TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/10 pp586–598
DOI: 10.26599/TST.2018.9010047
Volume 23, Number 5, October 2018

Side-Channel Attacks in a Real Scenario

Ming Tang*, Maixing Luo, Junfeng Zhou, Zhen Yang, Zhipeng Guo, Fei Yan, and Liang Liu

Abstract: Existing Side-Channel Attacks (SCAs) have several limitations and, rather than to be real attack methods, can only be considered to be security evaluation methods. Their limitations are mainly related to the sampling conditions, such as the trigger signal embedded in the source code of the encryption device, and the acquisition device that serves as the encryption-device controller. Apart from it being very difficult for an attacker to add a trigger into the original design before making an attack or to control the encryption device, there is a big gap in the capacity of existing SCAs to pose real threats to cipher devices. In this paper, we propose a new method, the sliding window SCA (SW-SCA), which can be applied in scenarios in which the acquisition device is independent of the encryption device and for which the encryption source code requires no trigger signal or modification. First, we describe the main issues in existing SCAs, then we theoretically analyze the effectiveness and complexity of our proposed SW-SCA —a method that can incorporate a sliding-window mechanism into almost all of the existing non-profiled SCAs. The experimental results for both simulated and physical traces verify the effectiveness of the SW-SCA and the appropriateness of its theoretical complexity.

Key words: side-channel attack; sliding window; trigger mechanism; soft K-means

1 Introduction

The use of Side-Channel Attacks (SCAs) has become an effective avenue for obtaining secret information from cryptographic devices, which seriously threatens their security. In 1999, Kocher et al.[1] proposed the use of Differential Power Analysis (DPA) to successfully recover the key of a cryptographic algorithm by analyzing the relationship between the power consumption of the cryptographic device during encryption and the intermediate value of the cryptographic algorithm. In addition, electromagnetic emanation[2], timing[3], and other physical leakage also can be used in SCAs. Generally, SCAs can be divided into two categories: profiled and non-profiled. A profiled SCA has two phases: a profiling phase in which an adversary is provided with a training device for testing that allows him to characterize physical leakages and obtain a precise leakage model; and an online exploitation phase in which an attack is mounted against a similar target device to perform a secret key extraction. A non-profiled SCA only requires the latter phase and assumes a less precise leakage model, typically based on engineering intuition. Non-profiled SCAs include DPA, Correlation Power Analysis (CPA)[4], Mutual Information Analysis (MIA)[5], Variance Ratio (VR)[6], and Differential Cluster Analysis (DCA)[7]. Profiled SCAs include Template Attacks (TA)[8] and Stochastic Approach (SA)[9]. Products that meet high security requirements must undergo an evaluation before

• Ming Tang, Maixing Luo, Junfeng Zhou, Zhen Yang, Zhipeng Guo, and Fei Yan are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China. E-mail: m.tang@126.com; lmx2016@whu.edu.cn; zhoujf620@zju.edu.cn; 2014301500142@whu.edu.cn; whu_guozhipeng@163.com; yanfei@whu.edu.cn.
• Liang Liu is with Beijing Smart-Chip Microelectronics Technology Company Limited, Beijing 100192, China. E-mail: liuliang2@sgitg.sgcc.com.cn.
* To whom correspondence should be addressed.
Manuscript received: 2017-10-16; accepted: 2017-12-23

entering the marketplace, for example, using the Common Criteria (CC)[10] and ISO/IEC 17825:2016[11]. A CC evaluation at the highest assurance level for penetration attacks requires that the device be capable of resisting attacks with 1 million traces. Similarly, the ISO/IEC 17825:2016 (application note for international standard ISO/IEC 19790, sibling to NIST/FIPS 140-2) requires a resistance capability against side-channel analysis with 10 000 traces (level 3) and 100 000 traces (level 4).

Most existing SCAs are carried out in an ideal measurement environment, with a trigger mechanism added to the source code to activate the acquisition device, which indicates where the start and end signals are in each encryption process. However, this approach is not applicable to real scenarios, because the source code can rarely be modified by an adversary. However, in a security evaluation, a trigger can be added to the source code, for the security assessment often has to consider the worst cases.

The main purpose of this paper is to move SCAs towards the realistic attack scenario. We consider an attack scenario in which an encryption target continuously encrypts a set of plaintexts, while an acquisition device works independently to alternately acquire and post power consumption data (referred to as curves). But without any trigger to control the cryptophytic device and activate the acquisition device, only partial and discontinuous power consumption data generated by the cryptographic device can be sampled. This leads to uncertainty about the correspondence between the plaintext and curve. Thus, all existing SCAs based on this correspondence will fail. In the above attack scenario, we assume that (1) the working cycle of the cryptographic device, based on its operation speed, is often public information, and (2) all of the plaintext or ciphertext of the entire encryption process can be acquired since the attacker can eavesdrop on the communication channel. We propose a method, the sliding-window SCA (SW-SCA), whereby a sliding window mechanism can be integrated into almost all the existing non-profiled SCAs to deal with the proposed attack scenario. Taking the sliding-window CPA (SW-CPA) as an example, we explain how to transform the traditional SCA into an SW-SCA that can handle this attack scenario. In addition, we propose a soft K-means preprocessing method that can improve the attack efficiency of CPA or SW-CPA.

The remainder of this paper is organized as follows. In Section 2, we describe the SW-SCA. In Section 3, we present two methods for handling attacks against no-trigger samples, the SW-CPA and SSW-CPA, which combine SW-CPA with soft K-means preprocessing. In Section 4, we analyze the complexity of the SW-SCA and estimate the number of curves required to successfully recover the key. Lastly, to verify that our analysis is consistent with reality, in Section 5, we perform several experiments on the SASEBO-W[12] and SASEBO-GII evaluation boards. We draw our conclusions in Section 6.

2 Sliding Window SCA

2.1 No-trigger sampling scenario

In the no-trigger sampling scenario shown in Fig. 1, there is an acquisition device under the control of a Personal Computer (PC), such as an oscilloscope as well as an encryption device, and these devices work independently of each other .

In this scenario, since the sampling rate of the oscilloscope is f_s, the sampling period is $\Delta = 1/f_s$. The encryption device sequentially performs the Advanced Encryption Standard (AES) on the N plaintexts $P_0, P_1, P_2, \ldots, P_{N-1}$. The encryption cycle is $T_0\Delta$, so, ideally, T_0 points can be sampled during each encryption period. After the encryption device works for an unknown period of time, the acquisition device starts to record the power consumption. The operation mode of the acquisition device is as follows: collect t_1 points, return data for $n_1\Delta$. As we can see, the working period of the acquisition device is $T_1\Delta = (t_1 + n_1)\Delta$, in which t_1 indicates the number of points collected, and n_1 indicates the number of points not collected. Assume that the total working time of the

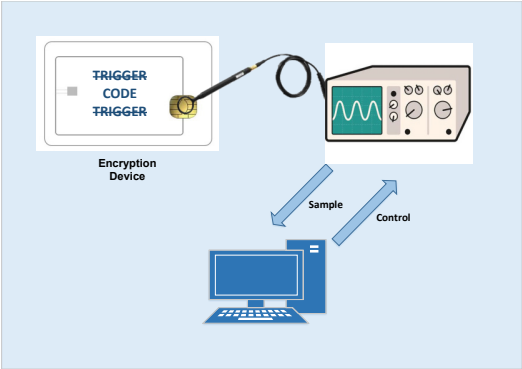

Fig. 1 Scenario of no-trigger sampling.

acquisition equipment is $L\Delta$. Since the acquisition device requires a certain time interval to return the samples, we use NaN (not a number) to denote the data corresponding to this interval. Then L sample points are divided into segments by the length of T_0 points, with each segment referring to a curve. There are a total of $M = L/T_0$ curves, and these curves are stored in a matrix $G_{M\times T_0}$ of M rows and T_0 columns, as shown in Fig. 2.

2.2 Analysis of no-trigger samples

According to the scenario described above, the curve has the following three characteristics:

(a) The curves may contain power consumption data of two plaintexts. This is because the acquisition and encryption devices work independently, and there is no guarantee that the acquisition device will start working from the beginning of a certain encryption process. We only consider all of the samples divided into curves by the length of T_0, because one curve at most covers two encryption processes, which leads to a situation in which one part of one curve comprises the power consumption data of one plaintext and another part comprises that of another plaintext.

