10,796
Autoine Wrucker	13,474
Mael Berthier	15,943
Alexis Bonnecaze	18,458
CT-CPA	495
Profiling SCA attacks	
Benoit Gerard, Nicolas Veyrat-Charvillon	439
Yang Li, Daisuke Nakatsu, Kazuo Sakiyama	2,256
Annelie Heuser, Michael Kasper, Werner	3,589

Table 4 shows the results of attacking the public databases of DPA Contest V2 submitted by participants, which can be obtained from [38]. As evident from Table 4, the number of power traces needed for the best nonprofiling SCA attack is 7,061. However, our CT-CPA only requires 495 power traces to recover AES's secret key for the public database of DPA Contest V2, which is a huge improvement compared to 7061. In addition, for the profiling SCA attacks, Benoit Gerard's method (mentioned in [38]) performs best and needs only 439 power traces. Since in general profiling SCA attacks perform better than that nonprofiling SCA variants, Tables 3 and 4 show that CT-CPA can significantly reduce the gap in the performance of these two classes of side-channel attacks.

(2) Discrimination.

For the purpose of evaluating the performance of different CPA methods, we introduce a new concept called *discrimination*. In brief, the discrimination describes the ratio between the correlation coefficient of the correct subkey candidate and of the maximum correlation value among incorrect subkey values. The following easy conclusions can be immediately deduced given this ratio:

(i) If discrimination is greater than 1, then the correct subkey value attains the maximum correlation, which implies that the considered subkey is successfully recovered.

(ii) If discrimination is less than 1, there exists an incorrectly guessed subkey having the maximum correlation, thus implying a failure in the process of this subkey recovery.

(iii) In general, a larger discrimination value implies a larger distance between the correlation of a correctly guessed subkey and of a wrong one, which gives a better resolution when distinguishing correct and wrong subkey values.

One can also consider the dependency of discrimination on the increased number of power traces. Especially, with respect to the recovery of the 15-th subkey we provide (in this context) a performance comparison of different attack methods in Figure 6 with the following conclusions:

(i) Discrimination of CT-CPA and PPA are greater than that of BS-CPA, CPA, and BIAS-CPA. It shows that CT-CPA and PPA are more successful in distinguishing the correct subkey than the other methods.

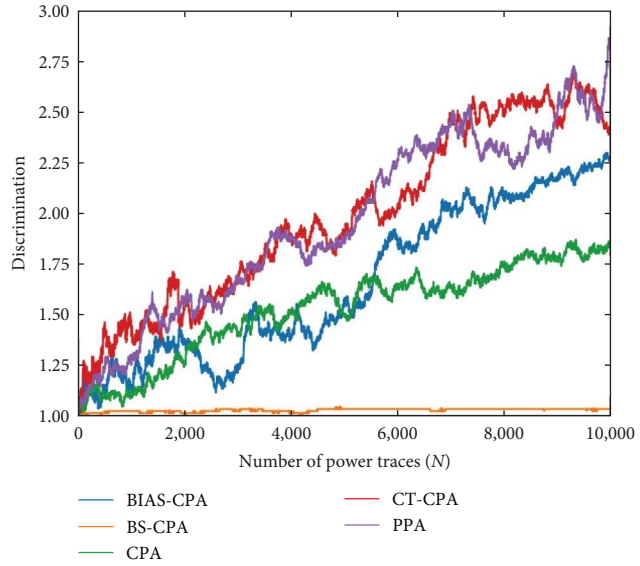

FIGURE 6: Variations in the discrimination of the 15-th subkey as a function of power traces for different methods.

