Time is not enough: Timing Leakage Analysis on Cryptographic Chips via Plaintext-Ciphertext Correlation in Non-timing Channel

Congming Wei[1], Guangze Hong[1], An Wang[1], Jing Wang[1], Shaofei Sun[1], Yaoling Ding[1], Liehuang Zhu[1] and Wenrui Ma[1]

Beijing Institute of Technology, Beijing, China
weicm@bit.edu.cn,honggz@bit.edu.cn,wanganl@bit.edu.cn,wangjing9624@163.com,sfsun@bit.edu.cn,dyl19@bit.edu.cn,liehuangz@bit.edu.cn,mawenrui@zjgsu.edu.cn

Abstract. In side-channel testing, the standard timing analysis works when the vendor can provide a measurement to indicate the execution time of cryptographic algorithms. In this paper, we find that there exists timing leakage in power/electromagnetic channels, which is often ignored in traditional timing analysis. Hence a new method of timing analysis is proposed to deal with the case where execution time is not available. Different execution time leads to different execution intervals, affecting the locations of plaintext and ciphertext transmission. Our method detects timing leakage by studying changes in plaintext-ciphertext correlation when traces are aligned forward and backward. Experiments are then carried out on different cryptographic devices. Furthermore, we propose an improved timing analysis framework which gives appropriate methods for different scenarios.

Keywords: Timing analysis · Side-channel analysis · Timing leakage · Plaintext-ciphertext correlation

1 Introduction

Side-channel analysis (SCA) has drawn increasing attention in the security community, which exploits side-channel information such as execution time [1], power consumption [2], electromagnetic (EM) radiations [3] and cache [4]. Timing analysis is one of the most common side-channel analysis which only needs to measure time information to attack cryptographic systems. The idea of timing analysis comes from the fact that changes of inputs will lead to differences in running time. By taking advantage of the time that devices take to run under different inputs, secret keys are successfully recovered with statistical analysis [1,5–7]. Besides, timing attack can even be carried out remotely, making it more stealthy and dangerous [8–10].

Since it is simply not feasible to apply all possible attacks to verify the security of cryptographic devices, testing laboratories need a standard method that performs fast and reliable side-channel evaluation of a given product. ISO/IEC 17825 (Testing methods for the mitigation of non-invasive attack classes against cryptographic modules) [11] is the main publicly available standard that focuses on vulnerabilities for non-invasive attacks against devices at security level 3 or 4. Therefore, from the perspective of academia and industry, the design of ISO/IEC 17825 test methods has received close attention [12–14].

ISO/IEC 17825 specifies a testing method for timing analysis (as we describe in Section 2.1) to test whether cryptographic modules meet the requirement in the ISO standard ISO/IEC 19790 [15]. However, there still remain some open problems when applying

ISO/IEC 17825 to a cryptographic module in practice.

1. How do we collect timing leakage?

   The testing method is based on a precondition that testing laboratories are able to measure execution time when performing timing analysis. In fact, leakage measurement and processing depend on cryptographic modules. Unfortunately, the leakage measurement part is not mentioned in the standard.

2. Is the testing method reliable?

   Even if the execution time has been successfully measured, we have to consider how accurate the execution time is when it comes to the reliability of timing analysis. If the measurement is not accurate enough, it seems inappropriate to claim that a cryptographic module passes the timing analysis test.

3. Is execution time the only timing leakage?

   Ideally, the general method could be able to detect whether the device under test is vulnerable to timing attacks or not. However, when the execution time is imprecise or simply impossible to measure, it is indeed infeasible. Instead, the introduction of non-traditional measurement methods is a powerful complement to standard timing analysis. Although there are several combined attacks which determine execution time of some particular operations by exploiting power consumption [16–18], timing leakage from other channels often goes ignored in timing analysis.

Considering the above questions, we aim to put forward a new leakage analysis framework for timing analysis to deal with scenarios where the conventional timing analysis method works ineffectively. It is worth noting that the new framework should include timing analysis of both time and other channel information. For cases where the execution time is difficult to measure, we perform timing analysis with the help of timing leakage derived from power consumption and electromagnetic radiation information. The idea originated from trace alignment, a technique commonly used before statistical analysis when we perform power analysis and electromagnetic analysis. We observed that in process of trace alignment once traces are aligned at the beginning of the encryption, the difference in execution time to a great extent leads to the traces being misaligned at the end, even though the misalignment may not be significant. Naturally, it occurs to us whether timing analysis could be performed by detecting misalignment of side-channel traces.

In this paper, we focus on timing leakage in the power/EM channels as a supplement to timing channel. To this end, we put forward a kind of timing analysis testing method based on power/EM trace alignment. This approach is based on the observation that the length of the execution segment in a power/EM trace is determined by the execution time of encryption. In other words, while traces are well aligned at the beginning (or the end), whether they are aligned at the end (or the beginning) depends on whether execution times are consistent or not. In this way, trace alignment can be used to determine whether there is timing leakage in power/EM traces. We do not claim that our method is superior and can replace standard methods, but we do believe that the length difference in traces could help testers to perform timing analysis when leakage in the timing channel is inaccurate or even unavailable.

For different devices under test, we should choose methods that are general enough to detect trace alignment. In particular, when there are no significant differences in execution segments, leakage detection is expected to provide statistical analysis as much as possible and to be largely independent of visual observation. To this end, our method uses plain-ciphertext correlation analysis to determine the position of the execution segment, and then detects the alignment before and after the execution segment. It is obvious that

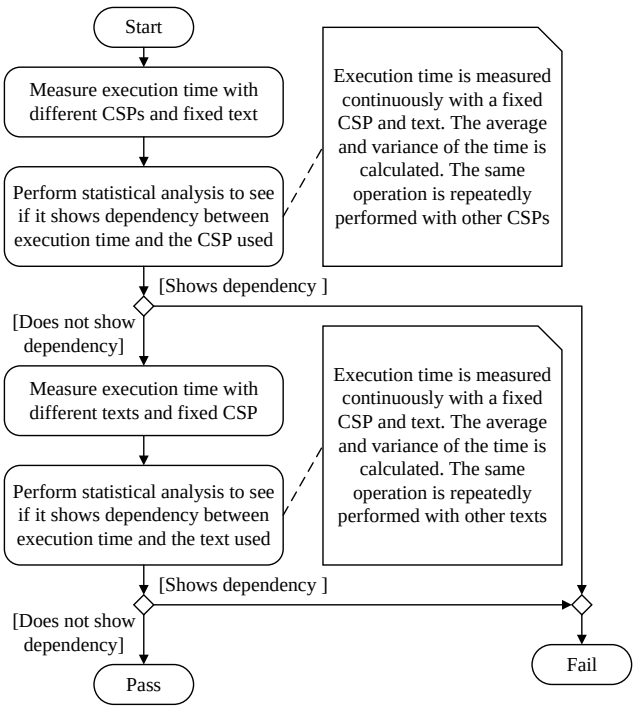


Figure 1: Timing analysis in ISO/IEC 17825

the plain-ciphertext correlation is higher in the trace-aligned case than in the unaligned case. Thereafter, we give a general measure by using statistical analysis such as Fisher transform and p-value.

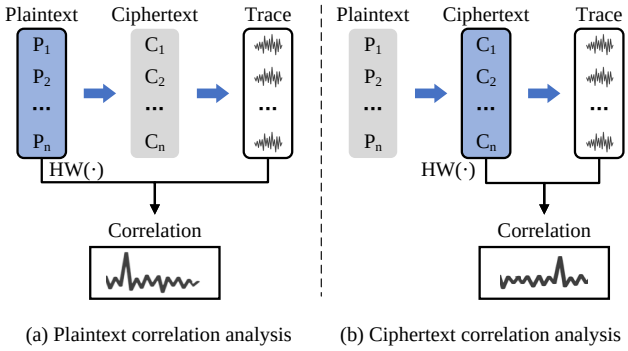
Furthermore, we put forward a timing analysis framework which divides target devices into three scenarios and presents a testing method for each scenario. For the first scenario where the cryptographic device is triggered through a GPIO interface or I/O signal, we first propose an I/O channel-based method that details the time measurement process, which is omitted by the previous timing analysis. For the second scenario where high signal-to-noise ratio (SNR) power/EM traces are available, our method via plaintext-ciphertext correlation analysis is adopted. For other scenarios, we use a method based on average denoising by detecting communication between computers and devices. The most prominent contribution of our framework is that testing laboratories have the flexibility to choose testing methods as needed, which makes timing analysis more complete.

2 Preliminaries

2.1 Timing Analysis in ISO/IEC 17825

ISO/IEC 17825 specifies a side-channel resistance test framework including standard timing analysis, simple power analysis and differential power analysis (DPA). Here we only introduce the timing analysis testing method.

As shown in Figure 1, the timing analysis testing method in ISO/IEC 17825 consists of two stages. In the first stage, execution time is measured with different secret keys and a fixed plaintext. Using n secret keys we could obtain n respective execution times, then perform statistical analysis to show if statistical characteristics of n execution times



(a) Plaintext correlation analysis (b) Ciphertext correlation analysis

Figure 2: Plaintext-ciphertext correlation analysis

are consistent or not. The testing result is Fail if execution times vary considerably according to different secret keys. Otherwise, the test proceeds to the next stage. In the second stage, execution time is measured with different plaintexts and a fixed secret key. Again, execution times are treated through statistical analysis. If the results do not show dependency between execution time and plaintexts, the test passes; otherwise, it fails. In the standard, two execution times T_1 and T_2 are compared by computing $|T_1 - T_2|$. The difference between T_1 and T_2 can be tolerant only when $|T_1 - T_2| < \varepsilon$ where ε equals a clock cycle.