(b) The curves contain missing data, as denoted by NaN. This is due to the fact that the acquisition device requires a certain time interval to return the samples.

(c) The correspondence between the curve and the plaintext is uncertain, because the acquisition and cryptographic devices work independently.

These three differences above mean that the original SCA cannot directly handle the current samples for the following reasons:

(1) The correspondence between the plaintext and the power curve is uncertain, so that the key cannot be recovered directly via the correlation between the power consumption and intermediate value.

(2) Due to the unknown number of absent samples, the original SCA cannot deal with this situation.

(3) Since there is no guarantee that a curve contains power consumption data for only one plaintext, an attack relying on multiple points in one curve will fail.

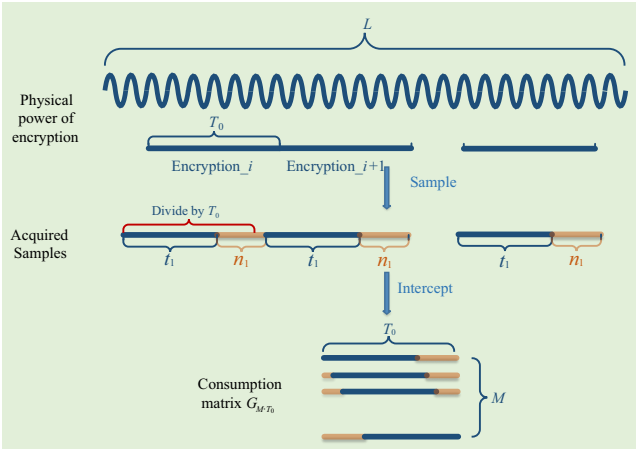

Fig. 2 No-trigger samples.

2.3 Steps of SW-SCA

To overcome the three difficulties described above, a new method is required.

To address the characteristic described above in Section 2.2a, since the number of time points for one encryption is T_0 and each curve also contains T_0 points, this ensures that each curve in the same position is doing the same operation. As such, there is no real impact on methods such as CPA, MIA, and VR that do not require the use of multiple power points on each curve. That is, for these methods, the curve has been aligned.

Regarding missing data denoted as NaN, as described in Section 2.2b, any data with the value NaN can be directly discarded. In the t-th column of the power consumption matrix $G_{M\times T_0}$, the number of samples actually used to calculate the correlation coefficient is not M, but $\lambda_t M$. λ_t is the proportion of non-NaN data in this column, which is related to the sizes of T_0, T_1, t_1, and n_1, which is discussed below in Section 4.1.

To overcome the uncertainty described in Section 2.2c, we propose the sliding window mechanism. Although the correspondence between the curves and the plaintexts are unknown, this correspondence can be determined if we can determine the plaintext of the first curve. Because the plaintext is encrypted in sequence and the samples are collected in the same order, when it is determined that the first curve corresponds to the s-th plaintext, then we known that the second curve corresponds to the $(s+1)$-th plaintext, ..., and the M-th curve corresponds to the $(s+M-1)$-th plaintext. So, by determining to which plaintext the first curve corresponds, we can know the correspondence between other curves and plaintexts. We call this correspondence the Start Point of Encryption (SPE).

Here, we make necessary adjustments to the original non-profiled SCAs, such as CPA and propose the SW-SCA method for no-trigger samples. The steps of SW-SCA are as follows (as shown in Fig. 3).

(1) Divide L samples by the length of T_0 to obtain a total of M $(M = L/T_0)$ traces with each segment

representing one trace. With these traces, build a matrix $G_{M\times T_0}$, where T_0 is the time of one encryption.

(2) Calculate the predicted power consumption matrix $H_{N\times K}$ according to the leakage model using the known plaintext blocks $P_0, P_1, P_2, \ldots, P_{N-1}$, for each guessing subkey k ($k = 0, 1, 2, \ldots, K-1$). The k-th column of $H_{N\times K}$ corresponds to the subkey k.

(3) Calculate the distinguisher value for each column H_k of $H_{N\times K}$, each column G_t of $G_{M\times T_0}$ and the starting point s for each sliding window, where $H_k^s = [H_{s,k}, H_{s+1,k}, \ldots, H_{s+M+1,k}]$. In another words, H_k^s is a column vector consisting of M numbers taken from the s-th element of the k-th column of $H_{N\times K}$ and G_t is the t-th column of $G_{M\times T_0}$. $d(H_k^s, G_t)$ is the distinguisher, which can be represented as the correlation coefficient in CPA, the mean difference in DPA, or the mutual information in MIA. However, since some elements of G_t may be NaN, adjustments must be made to the calculation of the distinguisher value as follows: if the element $G_{j,t}$ in G_t is NaN, remove $G_{j,t}$ and the corresponding j-th element $H_{j,k}^s$ in H_k^s before calculating the value of the distinguisher.

(4) The correct subkey k^*, the position t^* of the target operation in the traces, and the plaintext P_{s^*} of the first curve are chosen based on the maximum correlation coefficient:

$$(k^*, t^*, s^*) = \text{Argmax}_{k,t,s} \left|\rho(H_k^s, G_t)\right|.$$

3 Two Attack Methods Against No-Trigger Samples

In this section, we will first use CPA as an example and transform it into SW-CPA, as described in Section 2, then combine soft K-means with SW-CPA to create another attack method SSW-CPA.

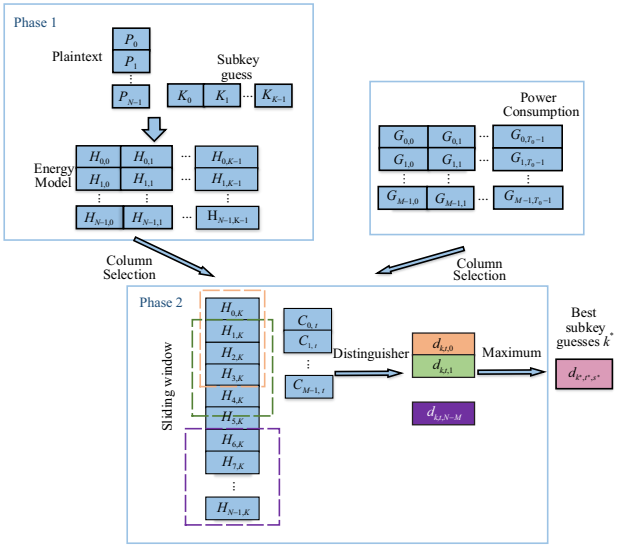

Fig. 3 Schematic of SW-SCA process.

Before analysis, we preprocess the collected power traces, divide the L samples $l_0, \ldots, l_{L-1}$ by the length of T_0 and obtain a matrix G of size $M \times T_0$. See Algorithm 1 for details regarding this preprocessing.

3.1 SW-CPA algorithm

3.1.1 Algorithm of SW-CPA

Based on the original CPA, we propose a new attack method, i.e., SW-CPA. The steps of SW-CPA in a real scenario are as follows:

Phase 1. For each guessing subkey k ($k = 0, 1, \ldots, K-1$), compute the intermediate value matrix $V_{N\times K}$ using plaintexts P_j ($j = 0, 1, \ldots, N-1$). Based on leakage model, the matrix $H_{N\times K}$ of the predicted power consumption is transformed from matrix $V_{N\times K}$, typically using the Hamming Weight (HW) or the Hamming Distance (HD).

Phase 2. Using the sliding window mechanism, compute the correlation coefficient $\rho(H_k^s, G_t)$ and obtain the matrix $R_{K\times T_0\times(N-M+1)}$. The size of the sliding window is equal to the number of M.

Phase 3. Choose the subkey k^* that maximizes the correlation coefficient as the correct subkey.

The time consumption mainly relies on the main function of the 8–15 line cycles. So the complexity of Algorithm 2 is $O(KL(N - L/T_0))$, where K is the number of guess subkeys, L is the total acquisition time, T_0 is the time of the once encryption, and N is the number of plaintext sets.