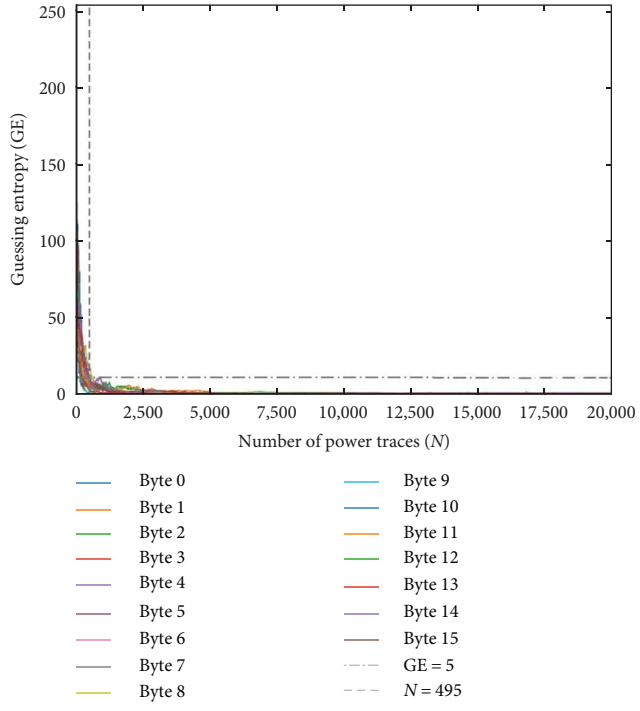

FIGURE 7: Guessing entropy for subkey bytes #1–#16 of CT-CPA.

(ii) Discrimination of CT-CPA and PPA remains roughly invariant when the number of power traces increases, assuming that the previous subkey has been successfully recovered. Therefore, CT-CPA and PPA perform similarly in terms of discrimination (resolution) but our approach requires much less power traces compared to PPA.

(iii) Discrimination for BS-CPA is always approximately 1 as the number of power traces increases, indicating that there always exist certain subkey values whose correlation remain the same as that of the correct one. This means that when the SNR of the power traces is relatively low then the BS-CPA method can perform quite badly, which then indicates that CT-CPA has a better discrimination ability than BS-CPA.

(3) Guessing Entropy.

During attacking each subkey of AES in DPA Contest V2, we sort all guessed keys in descending order based on