2.2 Plaintext-Ciphertext Correlation Analysis

When detecting similarity in side-channel analysis, correlation is one of the most popular distinguishers, widely used in both vertical [19] and horizontal attacks [20]. The Pearson correlation coefficient of two random variables is defined as the covariance of two variables divided by their standard deviations:

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X-\bar{X})(Y-\bar{Y})]}{\sigma_X \sigma_Y}, \tag{1}$$

where $\bar{X}, \bar{Y}, \sigma_X, \sigma_Y$ stand for the average and standard deviation of X and Y.

The plaintext correlation analysis is a vertical side-channel analysis, which exploits each column of sample points in traces. The Pearson correlation coefficient is applied to exhibit the relationship between measured leakage and calculated values derived from a leakage model. Taking the Hamming weight as a model, in the case when N traces are collected, N Hamming weights $\mathrm{H}(M_i)$ are calculated for messages $M_i, 0 \le i < N$. Then for each column of sample points $T_{*,j}$, we have

$$\rho_{\mathrm{H}(M),T_{*,j}} = \frac{\mathrm{Cov}(\mathrm{H}(M), T_{*,j})}{\sigma_{\mathrm{H}(M)} \sigma_{T_{*,j}}}. \tag{2}$$

As shown in Figure 2, performing plaintext correlation analysis on all sample points produces a correlation coefficient curve. The significant peaks exhibit points that have a strong correlation with plaintexts.

Similarly, ciphertext correlation analysis applies the Pearson correlation coefficient to show the relationship between sample points $T_{*,j}$ and ciphertexts C. The correlation coefficient is defined as follows:

$$\rho_{\mathrm{H}(C),T_{*,j}} = \frac{\mathrm{Cov}(\mathrm{H}(C), T_{*,j})}{\sigma_{\mathrm{H}(C)} \sigma_{T_{*,j}}}. \tag{3}$$

The significant peaks in a ciphertext correlation coefficient curve exhibit points that have a strong correlation with ciphertexts.

3 Timing Analysis Based on Plaintext-Ciphertext Correlation Analysis

3.1 Timing leakage in Power/EM Channels

Traditional timing analysis can measure and analyze execution time to show if time is dependent on secret information, which is natural and works in most cases where it is easy to obtain execution time of encryption modules. However, in practice, execution time can be unavailable or at least inefficient. For example, noise on the communication interface may be very serious, which in turn affects timing accuracy. Besides, an IO command can lead to multiple operations, not just encryption, resulting in timing results with a number of noises. Even in the absence of these noises, side-channel countermeasures like random delay could affect timing analysis. Once such a situation is encountered, our best approach is to look for new analysis methods rather than directly declaring that the device passed the test.

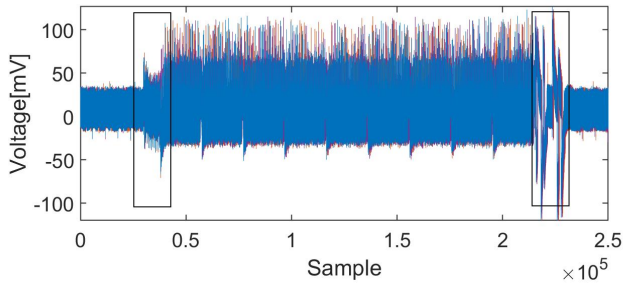
It seems that timing leakage does not only exist in traditional measurement, but there is much to discover. A straightforward thought coming to our mind is whether we could exploit timing leakage in other side channels, such as power and EM, in which cases lots of work focused on power analysis and electromagnetic analysis have been put forward while timing analysis is hardly considered. Thus, we take time information of traces into account. Representing an interval from the beginning to the end of an algorithm as an execution segment, it is obvious that the length of an execution segment depends on the execution time. If plaintexts and secret keys are fixed, then the same operation should be performed in a fixed amount of time in each encryption. If the cryptographic module uses different plaintexts and secret keys, the same operation in the encryption process may take inconsistent execution time due to different operands, resulting in different execution segment lengths. The phenomenon that segment length can reflect time changes prompts us to explore new methods for timing analysis.

The observed results further demonstrate the availability of time information in the power traces. Figure 3a illustrates a series of power traces collected from a smart card which performs the AES algorithm with different plaintexts. When we amplify the beginning and the end of execution segments, it appears that traces are aligned at the beginning of execution but clearly not aligned at the end, as shown in Figure 3b and Figure 3c. Intuitively, different plaintexts result in different execution segment lengths, that is, different execution time. It is proved that there exists timing leakage in the power channel.

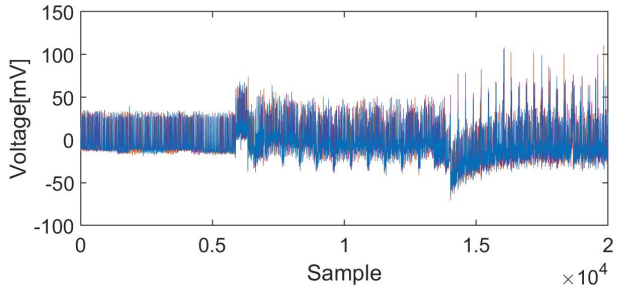
The above experiments show that power traces do help timing analysis testing since timing leakage can be detected by analyzing execution segment lengths. The same applies to EM channels. Execution segments with different plaintexts have the same length only when the cryptographic module produces no timing leakage. However, this result is obtained by the naked eye, which only applies when the execution segment gap is large, and in the next subsection we will provide a more reliable method to deal with complex cases.

3.2 Detecting Timing Leakage with Correlation Analysis

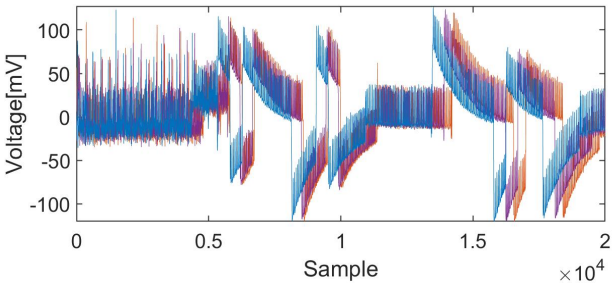
Section 3.1 pointed out that timing leakage does not only exists in the timing channel but can also be detected through power/EM traces. The execution segment length of a trace could indicate the execution time. However, it is inaccurate to compare execution segment length by visual observation. Besides, visual observation is too subjective to distinguish



(a) The whole traces



(b) Amplification at the beginning



(c) Amplification at the end

Figure 3: Three power traces collected from a smart card with different plaintexts. The boxes indicate the beginning and end of the algorithm execution.

execution segments with similar length. In this section, we aim to find a metric to detect timing leakage from power/EM traces.

The cryptographic operation generally consists of three parts, i.e., plaintext transmission, encryption and ciphertext transmission. When plaintext correlation analysis is performed, sample points relative to plaintext transmission and plaintext processing operations have high correlation with the Hamming weight of plaintext, as shown in Section 2.2. Similarly, samples relative to ciphertext transmission and ciphertext processing operations have high correlation with the Hamming weight of ciphertext. Assume that traces are well aligned at the beginning of the cryptographic operation. If execution segments with different plaintexts have the same length, then traces keep aligned in the ciphertext transmission process, resulting in a high ciphertext correlation. Otherwise, traces are not aligned in the ciphertext transmission process, which means that the ciphertext correlation will be somewhat reduced.

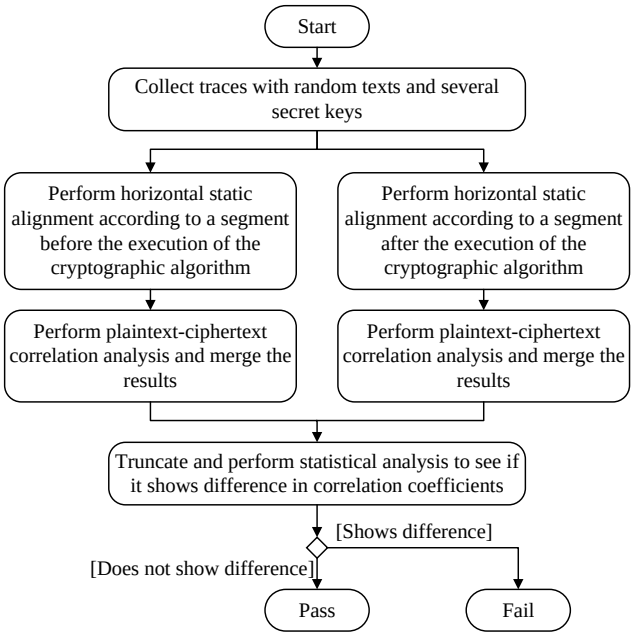


Figure 4: Timing analysis based on plaintext-ciphertext correlation analysis

3.3 Testing Process of Timing Analysis via Plaintext-Ciphertext Correlation Analysis

Our trace-based timing analysis is to evaluate the difference in execution time by calculating changes of plaintext-ciphertext correlation. Large changes indicate that there exists timing leakage.

We do timing leakage analysis using traces with different plaintexts and keys. Hence the timing difference consists of three parts, namely, plaintext timing leakage, key timing leakage and noises. The first two parts are leakages that need to be detected. Since the execution times under fixed plaintext and fixed key is relatively stable, in our paper, we assumed that plaintext and key timing leakages account for the majority of timing difference and noises is negligible.