3.1.2 Correctness of SW-CPA

Assuming that the correct subkey is k_0, the position of the target operation in the traces is t_0 and the correspondence between the first curve and the plaintext is s_0, then $|\rho(H_{k_0}^{s_0}, G_{t_0})| > 0$ and most likely

Algorithm 1 Data_divide

Input: $T_0, L, l_0, \ldots, l_{L-1}$
Output: $G_{M\times T_0}$

```
M = L/T0;
/*Divide samples by T0*/
for i =0 to M − 1 do
    for j =0 to T0 − 1 do
        G_{i,j} = l_{i×T0+j};
    end
end
return G_{M×T0};
```

Algorithm 2 SW-CPA

Input: $G_{M\times T_0}, P_0 \cdots P_{N-1}, K$
Output: subkey

for $n = 0$ to $N-1$ **do**
  for $k = 0$ to $K-1$ **do**
    $V_{n,k} = \mathbf{Sbox}(P_n \oplus k)$;
    $H_{n,k} = \mathbf{HW}(V_{n,k})$;
  end
end
/*Compute the correlation coefficient*/
for $k = 0$ to $K-1$ **do**
  for $t = 0$ to $T_0 - 1$ **do**
    for $s = 0$ to $N - M$ **do**
      $R_{k,t,s} = \left|\rho(H_k^s, G_t)\right|$;
    end
  end
end
$(k^*, t^*, s^*) = \text{Argmax}_{k,t,s} R_{k,t,s}$;
subkey $= k^*$;
return subkey;

$|\rho(H_{k_0}^{s_0}, G_{t_0})| = \max_{k,t,s}|\rho(H_k^s, G_t)|$. The reason for this is as follows:

(1) When $s = s_0$, H_k^s and G_t are independent from each other. So $|\rho(H_k^s, G_t)| = 0$.

(2) When $s = s_0$ and $k \neq k_0$, the correlation between H_k^s and G_t is weak, especially after the S-box and other non-linear components. So $|\rho(H_k^s, G_t)| \approx 0$.

(3) When $s = s_0$, $k = k_0$, and $t \neq t_0$, the intermediate value of the target operation may appear elsewhere in the curve, resulting in $|\rho(H_k^s, G_t)| > 0$. But compared to $|\rho(H_{k_0}^{s_0}, G_{t_0})|$, it is generally smaller. So there is no impact on recovering the subkey, except that it affects the location of the target operation at the curve position.

(4) When $s = s_0$, $k = k_0$, and $t = t_0$, there is a strong correlation between H_k^s and G_t. So $|\rho(H_{k_0}^{s_0}, G_{t_0})| > 0$ and most likely $|\rho(H_{k_0}^{s_0}, G_{t_0})| = \max_{k,t,s}\left|\rho(H_k^s, G_t)\right|$.

3.2 SSW-CPA

In SCA, the Gaussian Leakage assumption[13] is usually considered to be true, which indicates that the leakage related to the intermediate values subjects to the Gaussian mixture distribution. And the leakage of the intermediate value typically has a Gaussian mixture distribution if there is only one leakage point and the intermediate value is unknown. In other words, not only the randomness of the noise, but also the randomness of the plaintext input must be taken into account. Soft K-means is an algorithm that can effectively cluster data that obeys a Gaussian mixture distribution. In this subsection, we introduce the soft K-means algorithm into CPA as a new preprocessing method, which we refer to as SSW-CPA. The theoretical analysis results show that this algorithm can improve the attack efficiency of CPA.

3.2.1 Introduction of soft K-means

Clustering divides a set of data objects (usually represented by a vector of values from a series of features) into several categories. Objects belonging to the same class are as similar as possible, and objects belonging to different classes are as different as possible[14]. To date, there have been just a few papers introducing a clustering algorithm into SCA. In DCA, indicators (such as the sum of squared error and the sum-of-squares) that originally were used to evaluate clustering effects are used as distinguishers[7]. With the help of K-means clustering, Heyszl et al.[15] succeeded in attacking the exponentiation algorithms used in public key cryptography based on one power consumption curve. And Whitnall and Oswald[14] also makes the use of K-means, hierarchical, and other clustering methods to improve the robustness of profiled DPA. In this paper, as a preprocessing method, we use an extended version of K-means known as soft K-means[16] to improve the attack efficiency of SW-CPA or the original CPA.

In SCA, we generally assume that power consumption is consistent with a Gaussian mixture distribution, whereas the soft K-means clustering can be used for data sets subjected to that distribution. Compared to K-means, one advantage of soft K-means is that it can return the possibility that each sample point belongs to each cluster. The soft K-means algorithm is given below.

We have N data objects, with each object being an I-dimensional vector, and the n-th data object is denoted by $x^{(n)}$. To divide the N data objects into K clusters, the mean value of the k-th cluster is $m^{(k)}$, the variance is σ_k^2, and the weight is π_k. Each data point $x^{(n)}$ is given a soft degree of assignment to each of the means. We call the degree to which $x^{(n)}$ is assigned to cluster k the responsibility $r_k^{(n)}$ (the responsibility of cluster k for point n).

Assignment step. The responsibility is determined as follows:

$$r_k^{(n)}=\frac{\pi_k \frac{1}{\left(\sqrt{2\pi\sigma_k^2}\right)^I}\exp\left(-\frac{1}{2\sigma_k^2}d(m^{(k)},x^{(n)})\right)}{\sum_{k'}\pi_{k'}\frac{1}{\left(\sqrt{2\pi\sigma_{k'}^2}\right)^I}\exp\left(-\frac{1}{2\sigma_{k'}^2}d(m^{(k')},x^{(n)})\right)},$$

where I is the dimensionality of x.

Update step. Each cluster's parameters $m^{(k)}$, π_k, and σ_k^2 are adjusted to match the data points for which it is responsible.

$$m^{(k)} = \frac{\sum_n r_k^{(n)} x^{(n)}}{R^{(k)}},$$

$$\sigma_k^2 = \frac{\sum_n r_k^{(n)}(x^{(n)}-m^{(k)})^2}{I R^{(k)}},$$

$$\pi_k = \frac{R^{(k)}}{\sum_{k'} R^{(k')}},$$

where $R^{(k)}$ is the total responsibility of mean k, $R^{(k)} = \sum_n r_k^{(n)}$.

3.2.2 Soft K-means preprocessing method

Let G be the leakage and H be the HW of the intermediate value. Assume that under the condition of $H = h$, G obeys a Gaussian distribution with mean μ_h and variance σ_h^2, i.e., the conditional distribution is as follows:

$$f_{G|H=h}(g) = \frac{1}{\sqrt{2\pi\sigma_h^2}}\exp\left(-\frac{(g-m_h)^2}{2\sigma_h^2}\right).$$

Assuming that we obtain N power consumption values $g_0, g_1, \ldots, g_{N-1}$ of the target position and the corresponding plaintext $P_0, P_1, \ldots, P_{N-1}$, what we want to know is the HW corresponding to each power consumption value, that is, we perform the calculation $\Pr(H = h|G = g_n)$. Under the above assumptions, we have

$$\Pr(H=h|G=g_n) = \frac{\Pr(H=h)\Pr(G=g_n|H=h)}{\sum_{h'}\Pr(H=h')\Pr(G=g_n|H=h')} = \frac{\Pr(H=h)\frac{1}{\sqrt{2\pi\sigma_h^2}}\exp\left(-\frac{(g_n-m_h)^2}{2\sigma_h^2}\right)}{\sum_{h'}\Pr(H=h')\frac{1}{\sqrt{2\pi\sigma_{h'}^2}}\exp\left(-\frac{(g_n-m_{h'})^2}{2\sigma_h^2}\right)}.$$

It is clear that the formula above has the same form as that for calculating responsibility $r_k^{(n)}$ in the soft K-means algorithm. In fact, responsibility $r_k^{(n)}$ can be used as an estimation of $\Pr(H = h|G = g_n)$, because the means and variances of clusters in soft K-means can all be used as valid estimates of the mean and variance in the power consumption in the Gaussian mixture model when the number of clustered data objects is relatively large. All in all, we can use soft K-means clustering to obtain the probability distribution of the HWs of the intermediate values corresponding to each power consumption value $\Pr(H = h|G = g_n)$.

