Towards Leakage Exploitation Rate Optimality in Template Attack

Guangjun Fan[1], Yongbin Zhou[2], Hailong Zhang[2], Dengguo Feng[1]

[1] Trusted Computing and Information Assurance Laboratory,Institute of Software,Chinese Academy of Sciences,Beijing,China
guangjunfan@163.com , feng@tca.iscas.ac.cn

[2] State Key Laboratory of Information Security,Institute of Information Engineering,Chinese Academy of Sciences,Beijing,China
{zhouyongbin,zhanghailong}@iie.ac.cn

Abstract. Template Attack is widely accepted to be one of the most powerful side-channel attacks, because it is usually assumed that one has a full knowledge of the targeted crypto devices and thus be well capable of characterizing the side-channel leakages. However, the question of whether Template Attack is really optimal in terms of leakage exploitation rate is still unclear. In this paper, we present a negative answer to this crucial question, by introducing a normalization process into classical Template Attack. On the theoretical side, we prove that our Normalized Template Attack is (strictly) better in terms of leakage exploitation rate than the classical Template Attack; on the practical side, we evaluate the key-recovery efficiency of Normalized Template Attack and its classical counterpart as well under identical scenarios, by performing attacks against both simulated and real power traces. Our experimental results show that the proposed method is valid and effective. Interestingly enough, this normalization process is of extremely low computation cost, and thus is very easy-to-use in practice. Therefore, we argue that this normalization process should be integrated into Template Attack as one necessary step in the future, so that one could better understand the practical threats of Template Attack.

Keywords: Cryptography, Side-Channel Attacks, Power Analysis Attack, Template Attack.

1 Introduction

Side-channel attacks belong to an important kind of cryptanalysis techniques on cryptographic implementations. As a matter of fact, many implementations of traditional cryptosystems even provably secure in black-box model were broken by side-channel attacks using electromagnetic radiation [1,5], running-time [2], fault detection [3], power consumption [4] and many more [6,7].

Among those side-channel attacks, Power Analysis Attack is the most studied one. Power Analysis Attack exploits the fact that the instantaneous power consumption of a cryptographic device depends on the data it processes and the operations it performs. As an important attack method in Power Analysis

Attack, S. Chari et al. presented Template Attack in [8]. Template Attack is a kind of profiled side-channel attacks and is widely accepted to be the strongest side channel attack from an information theoretic point of view [8]. Because it assumes that one knows all the details of the targeted device and possesses a device which is identical or similar to the targeted device. While such an assumption is limited, it holds in many cases and has been used in other side-channel attacks [21,22].

Template Attack is a *two-stage* attack method. The first stage is a profiling stage and the second stage is an extraction stage. In the profiling stage, one can accurately characterize signals and noises in different times and builds templates for each key-dependent operation with a device which is identical or similar to the targeted device. In the extraction stage, one exploits one or a limited number of traces and the templates to classify the correct key. Nowadays, with the development of embedded device, Template Attack becomes more practical. Additionally, Template Attack is also an important tool to evaluate the security strength of the implementation of a cryptographic algorithm.

In many real world settings, one can not classify the correct key with only a single trace in the extraction stage due to noises and the accuracy of templates. Therefore, one needs more than one traces to classify the correct key. According to different attack scenarios, one may apply maximum likelihood approach on the product or the sum of conditional probabilities obtained from traces and the templates in the extraction stage to classify the correct key. Let's show the two cases and some specific examples for them in the following.

Case 1: When the traces are statistically independent, one will apply maximum likelihood approach on the product of conditional probabilities [13]. For convenience, we call the classical Template Attack in this case as "Template Attack for Case 1".

Example for Case 1: When one can attack the output of the S-boxs in the first round of AES-128 with random message inputs choosing by himself, he will apply maximum likelihood approach on the product of conditional probabilities. Because the traces are statistically independent due to the outputs of the S-boxs are random.

Case 2: When the traces are not statistically independent, one may apply maximum likelihood approach on the sum of conditional probabilities when the key-dependent operations in different traces are the same [9]. For convenience, we call the classical Template Attack in this case as "Template Attack for Case 2".