The testing process consists of four steps, as illustrated in Figure 4.

1) Power/EM trace measurement

Our method is based on high SNR power/EM traces. Here we assume that secret keys of cryptographic algorithms could be changed. During the trace collecting process, traces are measured continuously with different plaintexts and a fixed secret key. This operation is repeated several times with different secret keys so as to obtain a trace set. Note that traces should cover the whole execution, that is, the measurement is started before plaintext transmission and stopped after ciphertext transmission.

2) Static alignment

Before performing correlation analysis, traces should be well aligned. High plaintext correlation requires these traces to be aligned forward while high ciphertext correlation requires them to be aligned backward. Misalignment will affect the position and shape of peaks in correlation coefficient curves. Besides, since our purpose is to do timing analysis

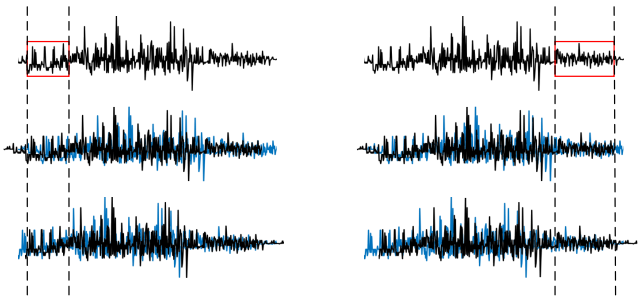


Figure 5: Trace alignment. The traces in blue are the initial traces, and the traces in black are the ones after alignment.

testing, the static alignment is adopted, which makes traces moved without stretching and compressing.

Static alignment selects one of traces as the reference trace.

During alignment, only a segment with significant peaks is considered as the reference. By analyzing traces, for example, finding 10 rounds of AES, the execution interval can be found. For forward alignment, the reference segments should be chosen before the execution interval, while a segment after the execution interval is chosen for backward alignment, which both do not include plaintext and ciphertext transmission. In this way, the difference mainly comes from execution segments, which significantly reduces the influence of reference points. Of course, when there are random delays after reference points, the selection of reference points will affect the offset, so our method is not applicable to random delays.

For each trace, the correlation between the reference segment and all segments within the offset range is calculated. If the maximum correlation reaches the threshold, the relative position with the maximum correlation is the offset value. Otherwise, the trace is discarded. In order to get better results, a segment with significant peaks is usually selected as the reference.

Our method needs two separate trace alignments. As shown in Figure 5, the forward alignment selects a segment before the execution of cryptographic algorithms as the reference while the backward alignment selects a segment after the execution. The statistical analysis below applies only to those aligned traces.

3) Plaintext-ciphertext correlation analysis

Plaintext-ciphertext correlation analysis consists of two parts, namely plaintext correlation and ciphertext correlation. Our method performs analysis on forward-aligned and backward-aligned traces and then merges the results.

As shown in Section 2.2, given the forward-aligned traces and their corresponding plaintexts and ciphertexts, plaintext-ciphertext correlation analysis produces correlation coefficient curves by calculating the Pearson correlation coefficients on each column of sample points. In general, plaintexts can be split, such as by bytes, with each part computed separately. Thus, if multiple parts are considered, multiple correlation coefficient curves are generated. The peaks on curves represent the sample points with high correlation. In plaintext correlation analysis, we mainly focus on the peaks before the execution of algorithms, corresponding to the plaintext transmission. In ciphertext correlation analysis, ciphertexts can also be divided into some parts so that each of them produces a cipher-

text correlation coefficient curve. Taking multiple parts will result in multiple curves. The peaks of ciphertext correlation coefficient curves usually appear in the segments of ciphertext transmission.

To facilitate the analysis, multiple correlation analysis curves need to be merged to form a single curve. For each column of sample points, select the sample point with the largest absolute value among all curves, and take its absolute value as the value of the merged curve at that point. The new generated curve will contain all the peaks associated with plaintext and ciphertexts.

Similarly, we perform plaintext-ciphertext correlation analysis on the backward-aligned traces and obtain correlation coefficient curves. These curves are then merged to create a new curve.

4) Truncation and comparison

In order to filter noises, merged curves should be truncated. A threshold is set such that curves only retains the values of sample points greater than the threshold while the values of sample points smaller than the threshold for both curves are set to zero. In this way, only the points with high plaintext or ciphertext correlation are retained.

By calculating the difference between the truncated curves of forward-aligned and backward-aligned traces, we can test whether there exists timing leakage. In order to compare two truncated curves, we introduce the Fisher transform formula which has been used in [21].

The Fisher transform is used in statistics to test hypotheses about correlation coefficients. It can transform variables that follow any probability distribution into Gaussian normal distribution.

Definition 1. *Given N pairs of variables $(X_i, Y_i), i = 0, 1, \cdots, N-1$, the correlation coefficient ρ is*

$$\rho = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}}.$$

Then through the Fisher transform defined as

$$z = \frac{1}{2}\ln\left(\frac{1+\rho}{1-\rho}\right) = \operatorname{arctanh}(\rho),$$

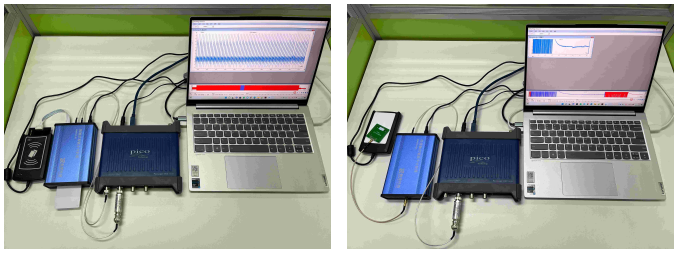
the correlation coefficient is approximately transformed into a Gaussian distribution with the standard deviation $\sigma = \frac{1}{\sqrt{N-3}}$.

Applying the Fisher transform to the two truncated curves, we obtain new curves with each sample points approximately subordinate to a Gaussian distribution. This allows us to compute the following p-value for a null hypothesis assuming the values of sample points z_i^1 and z_i^2 in two curves are the same:

$$p_i = 2 \times \left(1 - CDF\left(\frac{|z_i^1 - z_i^2|}{\sigma}\right)\right). \tag{4}$$

We set the threshold as 0.05, which corresponds to a 95% confidence for the decision made.

If $p_i < 0.05$, reject the null hypothesis, that is, the correlation coefficients of forward-aligned and backward-aligned traces at this point are significantly different. Otherwise, the correlation coefficients are similar. Hence if $p_i > 0.05$ for all sample points (z_i^1, z_i^2), then we claim that the timing analysis passes. Otherwise, it fails.



(a) Contact (b) Contactless

Figure 6: Trace acquisition platforms for contact smart cards and contactless smart cards

3.4 Comparison With TVLA

ISO/IEC 17825 specifies the Test Vector Leakage Assessment (TVLA) as the sole measure to assess whether a cryptographic device is vulnerable to differential side-channel attacks. TVLA divides traces into two groups, one for fixed and one for random inputs, and does statistical analysis by means of the Welchs t-test.

Since each point in time is evaluated independently, TVLA has shortcomings when it comes to horizontal patterns while our method is designed to exploit horizontal leakage. Thus, although both make use of traces, the TVLA test in ISO/IEC 17825 cannot replace our method. On the other hand, in the TVLA rule, cryptographic operations are assumed to always occur at the same moment in each measurement.

In this way, for a single point with obvious leakage, the TVLA value may even become smaller due to the error caused by the misalignment.

When there exists timing leakage in traces, that is, the cryptographic operations take inconsistent execution time, the misalignment caused by different execution segment length makes it easier for cryptographic devices to pass the TVLA test. As a result, TVLA is not only difficult to detect timing leakage, but also reduces its effect due to timing leakage. Section 4.7 will give an example of a cryptographic device that passes the TVLA test while fails the timing analysis test based on plaintext-ciphertext correlation analysis.

4 Evaluation Experiments

4.1 Measurement Setup

As shown in Figure 6a, a trace acquisition platform equipped with PicoScope 3403D oscilloscope, Mini-Circuits 1.9MHz low-pass filter and contact smart card reader is established. Next, we built a trace acquisition platform for a contactless IC card, as shown in Figure 6b.

We also built trace acquisition platforms for Microcontroller Units (MCUs). The 8-bit STC89C52RC Microcontroller Unit has 8k flash, 512-byte RAM and 12MHz crystal, as shown in Figure 7a.

The STM32F429BIT6 is based on the ARM Cortex-M4 32-bit core operating at a frequency of up to 180MHz, as shown in Figure 7b.

In order to validate our method, we performed experiments on different platforms with different algorithms. We use several smart cards in our experiment. According to the information provided by the manufacturer, smart card MMCrypt-21-002 labeled A and smart card MMCrypt-21-006 labeled B are contact smart cards equipped with AES and SM4 respectively, with a clock frequency of 24MHz. Smart card MMCrypt-21-007 labeled C is a contactless smart card configured with the DES algorithm and the clock

(a) 8-bit (b) 32-bit

Figure 7: Trace acquisition platforms for MCUs

Table 1: Details of the experimental cases

	Case 1	Case 2	Case 3	Case 4	Case 5
Platform	Contact card	Contact card	Contactless card	MCU	MCU
Algorithm	AES	SM4	DES	AES	RSA
Protection	Unprotected	Unprotected	Unprotected	Masked	Unprotected
Frequency	24MHz	24MHz	24MHz	12MHz	168MHz
Sampling Rate	5MHz	12.5MHz	125MHz	2.5MHz	12.5MHz
#Traces	9000	9000	9000	18000	1000
#Points	500000	80000	70000	1500000	5800000

frequency is 24MHz. Table 1 lists the details of experimental cases. For all case studies we followed the process in Sec 3.3.