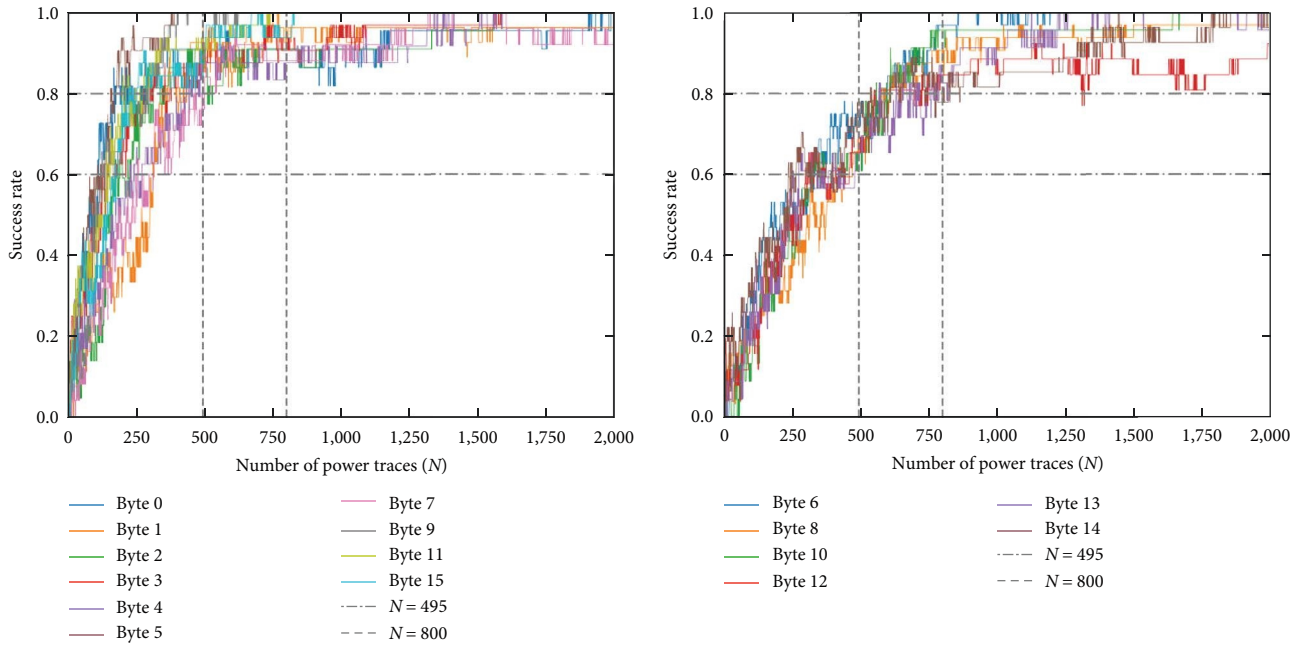

FIGURE 8: Success rate for all subkey bytes: (a) 0, …, 9, 11, 15 and (b) 6, 8, 10, 12, 13, 14 of AES using CT-CPA.

their correlation with N power traces. The guessing entropy, refers to as GE, indicates that the ranking of the correct guessed key among all guessed keys. The range of GE is from 0 to 255, where 0 represents this subkey is successfully recovered. We used CT-CPA to attack 6.4-million power traces of 32 keys (20,000 power traces for a key), and obtain the guessing entropy for each subkey byte, as shown in Figure 7. From Figure 7, we can see that when the number of power traces increases to about 495, the guessing entropy of the correct guessed key for all subkey bytes is less than 5. That is to say, when the number of power traces increases to 495, we can reduce the space complexity of guessed keys from 2^{128} to 2^{37}. In this case, we can obtain the correct key of AES through exhaustive search.

(4) Success Rate.

This parameter indicates the success probability of recovering a subkey or all keys of a cryptographic algorithm. Figure 8 shows that the success rate of recovering each subkey using our CT-CPA as the power traces increase from 0 to 2,000. From Figure 8, we can find that when the power traces increases to 495, the success rate of subkey bytes 0, …, 9, 11, and 15 can reach 80% (as shown in Figure 8(a)), and the success rate of the ramaining subkey bytes can reach 60%. In addition, we can discover that when the power traces increase to 800, the success rate of all subkeys ranges from 60% to 80% (as shown in Figure 8(b)).

5. Conclusions

In this paper, we have given an overview of the improved CPA methods and simulated their performance using the public databases of DPA Contest V2. To further improve the performance of CPA attacks, we have introduced a new CPA method (abbreviated as CT-CPA) which is suitable in the scenarios when cryptographic algorithm use parallel implementation of S-boxes. With respect to the public databases of DPA Contest V2, the experimental results show that CT-CPA only requires 495 power traces for recovering the secret key of AES which is a huge improvement over the other CPA techniques.

Appendix

In this section, we give a detailed derivation of Equation (8). It is worth noting that the symbols appearing below have the same meaning in the appendix as in the text. Equation (3) shows that the equation for calculating the correlation coefficient of the guessed subkey $k_{j,l}$ by CPA is,

$$r_{j,l} = \frac{N\sum_{i=1}^{N} W(P_i)h_{j,l,i} - \sum_{i=1}^{N} W(P_i)\sum_{i=1}^{N} h_{j,l,i}}{\sqrt{N\sum_{i=1}^{N} W(P_i)^2 - \left(\sum_{i=1}^{N} W(P_i)\right)^2} \cdot \sqrt{N\sum_{i=1}^{N} h_{j,l,i}^2 - \left(\sum_{i=1}^{N} h_{j,l,i}\right)^2}}. \tag{A.1}$$

Suppose that CT-CPA is used to estimate the subkey value $k_{j,l}$, assuming that p correct subkey values have been recovered. Considering the impact of the p correct subkey values on calculating the correlation coefficient of the candidate subkey $k_{j,l}$ (for the j-th subkey), the correlation coefficient $r'_{j,l}$ is expressed as follows:

$$r'_{j,l}=\frac{N\sum_{i=1}^{N}W(P_i)\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)}{\sqrt{N\sum_{i=1}^{N}W(P_i)^2-(\sum_{i=1}^{N}W(P_i))^2}\cdot\sqrt{N\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)^2-\left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)\right)^2}}. \tag{A.2}$$