The soft K-means preprocessing steps are as follows:

(1) Use the soft K-means algorithm to divide N power consumption value $g_0, g_1, \ldots, g_{N-1}$ into nine clusters, and obtain the mean m_k ($k = 0, 1, \ldots, 8$) of each cluster and the responsibility $r_k^{(n)}$ of each power consumption value.

(2) Convert the cluster label to the corresponding HW. If there are no other constraints, the number of correspondences between the cluster label and nine HWs is 9!. For the sake of simplicity, we can say that the greater is the HW, the greater is the within-class mean value. Thus, we only need to sort within-class mean value from small to large, and then map it to HW 0 to 8. In addition, we can obtain an estimate of the probability of the HW corresponding to each power consumption value $\Pr(H = h|G = g_n) = r_n^h$.

(3) According to the following formula, convert each g_n to g'_n.

$$g'_n = \sum_h h\Pr(H = h|G = g_n) = E(H|G = g_n).$$

3.2.3 Effectiveness of the soft K-means preprocessing method

In this section, we analyze the effectiveness of the soft K-means preprocessing method for CPA. Without preprocessing of the power consumption, we estimate $\rho(H, G)$ based on the sample values, whereas with preprocessing, we estimate $\rho(H, G')$, where $G' = E(H|G)$. In the following, we demonstrate that the correlation coefficient becomes larger after pretreatment, that is $|\rho(H, G)| \leqslant |\rho(H, G')|$, which indicates that the effect of a CPA attack after pretreatment will be improved. Proposition 1 below, as given in Ref. [17], is used to analyze the optimal prediction function of the second order DPA (i.e., the leakage model). Researchers[18] have cited Proposition 1, and proposed an excellent and simple method known as normalized inter-class variance.

We propose Proposition 2, an extension of

Proposition 1, which uses the properties of conditional expectation.

Proposition 1 Let X, Y be two random variables, and f be an arbitrary function defined in the value space χ of X, then we have

$$\rho(f(X),Y)=\rho(f(X),E(Y|X))\times\rho(E(Y|X),Y).$$

Lemma 1 Let X and Y be two random variables, then the conditional expectation have the following two properties:

(1) $E(E(g(X,Y)|X))=E(g(X,Y))$, particularly $E(E(Y|X))=E(Y)$.

(2) $E(g(X)Y|X)=g(X)E(Y|X)$.

Proposition 2 Let $X=(X_1,X_2,\ldots,X_n)$ and $Y=(Y_1,Y_2,\ldots,Y_n)$ be two random variables, f and g be two arbitrary functions. We then obtain the following:

$$\rho(f(X),g(Y))=$$
$$\rho(f(X),E(f(X)|Y))\times\rho(E(f(X)|Y),g(Y)).$$

Proof Using Lemma 1 repeatedly, and we have the following:

$$\begin{aligned}&\mathrm{Cov}(f(X),g(Y))=\\&E(f(X)g(Y))-E(f(X))\,E(g(Y))=\\&E(E(\,f(X)g(Y)\,|Y))-E(E(f(X)|Y))\,E(g(Y))=\\&E(E(f(X)|Y)g(Y))-E(E(f(X)|Y))\,E(g(Y))=\\&\mathrm{Cov}(E(f(X)|Y),g(Y)),\end{aligned}$$

and

$$\begin{aligned}&\mathrm{Cov}(f(X),E(f(X)|Y))=\\&E(f(X)\;E(f(X)|Y))-E(f(X))\;E(f(X)|Y)=\\&E(E(\;f(X)E(f(X)|Y)\;|Y))-\\&E(E(f(X)|Y))\;E(E(f(X)|Y))=\\&E(E(f(X)|Y)\;E(f(X)|Y))-\\&E(E(f(X)|Y))\;E(E(f(X)|Y))=\\&\mathrm{Var}(E(f(X)|Y)).\end{aligned}$$

Then we have

$$\begin{aligned}&\rho(f(X),E(f(X)|Y))\times\rho(E(f(X)|Y),g(Y))=\\&\frac{\mathrm{Cov}(f(X),E(f(X)|Y))}{\sqrt{\mathrm{Var}(f(X))\;\mathrm{Var}(E(f(X)|Y))}}\times\\&\frac{\mathrm{Cov}(E(f(X)|Y),g(Y))}{\sqrt{\mathrm{Var}(E(f(X)|Y))\mathrm{Var}(g(Y))}}=\\&\frac{\mathrm{Cov}(f(X),g(Y))}{\sqrt{\mathrm{Var}(H)\mathrm{Var}(E(f(X)|Y))}}\times\\&\frac{\mathrm{Cov}(f(X),g(Y))}{\sqrt{\mathrm{Var}(E(f(X)|Y))\mathrm{Var}(g(Y))}}=\\&\frac{\mathrm{Cov}(H,g(Y))}{\sqrt{\mathrm{Var}(f(X))\mathrm{Var}(g(Y))}}=\rho(f(X),g(Y)).\end{aligned}$$

And because of $|\rho(E(H|G),G)|\leqslant 1$, we have the following:

$$|\rho(H,G)|=|\rho(H,E(H|G))|\times|\rho(E(H|G),G)|\leqslant|\rho(H,E(H|G))|.$$

Corollary 1 When the consumption G becomes $G'=E(H|G)$ after soft K-means pretreatment, its correlation with the predicted power consumption H increase, that is $|\rho(H,G)|\leqslant|\rho(H,G')|$.

Proof By Proposition 2, we have the following:

$$\rho(H,G)=\rho(H,E(H|G))\times\rho(E(H|G),G).$$

Combined with $\rho(E(H|G),G)\leqslant 1$, we have the following:

$$|\rho(H,G)|=|\rho(H,E(H|G))|\times|\rho(E(H|G),G)|\leqslant|\rho(H,E(H|G)).$$

3.2.4 Steps of SSW-CPA

The soft K-means preprocessing method can also be combined with SW-CPA (SSW-CPA) to improve attack efficiency. The process of SSW-CPA process has the following three steps:

(1) Divide the collected curves into the power consumption matrices $G_{M\times T_0}$ described in Section 2.1;

(2) As the soft K-means preprocessing algorithm (Algorithm 3), apply the soft K-means preprocessing method to transform the power consumption matrix $G_{M\times T_0}$ into $G'_{M\times T_0}$.

(3) Use the SW-CPA algorithm to recover the subkey.

Here, we present the soft K-means preprocessing method for handling the data generated in the no-trigger scenario, which also requires that NaN cases be processed.

4 Complexity of SW-SCA

In this section, we analyze the complexity of SW- SCA.

Algorithm 3 Soft-kmeans-pre

Input: G, K
Output: G

```
for t = 0 to T_0 do
    temp =~ isnan(G(:,t)); //Select the subscript of the non-NaN value in G
    data = G(temp,t);
    [resp,center] = softkmeans(data,K);
    [~,hw2center] = sort(center);
    [~,center2hw] = sort(hw2center);
    center2hw = center2hw − 1
    for n = 1 to sum(temp) do
        data(n) = sum(resp(n,:) .* center2hw);
    end
    G(temp,t) = data;
end
return G;
```

We determine the data complexity by the number of encrypted plaintexts, which in the original SCA is known as the number of samples required[19]. We determine the time complexity by the time required to complete the entire attack process (including preprocessing).

4.1 Data complexity

Compared to the original SCA, the increased data complexity is due to the missing data, as indicated by NaN. When calculating the distinguisher value by the t-th column G_t of $G_{M\times T_0}$, we use only non-NaN values. If the proportion of the non-NaN values in G_t is recorded as λ_t, the actual number of samples used for the calculation is $\lambda_t M$. When the original SCA requires M_0 plaintext to perform a successful attack, $M_1 = \frac{1}{\lambda_{t_0}} M_0$ is required, where t_0 is the position of the leakage point that is most relevant to the intermediate value in the power curve. In other words, we only need to analyze the size of λ_t to determine the data complexity. In the following section, we analyze the relationship among λ_t, T_0, T_1, and t_1.