Examples for Case 2: In the following two specific examples, one may apply maximum likelihood approach on the sum of conditional probabilities. Example 1: If one can not obtain any information about the input and output of some symmetric cipher, he can only attack the key scheduling mechanism of the symmetric cipher and obtain more than one traces about the same key in the extraction stage. Usually, this situation may be caused by the following two reasons. Reason one is that one only has limited control about the targeted device in the extraction stage and he can only measure the power consumption. Reason two is that the Hamming Weight of some sensitive intermediate value can not be recovered with probability 1 using a single trace due to noises for some device whose leakage function is Hamming Weight leakage function [18]. Example 2: When one tries to attack some fixed substantial intermediate value such as the secret key of some asymmetric cryptosystem, he can only obtain more than one trace with the same key and applies Template Attack for Case 2. For instance, many public key encryption schemes and digital signature schemes are based on the Discrete Logarithm Problem or the Decisional Diffie-Hellman assumption. These schemes usually need to compute g^x, where g is an element of a group of prime order q and $x \in \mathbb{Z}_q$ is the secret key. Note that x is fixed during every invocation. When sliding-window exponentiation method [19] is used to compute g^x, one could build templates for the window and applies Template Attack for Case 2.

1.1 Motivations

Usually, a side-channel attack method with higher leakage exploitation rate will has higher success rate [10]. Research on Template Attack has concentrated mainly on the profiling stage and try to build more accurate templates in order to increase success rate (leakage exploitation rate) when the amount of information can be exploited by one remain unchanged[1]. In the extraction stage of Template Attack, one directly applies maximum likelihood approach on the product or the sum of conditional probabilities obtained from traces and templates to classify the correct key. However, it is unknown that whether this way of exploiting traces and templates optimize leakage exploitation rate and result in the highest success rate though accurate templates are built in the profiling stage. Therefore, a natural and important question is that does there exist a better way to exploit traces and templates which has higher leakage exploitation rate in the extraction stage and achieves higher success rate than the classical Template Attack? In this paper, we try to answer this question.

It is well known that the effectiveness of Template Attack is mainly affected by the accuracy of templates, noises in the target device, and the number of traces used in the extraction stage. However, how the accuracy of templates and noises affect the effectiveness of Template Attack when the number of traces used in the extraction stage is fixed has not been established. In this paper, we try to find some *quantitative* factors in the extraction stage which affect the effectiveness of Template Attack.

1.2 Contributions

[1] For example, the adversary has a fixed number of traces in the profiling stage and the extraction stage.

The main contributions of this paper are two-fold as follows. Although Template Attack is widely accepted to be the strongest side channel attack, we first prove that leakage exploitation rate of the classical Template Attack is not optimal. This observation is obtained by a new way of exploiting traces and the templates which has higher leakage exploitation rate (higher success rate) than the classical Template Attack in the extraction stage. Consequently, we give out two improved attack methods for each one of the two cases of classical Template Attack, which use the new way. Second, we find a quantitative factor which affects the effectiveness of the classical Template Attack. This quantitative factor give us a better understanding of the classical Template Attack.

1.3 Related Work

Template Attack was introduced in [8]. In [9], C. Rechberger et al. provided answers to some basic and practical issues of Template Attack, such as how to select interesting points in an efficient way and how to preprocess noisy data. Template Attack for Case 2 was also presented in [9]. Principal subspace-based Template Attacks were investigated in [15,16]. However, due to their high computational requirements, principal subspace-based Template Attacks are not used widely [9]. In [17], Template Attack on an implementation of a block cipher that uses a masking scheme was introduced that retrieves the secret key. In [23], an efficient leakage characterization method in the profiling stage for Template Attack was introduced. A simple pre-processing technique of Template Attack, normalizing the trace means and variances from the training and targeted devices was evaluated for various test data set sizes in [20]. In [29], the assumption of Template Based DPA was relax with machine learning techniques. Template Attack assumes that the adversary possesses a device which is identical or similar to the targeted device and can fully control it. The paper [30] relaxes this assumption by generalizing Template Attack using a method based on a semi-supervised learning strategy. However, our important discoveries are not considered or neglected in the previous work.

1.4 Organization of This Paper

The rest of paper is organized as follows. In section 2, we review Template Attack and the above two attack scenarios. We show our new way and explain why it is more effective in section 3. Our new way can be used in the two attack scenarios yielding two improved attack methods. The two improved attack methods are verified by simulated and practical experiments in section 4. In section 5, we conclude this paper.

2 Preliminaries

In this section, we briefly review Template Attack. In Template Attack, there exists two stages. The first stage is a profiling stage and the second stage is a extraction stage. We will introduce the two stages in the following.

2.1 The Profiling Stage

In the profiling stage, one has a cryptographic device which is identical or similar to the targeted device. One derives some traces from this device. These traces are used to build templates for each key (key-dependent operation).