4.2 Test on Contact Smart Cards with AES

We set three secret keys as

$$k_1 = \texttt{0x112233445566778899AABBCCDDEEFF00},$$
$$k_2 = \texttt{0x00FFEEDDCCBBAA9988776655443322 11},$$
$$k_3 = \texttt{0x887766554433221100FFEEDDCCBBAA99},$$

and then collect 3000 traces for each key with random plaintexts, using the trace acquisition platform in Figure 6a. Totally, there are 9000 traces. A trace after low-pass filtering is shown in Figure 8. Each trace has 500000 sample points, and the encryption is believed to be performed in the $25 \sim 70$ms interval.

Select a segment before the encryption as the reference and perform static alignment to traces. Then the plaintext and ciphertext are divided into 16 groups in bytes respectively. Plaintext-ciphertext correlation is analyzed between the aligned traces and the Hamming weights of plaintexts and ciphertexts. Figure 9a and 9b illustrates 32 plaintext-ciphertext correlation coefficient curves and their merged curve. As shown in Figure 9b, the segments before and after encryption have obvious correlation with plaintext and ciphertext, hence we speculate that the transmission of plaintext and ciphertext is carried out in these segments. We also found that the ciphertext correlation is higher than the plaintext correlation because, as shown in Figure 8, the segment corresponding to the ciphertext transmission is more pronounced.

Again, select a segment after the encryption as the reference and perform static alignment. Figure 9e illustrates merged correlation coefficient curves for backward-aligned traces. We found that the ciphertext correlation is significantly improved compared to the correlation coefficient curve in Figure 9b while the plaintext correlation is reduced.

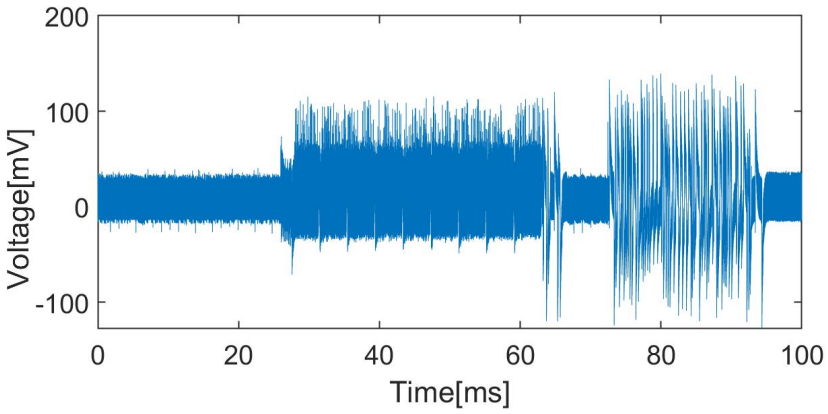


Figure 8: Low-pass filtering trace of smart card A with 500000 sample points

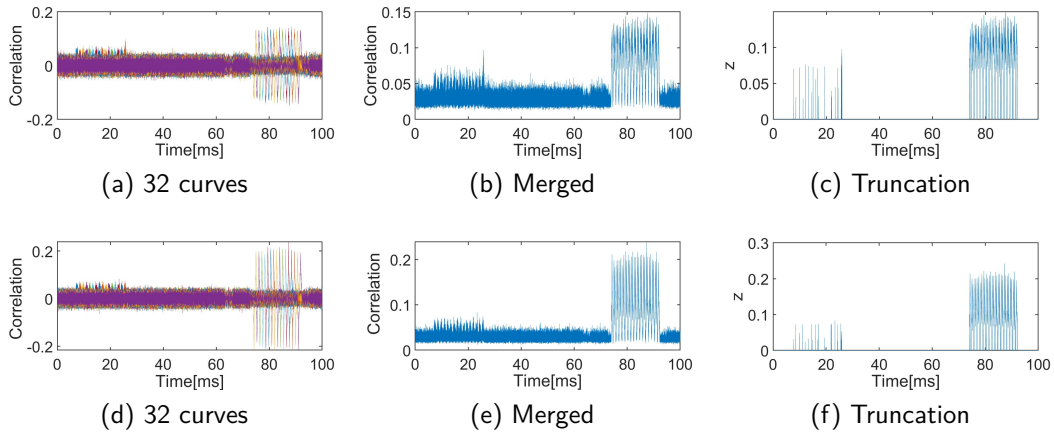


Figure 9: Correlation after amplification of smart card A. (a), (b) and (c) are correlation coefficient curves, their merged curve and truncation curve for forward-aligned traces. (d), (e) and (f) are curves for backward-aligned traces.

The next step is applying Fisher transform to curves and then truncating them. We are able to select an appropriate truncating threshold by observing the curve features. Here the threshold is set as 0.08. Figure 9c and 9f gives Fisher transform and truncation results for merged curves in Figure 9b and 9e. It can be found that the truncated curves are clearer and features look more obvious.

In order to test whether the execution time is consistent, we first calculate the difference of two truncated curves, as shown in Figure 10, and then calculate the p-value for each point according to (4). We have $p_{min} = 2 \times (1 - CDF(\frac{0.12}{\sigma})) \ll 0.05$ with $\sigma = \frac{1}{\sqrt{N-3}} \approx 0.01$. It can be determined that the algorithm implemented by smart card A has obvious timing leakage, and the test fails.

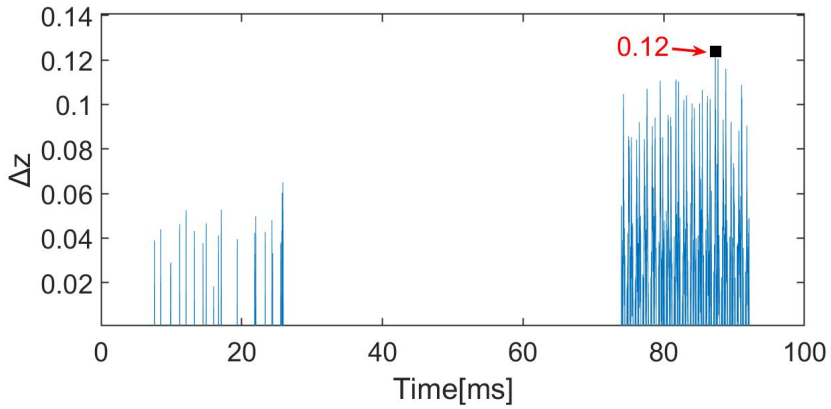


Figure 10: Difference between forward-aligned and backward-aligned correlation coefficient curves of smart card A

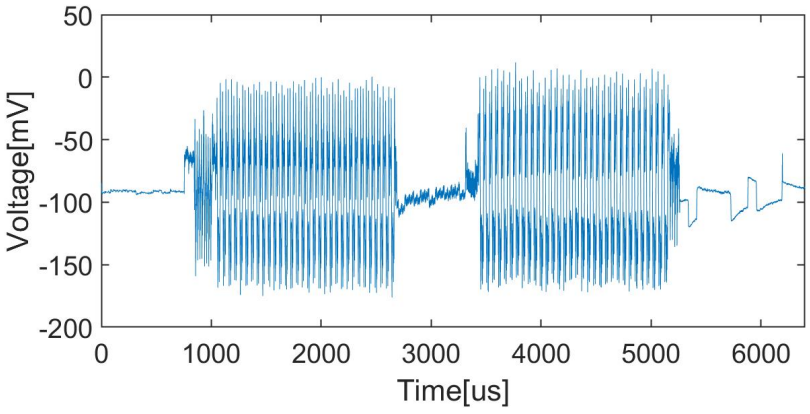


Figure 11: Low-pass filtering trace of smart card B with 80000 sample points

4.3 Test on Contact Smart Cards with SM4

We set three secret keys as

$$k_1 = \texttt{0x11223344556677889AABBCCDDEEFF00},$$
$$k_2 = \texttt{0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA},$$
$$k_3 = \texttt{0x0123456789ABCDEFFEDCBA9876543210},$$

then using the trace acquisition platform in Figure 6a to collect 3000 traces for each key with random plaintexts. A trace after low-pass filtering is shown in Figure 11. Each trace has 80000 sample points, and the algorithm is believed to be performed in the $0.7 \sim 5.4$ms interval.

Select a segment before the encryption as the reference and perform static alignment to traces. The plaintext and ciphertext are divided into 16 groups in bytes respectively. Then plaintext-ciphertext correlation is analyzed. Again, select a segment after the encryption as the reference and perform static alignment and plaintext-ciphertext correlation analysis. Figure 12 illustrates 32 plaintext-ciphertext correlation coefficient curves and their merged curves. By comparing Figure 12b and 12e, it is obvious that plaintext-ciphertext correlation is different under different alignment modes.