λ_t represents the proportion of non-NaN data in the t-th column G_t of the power consumption matrix $G_{M\times T_0}$. Let x be one of the L points collected in time order before the preprocessing dividing step and let it be denoted as $0, 1, 2, \ldots, L-1$. According to the data characteristics and processing methods, we can draw two conclusions:

$$x \text{ in } G_t \Leftrightarrow x = t \pmod{T_0} \tag{1}$$

$$x \neq \text{NaN} \Leftrightarrow x \equiv i \pmod{T_1} \wedge 0 \leqslant i \leqslant t_1 - 1 \tag{2}$$

Let $N_{t,i}$ be the number of solutions to the following congruence equations, where x is a value in $0, 1, 2, \ldots, L-1$, $t = 0, 1, 2, \ldots, T_0 - 1$, and $i = 0, 1, 2, \ldots, T_1 - 1$.

$$\begin{cases} x \equiv t \pmod{T_0}; \\ x \equiv i \pmod{T_1} \end{cases} \tag{3}$$

Let N_t be the number of solutions to the congruence equation $x \equiv t \pmod{T_0}$, then by Eqs. (1) and (2), we can deduce the following:

$$N_t = \sum_{i=0}^{T_1-1} N_{t,i}, \quad \lambda_t = \frac{\sum_{i=0}^{t_1-1} N_{t,i}}{N_t}.$$

Firstly, we deduce the number of solutions $N_{t,i}$ to the congruence system (Eq. (3)). In the following, we give two lemmas, and propose the proof of Theorem 1 based on these two lemmas. Lastly, we deduce the expression of λ_t using Theorem 1.

Lemma 2 Let a, b be integers, and T be a positive integer. Deduced by fundamental theorem of arithmetic, we have the following:

$T = p_1^{\alpha_1}, p_2^{\alpha_2}, \ldots, p_d^{\alpha_d}, \alpha_d \geqslant 0, h = 1, 2, \ldots, d$, where p_h $(h = 1, 2, \ldots, d)$ is prime, and we have the following:

$$a \equiv b \pmod{T} \Leftrightarrow \begin{cases} x \equiv b \pmod{p_1^{\alpha_1}}; \\ x \equiv b \pmod{p_2^{\alpha_2}}; \\ \vdots \\ x \equiv b \pmod{p_d^{\alpha_d}}. \end{cases}$$

Proof $p_h^{\alpha_h} | T$, and $T | a - b$, that is, the right can be derived from the left. Meanwhile, for all h, $p_h^{\alpha_h} | a - b$, so there is $T = \text{lcm}(p_1^{\alpha_1}, p_2^{\alpha_2}, \ldots, p_d^{\alpha_d}) | a - b$. The left can be derived from the right. Thus Lemma 2 is proved.

Lemma 3 Let a, b, c, α_1, and α_2 be integers, and p be a positive integer, so we have

$$\begin{cases} x \equiv a \pmod{p^{\alpha_1}}; \\ x \equiv b \pmod{p^{\alpha_2}} \end{cases} \Leftrightarrow \begin{cases} a \equiv b \pmod{p^{\min(\alpha_1,\alpha_2)}}; \\ x \equiv c \pmod{p^{\max(\alpha_1,\alpha_2)}}. \end{cases}$$

For c, if $\max(\alpha_1, \alpha_2) = \alpha_1$, then $c = a$; else $c = b$.

Proof Evidently $p^{\min(\alpha_1,\alpha_2)} | x - a$, and $p^{\min(\alpha_1,\alpha_2)} | x - b$, thus $p^{\min(\alpha_1,\alpha_2)} | a - b$, thereby the right can be derived from the left. On the other hand, because $p^{\min(\alpha_1,\alpha_2)} | p^{\max(\alpha_1,\alpha_2)}$ and $p^{\max(\alpha_1,\alpha_2)} | x - c$, we have $p^{\min(\alpha_1,\alpha_2)} | x - c$. Thus, $p^{\min(\alpha_1,\alpha_2)} | a - b$ and $p^{\min(\alpha_1,\alpha_2)} | x - c - a - b$ can be deduced. That is, the left can be derived from the right. Consequently, Lemma 3 is proved.

Theorem 1 Let t, i be integers, and T_0, T_1 be positive integers, the necessary and sufficient condition for the solutions to the congruence system (Eq. (3)) is

$$t = i \pmod{\gcd(T_0, T_1)}.$$

If a solution exists,then any two such solutions are congruent modulo $\text{lcm}(T_0, T_1)$. That is, if both x_1 and x_2 are the solutions to the congruence system (Eq. (3)), then we have

$$x_1 \equiv x_2 \pmod{\text{lcm}(T_0, T_1)}.$$

Proof According to the fundamental theorem of arithmetic, T_0 and T_1 can be decomposed as follows:

$$T_0 = p_1^{\alpha_1^0} p_2^{\alpha_2^0} \cdots p_d^{\alpha_d^0}, \ \alpha_h^0 \geqslant 0, \ h = 1, 2, \ldots, d;$$

$$T_1 = p_1^{\alpha_1^1} p_2^{\alpha_2^1} \cdots p_d^{\alpha_d^1}, \ \alpha_h^2 \geqslant 0, \ h = 1, 2, \ldots, d.$$

By Lemmas 2 and 3, we can deduce the following formula:

$$\begin{cases} x \equiv t \pmod{T_0}; \\ x \equiv i \pmod{T_1} \end{cases} \Leftrightarrow \begin{cases} x \equiv t \pmod{p_1^{\alpha_1^0}}; \\ x \equiv i \pmod{p_1^{\alpha_1^1}}; \\ x \equiv t \pmod{p_2^{\alpha_2^0}}; \\ x \equiv i \pmod{p_2^{\alpha_2^1}}; \\ \vdots \\ x \equiv t \pmod{p_d^{\alpha_d^0}}; \\ x \equiv i \pmod{p_d^{\alpha_d^1}}) \end{cases} \Leftrightarrow$$

$$\begin{cases} t \equiv i \pmod{p_1^{\min(\alpha_1^0,\alpha_1^1)}}; \\ x \equiv j_1 \pmod{p_1^{\max(\alpha_1^0,\alpha_1^1)}}; \\ t \equiv i \pmod{p_2^{\min(\alpha_2^0,\alpha_2^1)}}; \\ x \equiv j_2 \pmod{p_2^{\max(\alpha_2^0,\alpha_2^1)}}; \\ \vdots \\ t \equiv i \pmod{p_d^{\min(\alpha_d^0,\alpha_d^1)}}; \\ x \equiv j_d \pmod{p_d^{\max(\alpha_d^0,\alpha_d^1)}}; \end{cases}$$

in which formula, for all h, $j_h = t$ in the case of $\max(\alpha_h^0, \alpha_h^1) = \alpha_h^0$, otherwise $j_h = i$. With Lemma 2 and $\gcd(T_0, T_1) = p_1^{\min(\alpha_1^0,\alpha_1^1)} p_2^{\min(\alpha_2^0,\alpha_2^1)} \cdots p_d^{\min(\alpha_d^0,\alpha_d^1)}$, we have the following:

$$t \equiv i \pmod{\gcd(T_0, T_1)} \Leftrightarrow$$

$$\begin{cases} t \equiv i \pmod{p_1^{\min(\alpha_1^0,\alpha_1^1)}}; \\ t \equiv i \pmod{p_2^{\min(\alpha_2^0,\alpha_2^1)}}; \\ \vdots \\ t \equiv i \pmod{p_d^{\min(\alpha_d^0,\alpha_d^1)}}. \end{cases}$$

On the other hand, according to the Chinese remainder theorem, the following congruence equations must have solutions, and these solutions are congruent modulo $p_1^{\min(\alpha_1^0,\alpha_1^1)} p_2^{\min(\alpha_2^0,\alpha_2^1)} \cdots p_d^{\min(\alpha_d^0,\alpha_d^1)}$.

$$\begin{cases} x \equiv j_1 \pmod{p_1^{\max(\alpha_1^0,\alpha_1^1)}}; \\ x \equiv j_2 \pmod{p_2^{\max(\alpha_2^0,\alpha_2^1)}}; \\ \vdots \\ x \equiv j_d \pmod{p_d^{\max(\alpha_d^0,\alpha_d^1)}}. \end{cases}$$

And because $\operatorname{lcm}(T_0, T_1) = p_1^{\min(\alpha_1^0,\alpha_1^1)} p_2^{\min(\alpha_2^0,\alpha_2^1)} \cdots p_d^{\min(\alpha_d^0,\alpha_d^1)}$, these solutions also are congruent modulo $\operatorname{lcm}(T_0, T_1)$. In conclusion, Theorem 1 is proved.