We denote the nominator $A:=N\sum_{i=1}^{N}W(P_i)(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i})-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i})$ of the correlation coefficient $r'_{j,l}$ with $p\times(d+1)$ classes $G^*_{1,0},\ldots,G^*_{1,d},\ldots,G^*_{p,0},\ldots,G^*_{p,d}$ and $(d+1)$ classes $G_{j,l,0},\ldots,G_{j,l,d}$. Then, A becomes:

$$\begin{aligned}
A &= N\sum_{i=1}^{N}W(P_i)\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)\\
&=\left[N\sum_{i=1}^{N}\left(W(P_i)\sum_{k=1}^{p}h^*_{k,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}\right)\right]+\left[N\sum_{i=1}^{N}\left(W(P_i)h_{j,l,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}h_{j,l,i}\right]\\
&=\sum_{k=1}^{p}\left[N\sum_{i=1}^{N}\left(W(P_i)h^*_{k,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}h^*_{k,i}\right]+\left[N\sum_{i=1}^{N}\left(W(P_i)h_{j,l,i}\right)-\sum_{i=1}^{N}W(P_i)\sum_{i=1}^{N}h_{j,l,i}\right]\\
&=\sum_{k=1}^{p}\left[N\sum_{m=0}^{d}\sum_{G^*_{k,m}}(W(P_i)\cdot m)-\left(\sum_{m=0}^{d}\sum_{G^*_{k,m}}W(P_i)\right)\left(\sum_{n=0}^{d}\sum_{G^*_{k,n}}n\right)\right]\\
&\quad+\left[N\sum_{m=0}^{d}\sum_{G_{j,l,m}}(W(P_i)\cdot m)-\left(\sum_{m=0}^{d}\sum_{G_{j,l,m}}W(P_i)\right)\left(\sum_{n=0}^{d}\sum_{G_{j,l,n}}n\right)\right]\\
&=\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(N\cdot m\sum_{G^*_{k,m}}W(P_i)\right)-\left(\sum_{m=0}^{d}\sum_{G^*_{k,m}}W(P_i)\right)\left(\sum_{n=0}^{d}\left(N^*_{k,n}\cdot n\right)\right)\right]\\
&\quad+\left[\sum_{m=0}^{d}\left(N\cdot m\sum_{G_{j,l,m}}W(P_i)\right)-\left(\sum_{m=0}^{d}\sum_{G_{j,l,m}}W(P_i)\right)\left(\sum_{n=0}^{d}\left(N_{j,l,n}\cdot n\right)\right)\right]\\
&=\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(N\cdot m-\sum_{n=0}^{d}\left(N^*_{k,n}\cdot n\right)\right)\sum_{G^*_{k,m}}W(P_i)\right]+\left[\sum_{m=0}^{d}\left(N\cdot m-\sum_{n=0}^{d}\left(N_{j,l,n}\cdot n\right)\right)\sum_{G_{j,l,m}}W(P_i)\right].
\end{aligned} \tag{A.3}$$

Let us denote $\alpha^*_{k,m}=\frac{N^*_{k,m}}{N}(m-\sum_{n=0}^{d}\frac{N^*_{k,n}}{N}\cdot n)$ and $\alpha_{j,l,m}=\frac{N_{j,l,m}}{N}(m-\sum_{n=0}^{d}\frac{N_{j,l,n}}{N}\cdot n)$. The term A becomes:

$$A=\sum_{k=1}^{p}\left[N^2\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}}W(P_i)}{N^*_{k,m}}\right)\right]+\left[N^2\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}}W(P_i)}{N_{j,l,m}}\right)\right]. \tag{A.4}$$