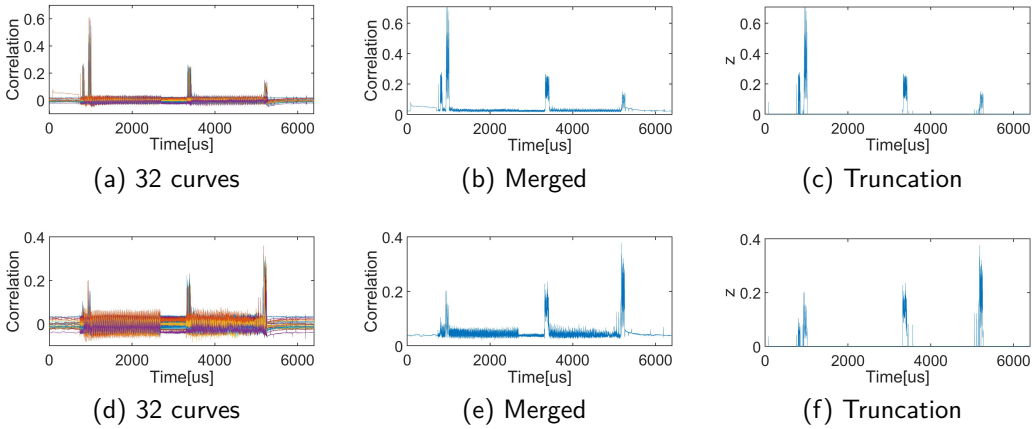


Figure 12: Correlation after amplification of smart card B. (a), (b) and (c) are correlation coefficient curves, their merged curve and truncation curve for forward-aligned traces. (d), (e) and (f) are curves for backward-aligned traces.

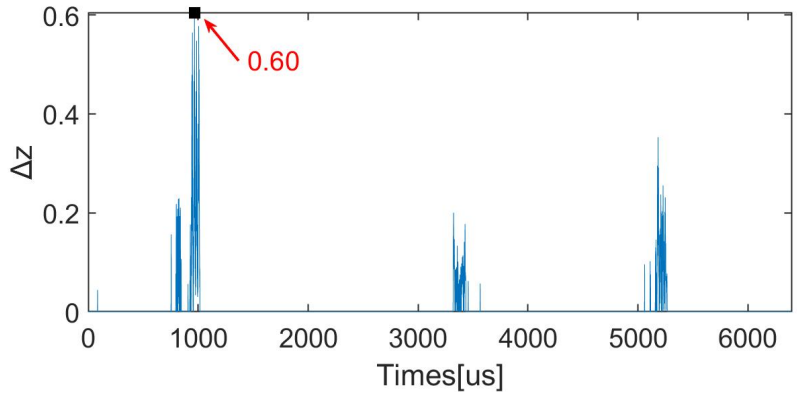


Figure 13: Difference between forward-aligned and backward-aligned correlation coefficient curves of smart card B

Then we apply Fisher transform to curves and truncate them. Here the threshold is set as 0.08. Figure 12c and 12f give Fisher transform and truncation results for merged curves. We calculate the difference of two truncated curves, as shown in Figure 13, and calculate the p-value for each point according to (4). We have $p_{min} = 2 \times (1 - CDF(\frac{0.60}{\sigma})) \ll 0.05$ with $\sigma = \frac{1}{\sqrt{N-3}} \approx 0.01$. It can be determined that the algorithm implemented by smart card B has obvious timing leakage, and the test fails.

4.4 Test on Contactless Smart Cards with DES

In this subsection, a smart card labeled C is used to conduct the black-box timing analysis experiment. Using the trace acquisition platform in Figure 6b, we collect 3000 traces with

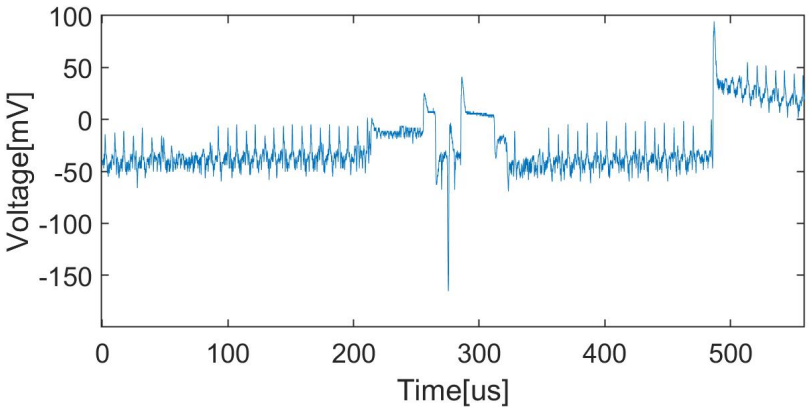


Figure 14: Trace of smart card C with 70000 sample points

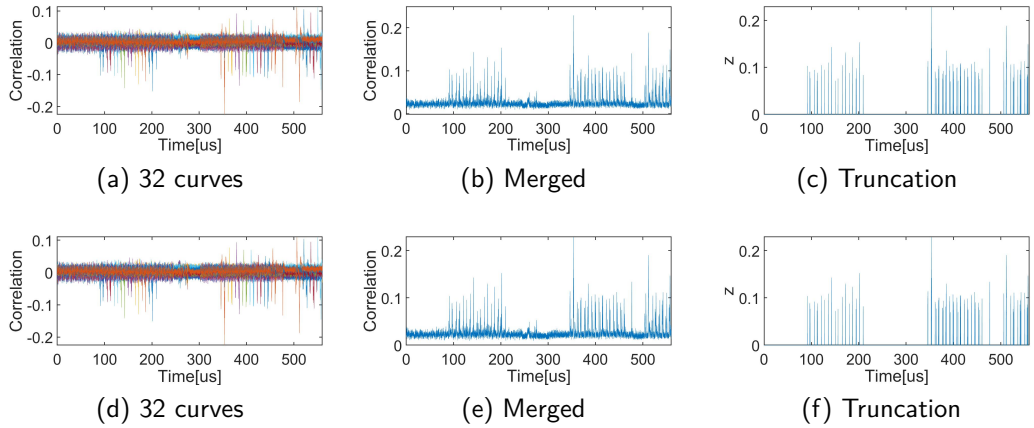


Figure 15: Correlation after amplification of smart card C. (a), (b) and (c) are correlation coefficient curves, their merged curve and truncation curve for forward-aligned traces. (d), (e) and (f) are curves for backward-aligned traces.

random plaintexts for each secret key

$$k_1 = \texttt{0x1122334455667788},$$
$$k_2 = \texttt{0x99AABBCCDDEEFF00},$$
$$k_3 = \texttt{0x8877665544332211},$$

and each trace has 70000 sample points. As illustrated in Figure 14, the trace has a significant downward spike in the interval from 200us to 330us, which is considered to be caused by performing encryption.

Select a segment before the spike as the reference and perform static alignment. The plaintext and ciphertext are divided into 8 groups in bytes respectively. Then plaintext-ciphertext correlation is analyzed between the aligned traces and the Hamming weights of 8-byte plaintext and ciphertext. Next, select a segment after the spike as the reference and perform static alignment. Figure 15 illustrates plaintext-ciphertext correlation coefficient curves for the forward-aligned and backward-aligned traces. Intuitively, the peaks on each correlation coefficient curve are very obvious, and the two curves look similar.

In order to test whether the execution time is consistent, we apply Fisher transform

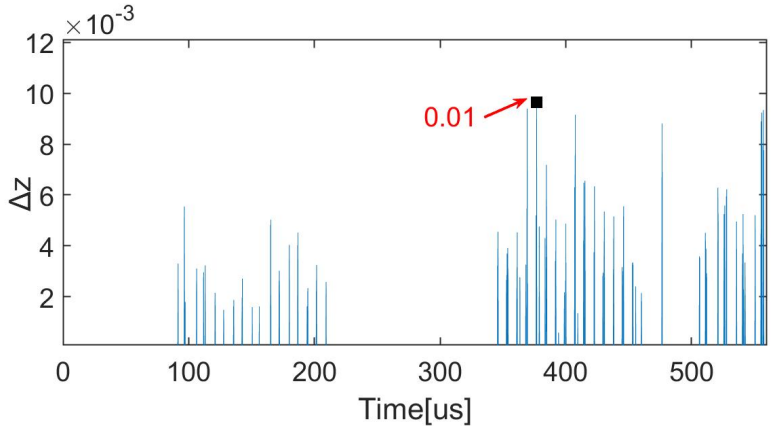


Figure 16: Difference between forward-aligned and backward-aligned correlation coefficient curves of smart card C

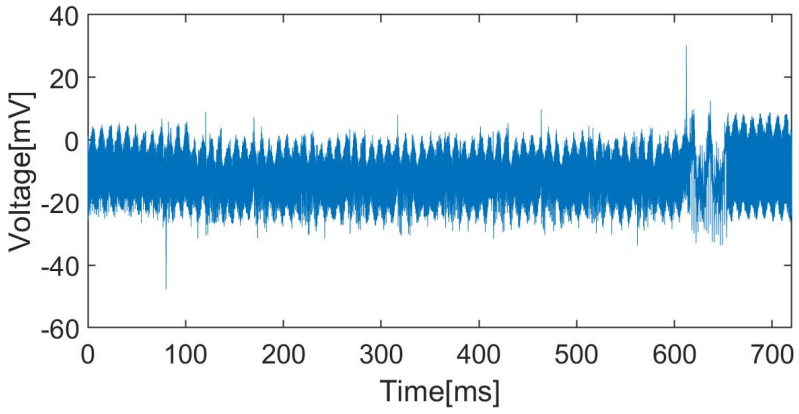


Figure 17: Trace of masked AES on a MCU with 1500000 sample points

to curves and truncate them. By observing curve characteristics, the threshold is selected as 0.07. Figure 15 gives Fisher transform and truncation results. Then we calculate the difference of two truncated curves, as shown in Figure 16, and calculate the p-value for each point according to (4). We have $p_{min} = 2 \times (1 - CDF(\frac{0.011}{\sigma})) \approx 0.34 > 0.05$ with $\sigma = \frac{1}{\sqrt{N-3}} \approx 0.01$. It can be determined that the algorithm implemented by smart card C has no obvious timing leakage, and the timing analysis test passes.