At this point, we begin to use Theorem 1 to calculate λ_t. Based on Theorem 1, we know that, for any t ($0 \leqslant t \leqslant T_0 - 1$), i ($0 \leqslant i \leqslant T_0 - 1$), and any integer h, when x is in range $\{0 + h \cdot \operatorname{lcm}(T_0, T_1),\ 1 + h \cdot \operatorname{lcm}(T_0, T_1), \ldots, \operatorname{lcm}(T_0, T_1) - 1 + h \cdot \operatorname{lcm}(T_0, T_1)\}$, there is one solution or no solution to the following congruence equations:

$$\begin{cases} x \equiv t \pmod{T_0}; \\ x \equiv i \pmod{T_1}. \end{cases}$$

And the equivalent condition of solution is $t \equiv i \pmod{\gcd(T_0, T_1)}$. Therefore, when $L = l \cdot \operatorname{lcm}(T_0, T_1)$, if x is in the range $\{0, 1, \ldots, \operatorname{lcm}(T_0, T_1) - 1\}$, the number of solutions to the congruence system above is $N_{t,i}^0$. Thus, when x is in the range $\{0, 1, \ldots, L-1\}$, the solution number $N_{t,i} = l \cdot N_{t,i}^0$, and

$$\lambda_t = \frac{\sum_{i=0}^{t_1-1} N_{t,i}}{N_t} = \frac{\sum_{i=0}^{t_1-1} N_{t,i}}{\sum_{i=0}^{T_1-1} N_{t,i}} = \frac{\sum_{i=0}^{t_1-1} l \cdot N_{t,i}^0}{\sum_{i=0}^{T_1-1} l \cdot N_{t,i}^0} = \frac{\sum_{i=0}^{t_1-1} N_{t,i}^0}{\sum_{i=0}^{T_1-1} N_{t,i}^0}.$$

Thus, without loss of generality, the only situation considered is when x is in the range $\{0, 1, \ldots, \operatorname{lcm}(T_0, T_1) - 1\}$. For any t ($0 \leqslant t \leqslant T_0 - 1$), let $t' = t \pmod{\gcd(T_0, T_1)}$ ($0 \leqslant t' \leqslant \gcd(T_0, T_1)$), and the above congruence equations have solutions when i is in range $\{t', t' + \gcd(T_0, T_1), \ldots, t' + h_{\max}^t \cdot \gcd(T_0, T_1)\ (t' + h_{\max}^t \cdot \gcd(T_0, T_1) < T_1)\}$. The number of such i values is as follows:

$$\sum_{i=0}^{T_1-1} N_{t,i}^0 = h_{\max}^t + 1 = \left\lceil \frac{T_1 - t'}{\gcd(T_0, T_1)} \right\rceil = \frac{T_1}{\gcd(T_0, T_1)}.$$

On the other side, when the boundary condition is set to be $t' + h_{\max}^t \cdot \gcd(T_0, T_1) < t_1$ instead of $t' + h_{\max}^t \cdot \gcd(T_0, T_1) < T_1$, we can calculate the following:

$$\sum_{i=0}^{t_1-1} N_{t,i}^0 = h_{\max}^t + 1 = \left\lceil \frac{t_1 - t'}{\gcd(T_0, T_1)} \right\rceil.$$

Hence we have

$$\lambda_t = \frac{\sum_{i=0}^{t_1-1} N_{t,i}^0}{\sum_{i=0}^{T_1-1} N_{t,i}^0} = \left\lceil \frac{t_1 - t'}{\gcd(T_0, T_1)} \right\rceil \cdot \frac{\gcd(T_0, T_1)}{T_1}.$$

From the above discussion, even if condition $L = l \cdot \operatorname{lcm}(T_0, T_1)$ is false, when L is sufficiently large, we obtain

$$\lambda_t \approx \left\lceil \frac{t_1 - t'}{\gcd(T_0, T_1)} \right\rceil \cdot \frac{\gcd(T_0, T_1)}{T_1},$$
$$t' = t \pmod{\gcd(T_0, T_1)},$$
$$0 \leqslant t' \leqslant \gcd(T_0, T_1).$$

Furthermore, $\lambda_t \approx t_1/T_1 = t_1/(t_1 + n_1)$ in the case when the value of $t_1/\gcd(T_0, T_1)$ is an integer or is relatively large. In particular, when $T_0 = T_1$, $T_0 = T_1 = \gcd(T_0, T_1)$, $t' = t$, and

$$\lambda_t \approx \left\lceil \frac{t_1 - t}{T_0} \right\rceil = \begin{cases} 1, & 0 \leqslant t \leqslant t_1 - 1; \\ 0, & t_1 \leqslant t \leqslant T_0 - 1. \end{cases}$$

In summary, the data complexity of SW-SCA is $\frac{1}{\lambda_{t_0}}$ times that of the original SCA. Where we have

$$\lambda_{t_0} \approx \left\lceil \frac{t_1 - t_0'}{\gcd(T_0, T_1)} \right\rceil \cdot \frac{\gcd(T_0, T_1)}{T_1},$$
$$t_0' = t_0 \pmod{\gcd(T_0, T_1)},$$
$$0 \leqslant t_0' \leqslant \gcd(T_0, T_1).$$

t_0 is the position of the leakage point that is most relevant to the intermediate value in the power curve. In particular, when $t_1/\gcd(T_0, T_1)$ is an integer or is relatively large, $\lambda_{t_0} \approx t_1/T_1 = t_1/(t_1 + n_1)$.

4.2 Time complexity

Because the acquisition and the encryption devices do not work at the same time, the time complexity of SW-SCA is also greater than the original SCA. For simplicity, we assume that the working time of the acquisition device is less than the total encryption time. If the working time of acquisition device is MT_0 and the working time of encryption device is NT_0. The corresponding plaintext of the first power curve has $N - M + 1$ possible cases. The current algorithm exhausts all of the possible $N - M + 1$ possible cases, so the time complexity is multiplied by $N - M + 1$. In addition, the encryption cycle is T_0, that is, a curve has T_0 points. Because we have no information to determine the target location, we must perform this calculation for all T_0 points. The original SCA can always select a certain range of points to finish the analysis, so the time complexity of SW-CPA is multiplied by T_0/W, where W is the number of points of each curve that original SCA used. Therefore, the time complexity of SW-SCA is $((N - M + 1)T_0)/W$ times that of the original SCA.

5 Experiments

In this section, we first perform a simulation experiment to verify the effectiveness of SW-CPA and SSW-CPA, and the correctness of the theoretical complexity evaluation, and then we finish our practical analyses of SW-CPA and SSW-CPA with respect to the scenario presented in this paper.

5.1 Experiments on the simulated traces

5.1.1 Measurement setup

Our analysis method is performed with respect to AES-128. Firstly, in the original scenario SCA attack scenario, we use an Agilent DSO-X 3034A oscilloscope to sample the power consumed by the SASEBO-GII board at a sampling rate of 5×10^7 s^{-1}, and a PA303 amplifier to amplify the signal power, and we obtained a power matrix of $20\,000 \times 3253$. To simulate the partial and discontinuous power consumptions in the no-trigger attack scenario, for the obtained initial power matrix, we set the parameters $t_1 = 1000$, $n_1 = 2000$. That is, we set the processing period to keep 1000 points and throwing away 2000 points, and we deleted 2000 curves each at the beginning and end of power matrix. The data thrown away is denoted by NaN.