The correlation coefficient $r'_{j,l}$ becomes:

$$\begin{aligned}
r'_{j,l} &= \frac{\sum_{k=1}^{p}\left[N^2\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}}W(P_i)}{N^*_{k,m}}\right)\right]+\left[N^2\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}}W(P_i)}{N_{j,l,m}}\right)\right]}{\sqrt{N\sum_{i=1}^{N}W(P_i)^2-(\sum_{i=1}^{N}W(P_i))^2}\cdot\sqrt{N\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)^2-\left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)\right)^2}}\\
&=\frac{\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}}W(P_i)}{N^*_{k,m}}\right)\right]+\left[\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}}W(P_i)}{N_{j,l,m}}\right)\right]}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}W(P_i)^2-\frac{1}{N^2}(\sum_{i=1}^{N}W(P_i))^2}\cdot\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)^2-\frac{1}{N^2}\left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p}h^*_{k,i}+h_{j,l,i}\right)\right)^2}}.
\end{aligned} \tag{A.5}$$

Let $\sigma_W = \sqrt{\frac{1}{N}\sum_{i=1}^{N} W(P_i)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N} W(P_i)\right)^2}$ and $\sigma'_H = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h^*_{k,i} + h_{j,l,i}\right)^2 - \frac{1}{N^2}\left(\sum_{i=1}^{N}\left(\sum_{k=1}^{p} h^*_{k,i} + h_{j,l,i}\right)\right)^2}$.

The correlation coefficient $r'_{j,l}$ can be written as follows:

$$r'_{j,l} = \frac{\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}} W(P_i)}{N^*_{k,m}}\right)\right] + \left[\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}} W(P_i)}{N_{j,l,m}}\right)\right]}{\sigma_W \cdot \sigma'_H}. \tag{A.6}$$

Since CT-CPA improves the performance of CPA by adding a constant ε to the standard deviation of the power traces σ_W, the correlation coefficient of CT-CPA $r_{j,l,\text{new}}$ is expressed as follows:

$$r_{j,l,\text{new}} = \frac{\sum_{k=1}^{p}\left[\sum_{m=0}^{d}\left(\alpha^*_{k,m}\frac{\sum_{G^*_{k,m}} W(P_i)}{N^*_{k,m}}\right)\right] + \left[\sum_{m=0}^{d}\left(\alpha_{j,l,m}\frac{\sum_{G_{j,l,m}} W(P_i)}{N_{j,l,m}}\right)\right]}{(\sigma_W + \varepsilon) \cdot \sigma'_H}. \tag{A.7}$$

Data Availability

The [DATA TYPE] data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The works of F. Yao and H. Chen are supported by the National Natural Science Foundation of China under Grant (No. 62172395). The works of Enes Pasalic are supported in part by the Slovenian Research Agency (research program P1-0404 and research projects J1-1694, N1-0159, J1-2451, and J1-4084). The works of Yongzhuang Wei are supported in part by the Natural Science Foundation of China (No. 61872103), in part by the Guangxi Natural Science Foundation (2019GXNS-FGA245004).

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO'96*, N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113, Springer, Berlin, Heidelberg, 1996.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO' 99*, M. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer, Berlin, Heidelberg, 1999.

[3] B.-S. Go, D.-V. Le, M.-G. Song, M. Park, and I.-K. Yu, "Design and electromagnetic analysis of an induction-type coilgun system with a pulse power module," *IEEE Transactions on Plasma Science*, vol. 47, no. 1, pp. 971–976, 2019.

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES 2004*, M. Joye and J. J. Quisquater, Eds., vol. 3156 of *Lecture Notes in Computer Science*, pp. 16–29, Springer, Berlin, Heidelberg, 2004.

[5] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems—CHES 2008*, E. Oswald and P. Rohatgi, Eds., vol. 5154 of *Lecture Notes in Computer Science*, pp. 426–442, Springer, Berlin, Heidelberg, 2008.

[6] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2002*, B. S. Kaliski, Ç. K. Koç, and C. Paar, Eds., vol. 2523 of *Lecture Notes in Computer Science*, pp. 13–28, Springer, Berlin, Heidelberg, 2003.