4.5 Timing Analysis Test on a MCU with masked AES

In this subsection the trace acquisition platform in Figure 7a is used for the experiment. We implemented a masked AES on the MCU and set three secret keys as

$$k_1 = \texttt{0x11223344556677889AABBCCDDEEFF00},$$
$$k_2 = \texttt{0x11111111111111111111111111111111},$$
$$k_3 = \texttt{0x1234567890abcdeffedcba0987654321}.$$

For each keys, 6000 traces are collected with random plaintexts, each containing 1500000 sample points. A trace after low-pass filtering is shown in Figure 17. While the trace is

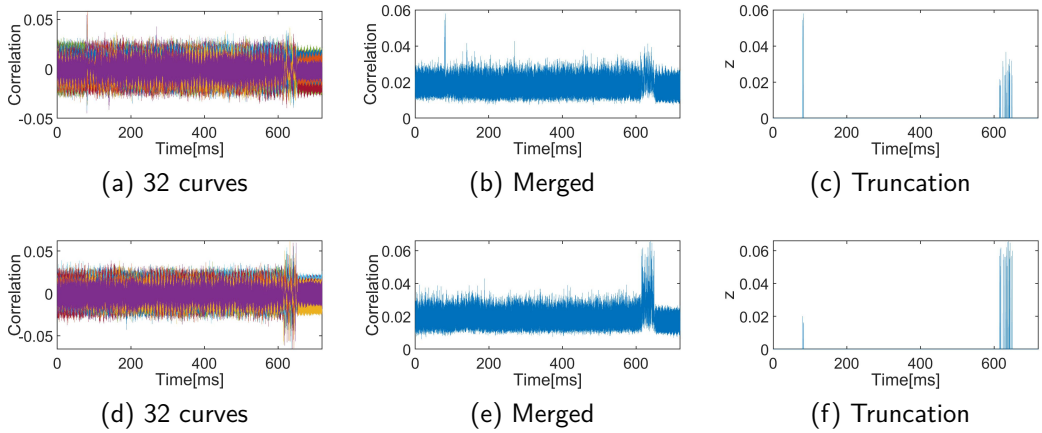


Figure 18: Correlation after amplification of the MCU with masked AES. (a), (b) and (c) are correlation coefficient curves, their merged curve and truncation curve for forward-aligned traces. (d), (e) and (f) are curves for backward-aligned traces.

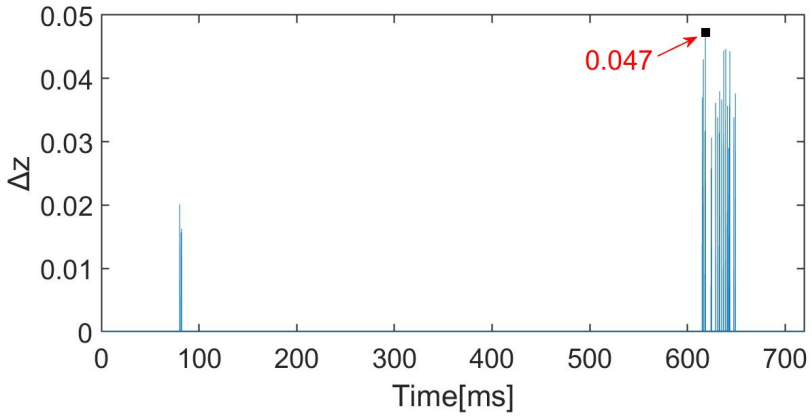


Figure 19: Difference between forward-aligned and backward-aligned correlation coefficient curves of the MCU with masked AES

not ideal, the algorithm encryption is believed to be performed in the interval of $80 \sim 620$ ms.

Select a segment before the encryption as the reference and perform static alignment. The plaintext and ciphertext are divided into 16 groups in bytes respectively. Then the plaintext-ciphertext correlation is analyzed. Figure 18a and 18b illustrate 32 plaintext-ciphertext correlation coefficient curves and their merged curve. After that, we select a segment after the encryption as the reference and repeat the operation. As shown in Figure 18, correlation coefficient curves have spikes regardless of whether it is aligned forward or backward, except that the correlation values change with different alignment. The difference in values reflects the difference in execution time.

Further, we apply Fisher transformation to the curves and truncated them with a threshold of 0.05. After calculating the difference of two truncated curves, as shown in Figure 19, we calculate the p-value for each point according to (4). $p_{min} = 2 \times (1 - CDF(\frac{0.047}{\sigma})) \ll 0.05$ when $\sigma = \frac{1}{\sqrt{N-3}} \approx 0.01$. It can be determined that the masked AES implemented on MCU has obvious timing leakage, and the timing analysis test fails.

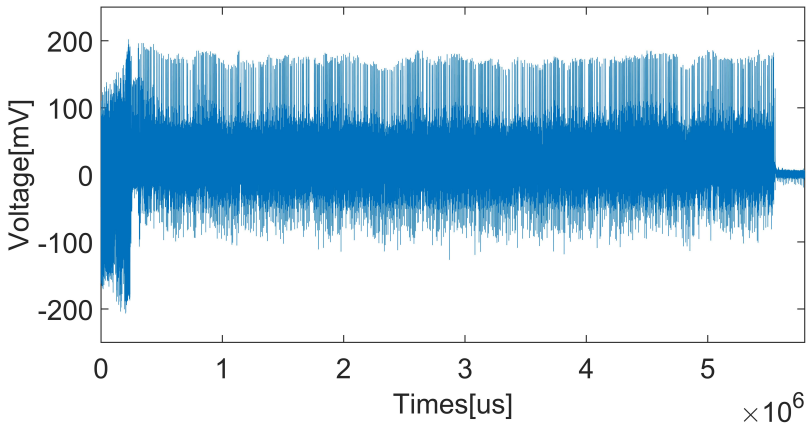


Figure 20: Trace of RSA on a MCU with 5800000 sample points

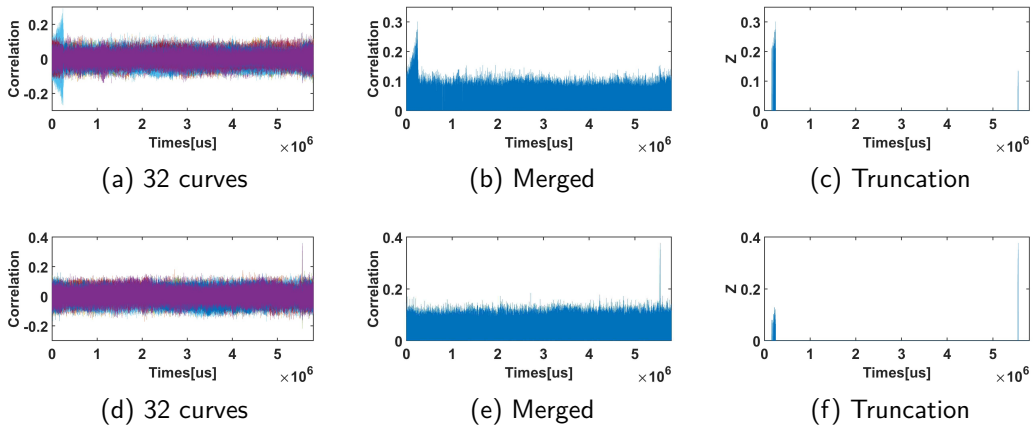


Figure 21: Correlation after amplification of the MCU with RSA. (a), (b) and (c) are correlation coefficient curves, their merged curve and truncation curve for forward-aligned traces. (d), (e) and (f) are curves for backward-aligned traces.

4.6 Timing Analysis Test on a MCU with RSA

In this subsection the trace acquisition platform in Figure 7b is used for the experiment. We implemented a RSA on the MCU and collect 1000 traces with random plaintexts, each containing 5800000 sample points.A trace after low-pass filtering is shown in Figure 20.

Select a segment before the encryption as the reference and perform static alignment. The plaintext and ciphertext are divided into 8 groups in bytes respectively. Then the plaintext-ciphertext correlation is analyzed. Figure 21a and 21b illustrate 16 plaintext-ciphertext correlation coefficient curves and their merged curve. After that, we select a segment after the encryption as the reference and repeat the operation. As shown in Figure 21, correlation coefficient curves have spikes regardless of whether it is aligned forward or backward, except that the correlation values change with different alignment. The difference in values reflects the difference in execution time.

Further, we apply Fisher transformation to the curves and truncated them with a threshold of 0.2. After calculating the difference of two truncated curves, as shown in Figure 19, we calculate the p-value for each point according to (4). $p_{min} = 2 \times (1 -$

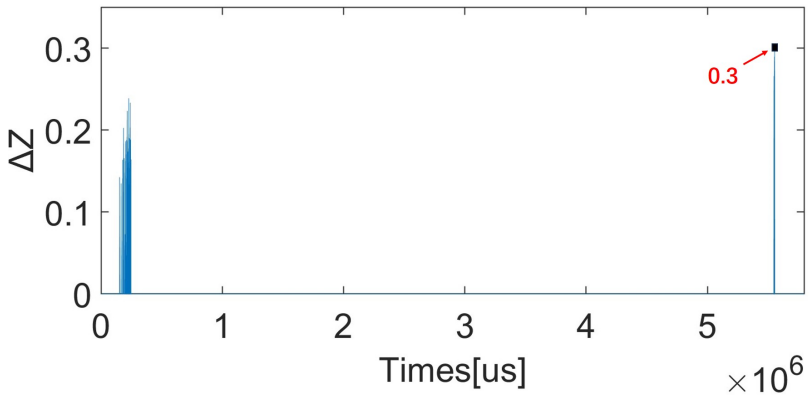


Figure 22: Difference between forward-aligned and backward-aligned correlation coefficient curves of the MCU with RSA

$CDF(\frac{0.047}{\sigma})) \ll 0.05$ when $\sigma = \frac{1}{\sqrt{N-3}} \approx 0.03$. It can be determined that the RSA implemented on MCU has obvious timing leakage, and the timing analysis test fails.