5.1.2 Results

We performed the first SW-CPA and SSW-CPA on the SASEBO-GII samples using the AES-128 algorithm. Figures 4 and 5 show the differences obtained when guessing different subkeys for different SPEs, with the correct subkey plotted in black and the rest are plotted in gray. The correct subkey 83 corresponds to the highest spike, which also tells us that the next M plaintext starting from position 2000 is the sequence corresponding to the power consumption.

Figures 6 and 7 show the experimental results when taking the number of traces as a variable. The more traces are used, the more distinguishable is the correct subkey. The correct subkey 83 converges to about 0.072, whereas the others converge to about 0.01.

Based on the traces of 4000 (M_1) or more, it

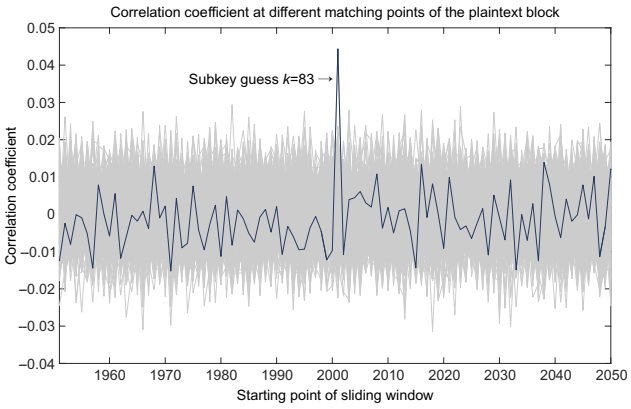

Fig. 4 SW-CPA result as SPE changes in simulation experiment.

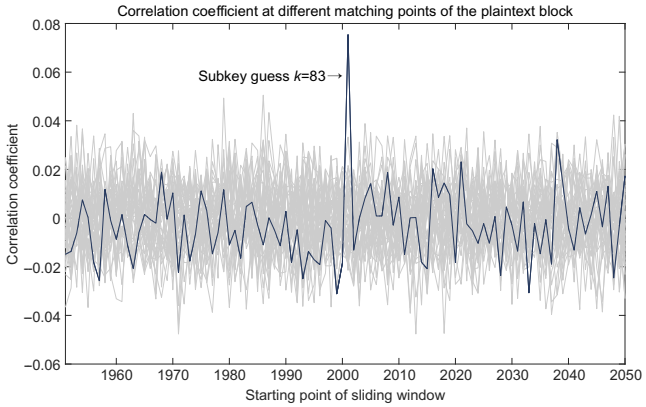

Fig. 5 SSW-CPA result as SPE changes in simulation experiment.

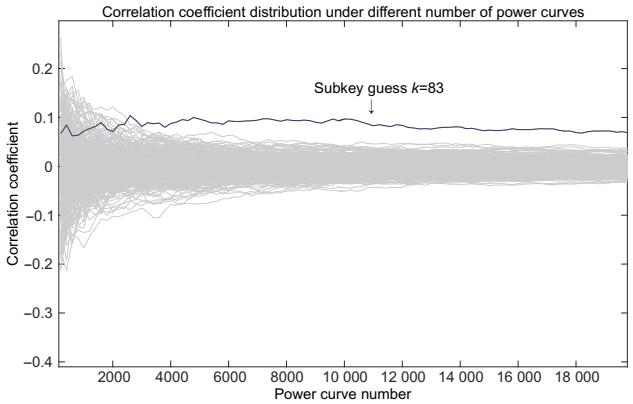

Fig. 6 SW-CPA result as the number of traces changes in simulation experiment.

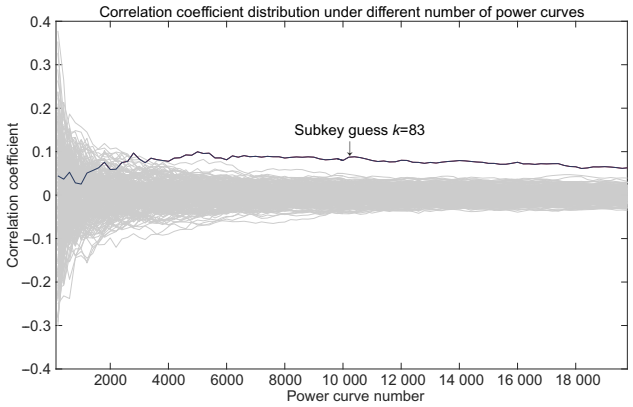

Fig. 7 SSW-CPA result as the number of traces changes in simulation experiment.

is possible to obtain the stable maximum correlation coefficient, whereas in the original with-trigger scenario, the required number of curves for a successful attack is about 1600 (M_0), as shown in Fig. 8. According to the proofs presented in Section 4.1, we calculated that $\lambda_t \approx 0.332$, $M_1 = \frac{1}{\lambda} M_0 = 4820 \approx$

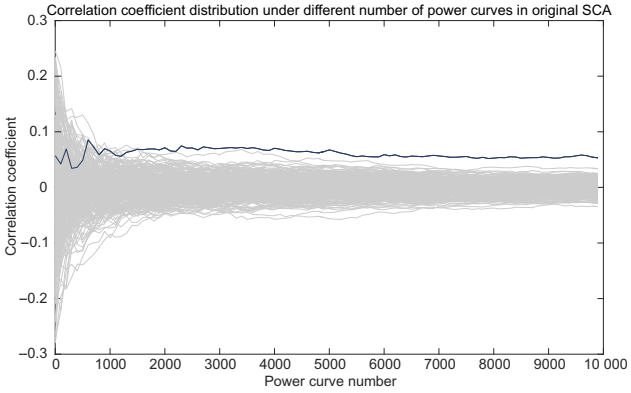

Fig. 8 CPA result as the number of traces changes.

4000, so we can accept that the number of curves in the simulation experiment and the previous theoretical estimate are basically the same.

5.2 Experiments on the physical traces

We conducted practical experiments on SASEBO-W board, which is connected only to the oscilloscopes channel-1 by its port-J2 without any computer control or triggering system. The power consumption sampled by the oscilloscope is stored on a PC, and the entire scenario was like that shown in Fig. 1. The parameters in this experiment were $T_1 = 6\,000\,052$, $T_0 = 7326$, and $t_1 = 20\,000$, and we sampled a total of $40 \times 20\,000$ power consumption data items. We performed both SW-CPA and SSW-CPA, and the experimental results are shown in Figs. 9 and 10, where the correct subkey 43 is plotted in black and the rest are in gray. From the figures we can see that SSW-CPA succeeds in recovering the correct subkey 43, whereas SW-CPA failed.

Then, we applied SSW-CPA to the case with a different number of curves. The result, as shown in

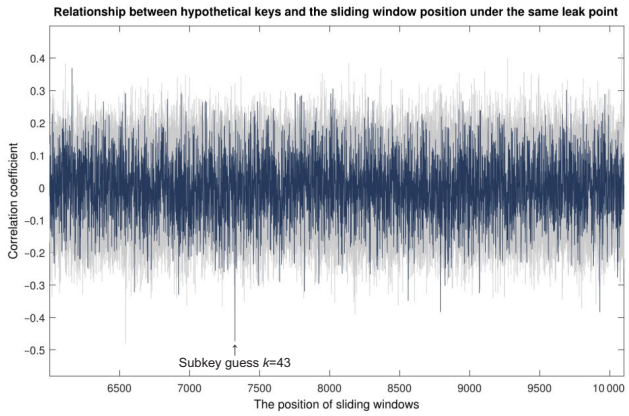

Fig. 9 SW-CPA result as SPE changes in practical experiment.

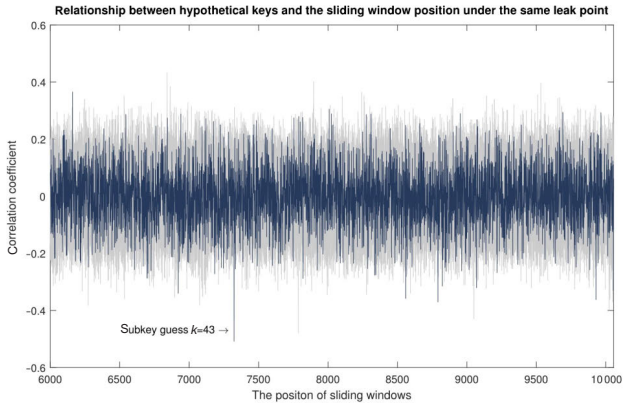

Fig. 10 SSW-CPA result as SPE changes in practical experiment.