[7] W. Schindler, K. Lemke, and C. Paar, "A stochastic model for differential side channel cryptanalysis," in *Cryptographic Hardware and Embedded Systems—CHES 2005*, J. R. Rao and B. Sunar, Eds., vol. 3659 of *Lecture Notes in Computer Science*, pp. 30–46, Springer, Berlin, Heidelberg, 2005.

[8] G. Hospodar, B. Gierlichs, E. De Mulder, I. Verbauwhede, and J. Vandewalle, "Machine learning in side-channel analysis: a first study," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 293–302, 2011.

[9] F.-X. Standaert and C. Archambeau, "Using subspace-based template attacks to compare and combine power and electromagnetic information leakages," in *Cryptographic Hardware and Embedded Systems—CHES 2008*, E. Oswald and P. Rohatgi, Eds., vol. 5154 of *Lecture Notes in Computer Science*, pp. 411–425, Springer, Berlin, Heidelberg, 2008.

[10] J. G. J. van Woudenberg, M. F. Witteman, and B. Bakker, "Improving differential power analysis by elastic alignment," in *Topics in Cryptology—CT-RSA 2011*, A. Kiayias, Ed., vol. 6558 of *Lecture Notes in Computer Science*, pp. 104–119, Springer, Berlin, Heidelberg, 2011.

[11] L. Batina, J. Hogenboom, and J. G. J. van Woudenberg, "Getting more from PCA: first results of using principal component analysis for extensive power analysis," in *Topics in Cryptology—CT-RSA 2012*, O. Dunkelman, Ed., vol. 7178 of *Lecture Notes in Computer Science*, pp. 383–397, Springer, Berlin, Heidelberg, 2012.

[12] S. M. Del Pozo and F.-X. Standaert, "Blind source separation from single measurements using singular spectrum analysis," Cryptology ePrint Archive Report 2016/314, 2016.

[13] L. Wei, B. Luo, Y. Li, Y. Liu, and Q. Xu, "I know what you see: power side-channel attack on convolutional neural network accelerators," in *ACSAC '18: Proceedings of the 34th Annual Computer Security Applications Conference*, pp. 393–406, Association for Computing Machinery, December 2018.

[14] A. Heuser, O. Rioul, and S. Guilley, *Good is not good enough: deriving optimal distinguishers from communication theory*, Cryptology ePrint Archive Report 2014/527, 2014.

[15] S. Guilley, A. Heuser, and O. Rioul, "A key to success—success exponents for side-channel distinguishers," in *Progress in Cryptology—INDOCRYPT 2015*, A. Biryukov and V. Goyal, Eds., vol. 9462 of *Lecture Notes in Computer Science*, pp. 270–290, Springer, Cham, 2015.

[16] E. de Chérisey, S. Guilley, O. Rioul, and P. Piantanida, "Best information is most successful: mutual information and success rate in side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, pp. 49–79, 2019.

[17] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F.-X. Standaert, "Simpler and more efficient rank estimation for side-channel security assessment," in *Fast Software Encryption. FSE 2015*, G. Leander, Ed., vol. 9054 of *Lecture Notes in Computer Science*, pp. 117–129, Springer, Berlin, Heidelberg, 2015.

[18] L. David and A. Wool, "A bounded-space near-optimal key enumeration algorithm for multi-dimensional side-channel attacks," Cryptology ePrint Archive Report 2015/1236, 2015.

[19] D. P. Martin, L. Mather, and E. Oswald, "Two sides of the same coin: counting and enumerating keys post side-channel attacks revisited," in *Topics in Cryptology—CT-RSA 2018*, N. Smart, Ed., vol. 10808 of *Lecture Notes in Computer Science*, pp. 394–412, Springer, Cham, 2018.

[20] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert, "Security evaluations beyond computing power," in *Advances in Cryptology —EUROCRYPT 2013*, T. Johansson and P. Q. Nguyen, Eds., vol. 7881 of *Lecture Notes in Computer Science*, pp. 126–141, Springer, Berlin, Heidelberg, 2013.