4.7 Discussion

We conducted experiments on different devices to demonstrate the reliability of our method. In addition, we will discuss several important aspects.

Alignment The core of our approach is to perform forward alignment and backward alignment, and then detect changes through correlation analysis. During trace alignment, it may be difficult to find the optimal segment as a reference. In fact, we do not have to choose the optimal segments. Instead, we can select spike segments, as long as the plaintext correlation of the forward alignment is not lower than that of the backward alignment, and the ciphertext correlation of the forward alignment is not higher than that of the backward alignment.

Peaks In general, there are two significant peaks in correlation coefficient curves representing points that have a strong correlation with plaintexts and ciphertexts. However, in some cases, correlation coefficient curves may have only one peak. For this case, if traces are well aligned, our timing analysis still works.

SNR Our approach works when traces with high SNR are obtained. The SNR level is judged mainly on the basis of whether it affects the calculation of correlation. A relationship between SNR and the correlation coefficient is represent as

$$\rho(P_{total}, H) = \frac{\rho(P_{leakage}, H)}{\sqrt{1 + 1/SNR}},$$

where the total power/EM value P_{total} is the sum of signal components $P_{leakage}$ and noise components P_{noise}. $\rho(P_{leakage}, H)$ could be obtained by simulation, or we can collect some traces to calculate a pair of ρ^* and SNR^* [22], then ρ for an arbitrary SNR can be calculated by

$$\rho(P_{total}, H) = \rho^* \frac{\sqrt{1 + \frac{1}{SNR^*}}}{\sqrt{1 + \frac{1}{SNR}}}.$$

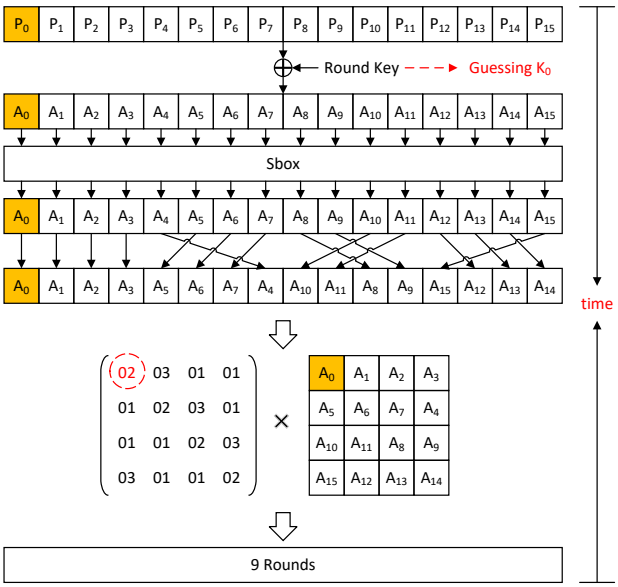


Figure 23: An SCA attack on AES

The number of traces n necessary for determining a significant difference between zero and ρ is calculated by

$$n = 3 + 8 \times \frac{z^2_{1-\alpha}}{\ln^2 \frac{1+\rho}{1-\rho}},$$

where α is a confidence level and $z_{1-\alpha}$ is a value obtained from a percentage point of the normal distribution [23]. The high SNR is to ensure that plaintext and ciphertext correlations can be distinguished within a limited number of traces n.

extract the secret keys After identifying the timing leakage, the next step is to consider how to conduct an SCA attack. For asymmetric ciphers like RSA, the timing difference may come from the fact that the subtraction happens or not during the multiplication and a timing attack is launched based on the leakage [24]. For block ciphers like AES, the timing difference may come from time-inconsistent algorithm implementations. For example, an XOR happens only when a carry bit occurs in multiplication of $0x02$ in the MixColumn operation. Here we try to give an SCA attack on AES. Taking the first key byte k_0 as an example, as shown in Figure 23, the attack works as follows:

1. Perform AES with numerous random plaintexts.
2. Guess a value of k_0.
3. Construct two sets of plaintexts depending on whether an XOR occurs when multiplying $0x02$.
4. If the timing leakages of two sets are statistically distinguishable, then the guessing value is a candidate. Otherwise, it is discarded.

Then the rest of key bytes can be retrieved with this method. We recommend using a time-consistent implementation to avoid such potential risks.

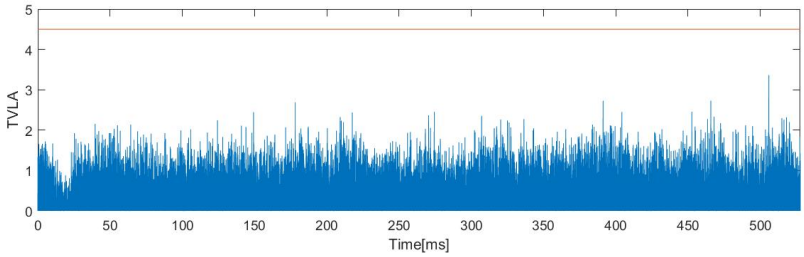


Figure 24: TVLA on masked AES on the MCU

Table 2: A new timing analysis framework

Timing analysis method	Scenario	Typical devices	Accuracy
I/O channel	GPIO	Chip board	High
	I/O signal	Smart card	High or Medium
Plaintext-ciphertext correlation	power/EM signal	Smart card	High
Average denoising	Communication with computer	HSM	Very Low

TVLA Compared with traditional power/EM analysis, we make full use of timing leakage, that is, our method is more concerned with leaks on the horizontal time axis than leaks on the vertical power/EM axis. For example, although Figure 24 illustrates that the masked AES implemented by MCU passes TVLA, it does have timing leakage as shown in Sec 4.5. TVLA cannot replace our timing analysis when testing cryptographic modules.

5 Leakage Analysis Framework for Timing Analysis

In this section, we propose a new timing analysis framework which divides target devices into three scenarios and provides corresponding methods for each scenario, as shown in Table 2. Figure 25 shows how our framework selects the appropriate method for timing analysis.

GPIO interface or I/O signal This scenario has the strictest requirements that the cryptographic devices can be triggered through a GPIO interface or I/O signal. For this scenario, the detector can obtain the accurate execution time and conduct statistical analysis. More details are described in Section 5.1.

High SNR power/EM signal When triggering through the I/O channel is not feasible but high SNR power/EM traces are available, we adopt the timing analysis method based on plaintext-ciphertext correlation analysis in Section 3.

No trigger, low SNR power/EM signal We provide a timing analysis method based on average denoising, which can be used as a supplement when the first two methods cannot be used. More details are described in Section 5.2.

5.1 Timing Analysis Based on I/O Channel

Depending on triggering modes, we choose different methods to calculate the execution time. For GPIO interfaces, the inspector can ask the manufacturer to pull up the GPIO level before the algorithm starts and pull down the GPIO level after the algorithm. For

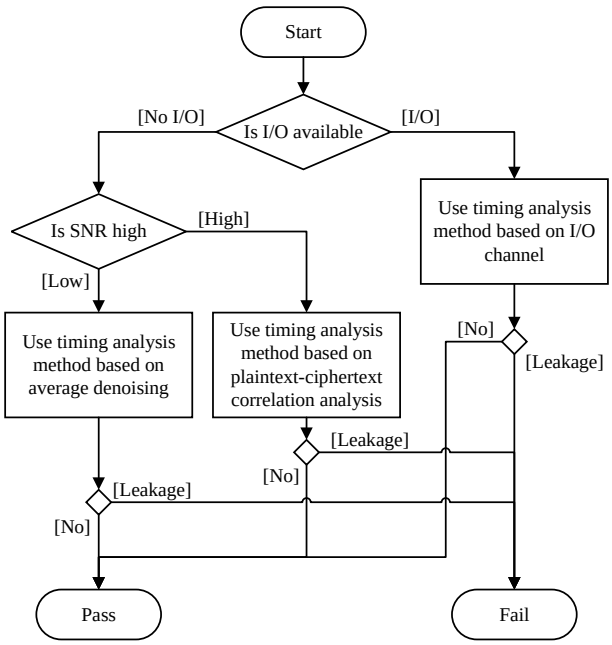


Figure 25: Timing analysis framework

I/O signals, according to the ISO/IEC 7816 protocol [25], the I/O level is high when bit 1 is transmitted and low when bit 0 is transmitted. After data transmission is completed, I/O returns to a high level. Thus, it is difficult to separate transmission of the last consecutive bits 1s from the subsequent encryption operation. So we select the total time of data transmission and algorithm execution. Using the total time is equivalent to using the execution time since data of the same length take the same transmission time.

Following the testing framework in Section 2.1, the method consists of the following steps.

1. Given a fixed plaintext and random keys, encrypt m times to collect m signal traces of I/O channels, with each trace containing k sampling points. Denote the sampling rate as s.