Fig. 11, indicate that as the number of traces increases to 10 000 or more, the distinction between the correct and incorrect subkeys becomes more pronounced.

6 Conclusion

In this paper, we applied existing SCAs from experimental conditions in a real attack scenario that had two features. First, the acquisition device worked independently from the encryption target, and second, there was no modification or trigger signal in the encryption source code and the encryption pattern was not obvious in the side channel samples. For SCAs that cannot handle this scenario, we proposed a method called SW-SCA that can.

We also improved the efficiency of SW-SCA using the soft K-means preprocessing method, and proved the effectiveness of incorporating this method. Moreover, we evaluated the time and data complexities of SW-SCA, which we determined by the parameters in the sampling conditions and encryption. Quantitative

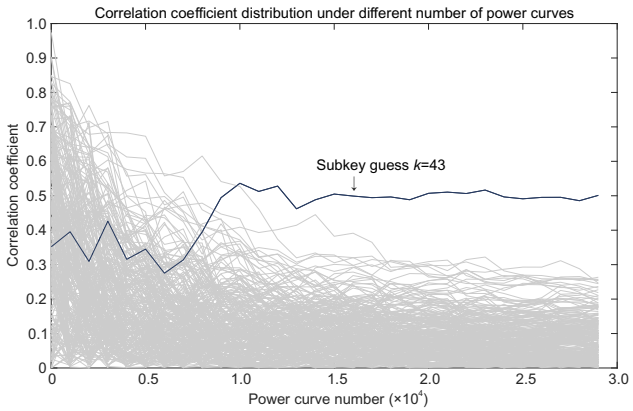

Fig. 11 SSW-CPA result as the number of traces changes in practical experiment.

estimation can help the designer or evaluator determine the real threat from an SCA attacker and improve security in real applications. The experimental results verified the effectiveness of SW-SCA and the correctness of the complexity evaluations.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61472292), the Technological Innovation of Hubei Province (No. 2018AAA046), and the Key Technology Research of New-Generation High-Speed and High-Level Security Chip for Smart Grid (No. 526816160015).

References

[1] P. Kocher, J. Jaffe, and B. Jun, Differential power analysis, *Lecture Notes in Computer Science*, vol. 1666, pp. 388–397, 1999.

[2] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, The EM side-channel(s), *Lecture Notes in Computer Science*, vol. 2523, pp. 29–45, 2002.

[3] P. C. Kocher, Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems, in *International Cryptology Conference on Advances in Cryptology*, 1996, pp. 104–113.

[4] E. Brier, C. Clavier, and F. Olivier, Correlation power analysis with a leakage model, in *Proc. 6^{th} Int. Workshop on Cryptographic Hardware and Embedded Systems*, Cambridge, MA, USA, 2004, pp. 16–29.

[5] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, Mutual information analysis: A generic side-channel distinguisher, in *Proc. 10^{th} Int. Workshop on Cryptographic Hardware and Embedded Systems*, Washington, DC, USA, 2008, p. 1137.

[6] F. X. Standaert, B. Gierlichs, and I. Verbauwhede, Partition vs. comparison side-channel distinguishers: An empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOS devices, in *International Conference Information Security and Cryptology (ICISC 2008)*, P. J. Lee and J. H Cheon, eds. Berlin, Germany: Springer-Verlag, 2009, pp. 253–267.

[7] L. Batina, B. Gierlichs, and K. LemkeRust, Differential cluster analysis, in *Cryptographic Hardware and Embedded Systems (CHES 2009)*, C. Clavier and K. Gaj, eds. Berlin, Germany: Springer, vol. 5747, pp. 112–127, 2009.

[8] S. Chari, J. R. Rao, and P. Rohatgi, Template attacks, in *Proc. 4^{th} Int. Workshop Redwood Shores*, Berlin, Heidelberg, 2002, pp. 13–28.

[9] W. Schindler, K. Lemke, and C. Paar, A stochastic model for differential side channel cryptanalysis, in *Proc. 7^{th} Int. Workshop*, Edinburgh, UK, 2005, pp. 30–46.

[10] C. C. Consortium, Commoncriteria (aka CC) for information technology security evaluation (ISO/IEC15408), https://en.wikipedia.org/wiki/Common_Criteria, 2005.

[11] R. J. Easter, J. P. Quemard, and J. Kondo, Text for ISO/IEC 1st CD 17825-information technology-security

techniques-non-invasive attack mitigation test metrics for cryptographic modules, https://www.iso.org/standard/60612.html, 2014.

[12] AIST, Side-channel attack standard evaluation board (SASEBO), http:// satoh.cs.uec.ac.jp/SASEBO/en/board/sasebo-g2.html, 2009.

[13] V. Lomné, E. Prouff, M. Rivain, T. Roche, and A. Thillard, How to estimate the success rate of higher-order sidechannel attacks, in *Proc. 16th Int. Workshop on Cryptographic Hardware and Embedded Systems*, Busan, South Korea, 2014, pp. 35–54.

[14] C. Whitnall and E. Oswald, Robust profiling for DPA-style attacks, in *Proc. 17th Int. Workshop on Cryptographic Hardware and Embedded Systems*, Saint-Malo, France, 2015, pp. 3–21.

[15] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl, Clustering algorithms for non-profiled single-execution attacks on exponentiations, in *Proc. 12th Int. Conf. on Smart Card Research and Advanced Applications*, Berlin, Germany, 2013, pp. 79–93.

[16] D. J. MacKay, *Information Theory Inference and Learning Algorithms*, Cambridge, UK: Cambridge University Press, 2003.

[17] E. Prouff, M. Rivain, and R. Bevan, Statistical analysis of second order differential power analysis, *IEEE Trans. Comput.*, vol. 58, no. 6, pp. 799–811, 2009.

[18] S. Bhasin, J. L. Danger, S. Guilley, and Z. Najm, Sidechannel leakage and trace compression using normalized inter-class variance, in *Proc. 3rd Workshop on Hardware and Architectural Support for Security and Privacy*, New York, NY, USA, 2014, p. 7.

[19] S. Mangard, Hardware countermeasures against DPA—A statistical analysis of their effectiveness, in *Cryptographers' Track at the RSA Conference*, T. Okamoto, ed. Berlin, Germany: Springer, 2004, pp. 222–235.

Ming Tang received the PhD degree from Wuhan University, China, in 2007. She is currently a professor at the School of Cyber Science and Engineering, Wuhan University, China. Her research interests include cryptography and side channal analysis, innovative analysis method in side channal analysis, secure design of cryptographic chips, and other systematic research on side channel analysis.

Maixing Luo received the bachelor degree from Wuhan Textile University, Wuhan, China, in 2016. He is currently pursuing the master degree in computer application technology at Wuhan University. His current research interests include side channel analysis and cryptanalysis.

Junfen Zhou is currently a graduate student in the College of Computer Science at Zhejiang University. He received the BS degree from Wuhan University. His currently research interests include big data driven security and adversarial learning.

Zhen Yang received the bacheor degree from Wuhan University, Wuhan, China, in 2018. She is currently pursuing the master degree in information security at Wuhan University. Her current research interests include side channel analysis and cryptanalysis.

Zhipeng Guo received the master degree from Wuhan University, Wuhan, China, in 2016. He is currently pursuing the PhD degree in information security at Wuhan University. His current research interests include side channel analysis and cryptanalysis.

Fei Yan received the PhD degree from Wuhan University, Wuhan, China, in 2007. He is currently an associate professor in system security at Wuhan University. His main research interests include side-channel analysis, mobile security, and trusted computing.

Liang Liu received the master degree from Beihang University, Beijing, China, in 2006. He is currently a deputy manager in the Complementary Metal-Oxide-Semiconductor (CMOS) Design Department at Beijing Smart-Chip Microelectronics Technology Company Limited. His current research interests include Application-Specific Integrated Circuit (ASIC) security chip design, side-channel attacks, and countermeasures for cryptographic algorithms.