[21] D. J. Bernstein, T. Lange, and C. van Vredendaal, "Tighter, faster, simpler side-channel security evaluations beyond computing power," Cryptology ePrint Archive Report 2015/221, 2015.

[22] B. Timon, "Non-profiled deep learning-based side-channel attacks," Cryptology ePrint Archive Report 2018/196, 2018.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[24] Z. Martinasek, P. Dzurenda, and L. Malina, "Profiling power analysis attack based on mlp in DPA contest V4.2," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 223–226, IEEE, Vienna, Austria, June 2016.

[25] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Cryptographic Hardware and Embedded Systems—CHES 2011*, B. Preneel and T. Takagi, Eds., vol. 6917 of *Lecture Notes in Computer Science*, pp. 49–62, Springer, Berlin, Heidelberg, 2011.

[26] T.-H. Le, J. Clédière, C. Canovas, B. Robisson, C. Servière, and J.-L. Lacoume, "A proposition for correlation power analysis enhancement," in *Cryptographic Hardware and Embedded Systems—CHES 2006*, L. Goubin and M. Matsui, Eds., vol. 4249 of *Lecture Notes in Computer Science*, pp. 174–186, Springer, Berlin, Heidelberg, 2006.

[27] K. Yongdae, S. Takeshi, H. Naofumi et al., "Biasing power traces to improve correlation in power analysis attacks," in *Proceedings of Constructive Side-Channel Analysis and Secure Design - COSADE 2010*, pp. 77–80, Berlin, Heidelberg Springer, 2011.

[28] Y. Komano, H. Shimizu, and S. Kawamura, "BS-CPA: built-in determined sub-key correlation power analysis," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E93-A, no. 9, pp. 1632–1638, 2010.

[29] D. Oswald and C. Paar, "Improving side-channel analysis with optimal linear transforms," in *Smart Card Research and Advanced Applications. CARDIS 2012*, S. Mangard, Ed., vol. 7771 of *Lecture Notes in Computer Science*, pp. 219–233, Springer, Berlin, Heidelberg, 2013.

[30] Y. Kim and H. Ko, "Using principal component analysis for practical biasing of power traces to improve power analysis attacks," in *Information Security and Cryptology—ICISC 2013*, H. S. Lee and D. G. Han, Eds., vol. 8565 of *Lecture Notes in Computer Science*, pp. 109–120, Springer, Cham, 2014.

[31] Z. Zhang, L. Wu, A. Wang, Z. Mu, and X. Zhang, "A novel bit scalable leakage model based on genetic algorithm," *Security and Communication Networks*, vol. 8, no. 18, pp. 3896–3905, 2015.

[32] P. Robyns, P. Quax, and W. Lamotte, "Improving CEMA using correlation optimization," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 1, pp. 1–24, 2018.

[33] D. Kwon, H. Kim, and S. Hong, "Improving non-profiled side-channel attacks using autoencoder based preprocessing," Cryptology ePrint Archive, Report 2020/396, 2020.

[34] Y. Ding, Y. Shi, A. Wang, Y. Wang, and G. Zhang, "Block-oriented correlation power analysis with bitwise linear leakage: an artificial intelligence approach based on genetic algorithms," *Future Generation Computer Systems*, vol. 106, pp. 34–42, 2020.

[35] S. Press, W. Teukolsky, and B. Flannery, *Numerical recipes in* c^{++}, Cambridge University Press, 2nd edition, 2002.

[36] J. Xu, Y. Tang, Y. Wang, and X. Wang, "A practical side-channel attack of a lorawan module using deep learning," in *2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pp. 17–21, IEEE, Xiamen, China, October 2019.

[37] "DPA Contest," 2008/2009, http://www.dpacontest.org/.

[38] C. Clavier, J.-L. Danger, G. Duc et al., "Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest," *Journal of Cryptographic Engineering*, vol. 4, no. 4, pp. 259–274, 2014.