2. Calculate the execution time for each trace.

   - If the device is triggered through the GPIO interface, set the voltage threshold v_0. For each trace, select the interval where the voltage changes from low level to high level and find the first sample point $i(i < k)$ in the interval with the sample value above v_0. Then find the first sample point $j(i < j < k)$ which has the value below v_0 in the interval where the voltage changes from high level to low level. The execution time is $\frac{j-i}{s}$.
   - If the device is triggered through the I/O signals, set the voltage threshold v_0. For each trace, select the interval where the voltage changes from high level to low level after the algorithm starts and find the first sample point $j(j < k)$ in the interval with the value below v_0. Since trigger delay is set as 0, the execution time is $\frac{j-1}{s}$.

3. Calculate the average execution time for each key to get n average values $T_1, T_2, \cdots, T_n$. Then calculate the difference Δ between each two time values and compare it with

a clock cycle ε. If there exists $\Delta > \varepsilon$, the test fails, otherwise continue to the next step.

4. Given a fixed key and random plaintexts, encrypt m times to collect m signal traces of I/O channels. Then calculate the execution time following the process in Step 2 and Step 3. If there exists $\Delta > \varepsilon$, the test fails, otherwise the test passes.

5.2 Timing Analysis Based on Computer Average Denoising

The method is based on average denoising and the procedure is as follows:

1. Given a fixed key (resp. plaintext) and several different plaintexts (resp. keys), get the execution time through communication between the PC and devices, and then average the time values of each plaintext (resp. key) for denoising.

2. Calculate the difference of time for different plaintexts (resp. keys). If there is a time difference greater than a clock cycle, there exists timing leakage and the test fails. Otherwise, it passes.

This method is similar to the I/O channel-based method but has a low signal-to-noise ratio, resulting in inaccurate results. The standard deviation of noise is one of the most important factors affecting analysis results. Suppose that the measured time $T_1, T_2, T_3, \cdots, T_n$ are independently and identically distributed with the mean μ and standard deviation σ. According to the center limit theorem, the standard deviation of the average $\overline{T}$ is $\frac{\sigma}{\sqrt{n}}$. Since the test passes only when the difference of execution time is less than the clock cycle ε, $\frac{\sigma}{\sqrt{n}}$ should be less than ε, thus the measurement number should meet $n > \frac{\sigma^2}{\varepsilon^2}$. Taking a chip with a clock frequency of 10MHz as an example, assuming that σ is 1ms, n should be greater than $1,000,000,000$. Therefore, we do not recommend using it for timing analysis.

6 Conclusion

Our paper is focused on how to perform timing analysis when the standard time analysis is not suitable and proposes a new method which takes advantage of timing leakage in power/EM channels. Based on plaintext-ciphertext correlation analysis, our method successfully measures the difference in execution segment length and determines whether timing leakage exists. In addition, a leakage analysis framework of timing analysis is put forward for different scenarios.

The difficulty of timing analysis comes from the choice of measurement and analysis methods for different devices. Our paper has shown how this can be done. However, the method for the last scenario needs to be improved. Besides, apart from the timing analysis method based on plaintext-ciphertext correlation analysis, our future work is to find efficient ways to exploit timing leakage in power/EM channels and overcome the problem of delay noise.

Acknowledgements

This work was supported by the National Key R&D Plan of China (2022YFB3103800), the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province (Grant No. SKLACSS-202207) and National Natural Science Foundation of China (62302036, 62272047). The main corresponding author is An Wang.

References

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, ser. Lecture Notes in Computer Science, N. Koblitz, Ed., vol. 1109. Springer, 1996, pp. 104–113. [Online]. Available: https://doi.org/10.1007/3-540-68697-5_9

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 388–397. [Online]. Available: https://doi.org/10.1007/3-540-48405-1_25

[3] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, ser. Lecture Notes in Computer Science, Ç. K. Koç, D. Naccache, and C. Paar, Eds., vol. 2162. Springer, 2001, pp. 251–261. [Online]. Available: https://doi.org/10.1007/3-540-44709-1_21

[4] D. Page, "Theoretical use of cache memory as a cryptanalytic side-channel," *IACR Cryptol. ePrint Arch.*, p. 169, 2002. [Online]. Available: http://eprint.iacr.org/2002/169

[5] J. Danger, N. Debande, S. Guilley, and Y. Souissi, "High-order timing attacks," in *Proceedings of the First Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC 2014, Vienna, Austria, January 20, 2014*, J. Knoop, V. Salapura, I. Koren, and G. Pelosi, Eds. ACM, 2014, pp. 7–12. [Online]. Available: https://doi.org/10.1145/2556315.2556316

[6] W. Schindler, "A timing attack against RSA with the chinese remainder theorem," in *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, ser. Lecture Notes in Computer Science, Ç. K. Koç and C. Paar, Eds., vol. 1965. Springer, 2000, pp. 109–124. [Online]. Available: https://doi.org/10.1007/3-540-44499-8_8

[7] Q. Guo, C. Hlauschek, T. Johansson, N. Lahr, A. Nilsson, and R. L. Schröder, "Don't reject this: Key-recovery timing attacks due to rejection-sampling in HQC and BIKE," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2022, no. 3, pp. 223–263, 2022. [Online]. Available: https://doi.org/10.46586/tches.v2022.i3.223-263

[8] D. Brumley and D. Boneh, "Remote timing attacks are practical," in *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003*. USENIX Association, 2003. [Online]. Available: https://www.usenix.org/conference/12th-usenix-security-symposium/remote-timing-attacks-are-practical

[9] B. B. Brumley and N. Tuveri, "Remote timing attacks are still practical," in *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings*, ser. Lecture Notes in Computer Science, V. Atluri and C. Díaz, Eds., vol. 6879. Springer, 2011, pp. 355–371. [Online]. Available: https://doi.org/10.1007/978-3-642-23822-2_20

[10] Y. Yarom, D. Genkin, and N. Heninger, "Cachebleed: A timing attack on openssl constant time RSA," in *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19,*

2016, Proceedings, ser. Lecture Notes in Computer Science, B. Gierlichs and A. Y. Poschmann, Eds., vol. 9813. Springer, 2016, pp. 346–367. [Online]. Available: https://doi.org/10.1007/978-3-662-53140-2_17

[11] *ISO/IEC 17825:2016 Information technology Security techniques Testing methods for the mitigation of non-invasive attack classes against cryptographic modules*, International Organization for Standardization Std., 2016.

[12] A. Moradi, B. Richter, T. Schneider, and F.-X. Standaert, "Leakage detection with the x2-test," *Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, Issue 1, pp. 209–237, 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/838

[13] C. Whitnall and E. Oswald, "A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules')," in *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11923. Springer, 2019, pp. 256–284. [Online]. Available: https://doi.org/10.1007/978-3-030-34618-8_9

[14] O. Bronchain, T. Schneider, and F.-X. Standaert, "Multi-tuple leakage detection and the dependent signal issue," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 2, p. 318345, Feb. 2019. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/7394

[15] *ISO/IEC 19790:2012 Information technology Security techniques Security requirements for cryptographic modules*, International Organization for Standardization Std., 2012.

[16] C. D. Walter and S. Thompson, "Distinguishing exponent digits by observing modular subtractions," in *Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings*, ser. Lecture Notes in Computer Science, D. Naccache, Ed., vol. 2020. Springer, 2001, pp. 192–207. [Online]. Available: https://doi.org/10.1007/3-540-45353-9_15

[17] W. Schindler, "A combined timing and power attack," in *Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12-14, 2002, Proceedings*, ser. Lecture Notes in Computer Science, D. Naccache and P. Paillier, Eds., vol. 2274. Springer, 2002, pp. 263–279. [Online]. Available: https://doi.org/10.1007/3-540-45664-3_19

[18] W. Schindler and C. D. Walter, "More detail for a combined timing and power attack against implementations of RSA," in *Cryptography and Coding, 9th IMA International Conference, Cirencester, UK, December 16-18, 2003, Proceedings*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 2898. Springer, 2003, pp. 245–263. [Online]. Available: https://doi.org/10.1007/978-3-540-40974-8_20

[19] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, ser. Lecture Notes in Computer Science, M. Joye and J. Quisquater, Eds., vol. 3156. Springer, 2004, pp. 16–29. [Online]. Available: https://doi.org/10.1007/978-3-540-28632-5_2

[20] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, ser. Lecture Notes in Computer Science, M. Soriano, S. Qing, and J. López, Eds., vol. 6476. Springer, 2010, pp. 46–61. [Online]. Available: https://doi.org/10.1007/978-3-642-17650-0_5

[21] F. Durvaux and F. Standaert, "From improved leakage detection to the detection of points of interests in leakage traces," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, M. Fischlin and J. Coron, Eds., vol. 9665. Springer, 2016, pp. 240–262. [Online]. Available: https://doi.org/10.1007/978-3-662-49890-3_10

[22] Y. Yano, T. Teshima, K. Iokibe, and Y. Toyota, "Experimental identification of relationship between leakage trace snr and correlation coefficient in differential power analysis," in *2019 Joint International Symposium on Electromagnetic Compatibility, Sapporo and Asia-Pacific International Symposium on Electromagnetic Compatibility (EMC Sapporo/APEMC)*, 2019, pp. 1–4.

[23] S. Mangard, E. Oswald, and T. Popp, *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[24] J. Dhem, F. Koeune, P. Leroux, P. Mestré, J. Quisquater, and J. Willems, "A practical implementation of the timing attack," in *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, ser. Lecture Notes in Computer Science, J. Quisquater and B. Schneier, Eds., vol. 1820. Springer, 1998, pp. 167–182. [Online]. Available: https://doi.org/10.1007/10721064_15

[25] *ISO/IEC 7816-3:2006 Identification cards Integrated circuit cards Part 3: Cards with contacts Electrical interface and transmission protocols*, International Organization for Standardization Std., 2006.