Let us assume there exist K different (sub)key $key_i, i = 0, 1, \ldots, K-1$ which need to be classified. There also exist K different key-dependent operations O_i with $i = 0, 1, \ldots, K-1$. Usually, one will generate K templates, one for each key-dependent operation O_i. In each one of the templates, there exists two parts. The first part in a template estimates the data-dependent portion of the side channel leakages. It is the average signal M_i for each of the operations. The second part in a template estimates the probability density of the noises in the side channel leakages. One can exploit advanced techniques [9,14] to choose N selected instants $(P_1, P_2, \ldots, P_N)$ in each sample. It is assumed that the noises in the side channel leakages approximately have a multivariate normal distribution with respect to the selected N interesting points $(P_1, P_2, \ldots, P_N)$. A N dimensional noise vector $n_i(S)$ is extracted from each sample S (a trace) representing the template's key dependency O_i as $n_i(S) = (S[P_1] - M_i[P_1], \ldots, S[P_N] - M_i[P_N])$. One computes the $(N \times N)$ covariance matrix C_i from these noise vectors. The probability density of the noises occurring under key-dependent operation O_i is given by the N dimensional multivariate Gaussian distribution $p_i(\cdot)$ where the probability of observing a noise vector $n_i(S)$ is

$$p_i(n_i(S)) = \frac{1}{\sqrt{(2\pi)^N |C_i|}} exp\Big(-\frac{1}{2} n_i(S)^T C_i^{-1} n_i(S)\Big) \quad n_i(S) \in \mathbb{R}^N, \qquad (1)$$

where $|C_i|$ denotes the determinant of C_i and C_i^{-1} its inverse.

2.2 The Extraction Stage

In this stage, one tries to classify the correct key with one or a limited number of traces obtained from the targeted device. Usually, due to noises and the accuracy of templates, one can not recover the correct key with only one single trace. When one obtains more than one trace in the extraction stage, according to different attack scenarios, his strategy to classify the correct key is to apply maximum likelihood approach on the product or the sum of conditional probabilities obtained from traces and the templates.

Assume that one obtains t traces (denoted by $S_1, S_2, \ldots, S_t$) in the extraction stage.

When the traces are statistically independent (Case 1), one will apply maximum likelihood approach on the product of conditional probabilities [13], i.e.

$$key_{ck} = argmax_{key_i}\Big\{\prod_{j=1}^{t} Pr(S_j|key_i), i = 0, 1, \ldots, K-1\Big\},$$

where $Pr(S_j|key_i) = p_{f(S_j,key_i)}(n_{f(S_j,key_i)}(S_j))$. The key_{ck} is considered to be the correct key. The output of the function $f(S_j, key_i)$ is the index of some key-dependent operation. For example, when one attacks the output of the first S-box (denoted by $Sbox$) in the first round of AES-128, one builds templates for each output of the S-box (The key-dependent operation is the output of the S-box.). In this case, $f(S_j, key_i) = Sbox(m_j \oplus key_i)$, where m_j is the plaintext of the power trace S_j.

When the traces are not statistically independent, in many real world settings, one can only obtains more than one traces with the same key-dependent operation in the extraction stage (Case 2). In this case, one can computes $\sum_{j=1}^{t} Pr(S_j|key_i)$ for each key $key_i, i = 0, 1, \ldots, K-1$ and apply maximum likelihood approach on the sum of conditional probabilities, i.e.

$$key_{ck} = argmax_{key_i}\Big\{\sum_{j=1}^{t} Pr(S_j|key_i), i = 0, 1, \ldots, K-1\Big\},$$

where $Pr(S_j|key_i) = p_{f(key_i)}(n_{f(key_i)}(S_j))$ [9]. The key_{ck} is considered to be the correct key. The output of the function $f(key_i)$ is the index of some key-dependent operation. In this case, the output of $f(key_i)$ only depends on key_i. For example, when one attacks the output of some S-box in the key expansion algorithm of AES-128, $f(key_i) = Sbox(key_i)$.

3 Normalized Template Attack

In this section, we first introduce the main idea of our new way which has higher leakage exploitation rate. Then we show our new way formally. Finally, we explain why our new way has higher leakage exploitation rate. We take Case 1 for example. For Case 2, we have similar results.

Our new way introduces a normalization process in the extraction stage and does not change the profiling stage. The normalization process makes the attack method has higher leakage exploitation rate because it reduces the effect of noises in each trace and the inaccuracy of templates by exploiting normalized conditional probability instead of conditional probability.

Now, we give out an improved Template Attack for Case 1 and Case 2 respectively, which uses our new way. The two improved Template Attacks are called "Normalized Template Attack for Case 1" and "Normalized Template Attack for Case 2" and are shown in Algorithm 1 and Algorithm 2 respectively. The profiling stages of the two improved Template Attacks are the same as the classical Template Attack. Therefore, we ignore the profiling stages here and only show the extraction stages of them.

For simplicity, we rewrite $Pr(S_j|key_i)$ as $P(i,j), i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t$.

```
Algorithm 1 Normalized Template Attack for Case 1
  Input: $P(i,j), i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t$
  Output: a candidate key $key_{ck}, ck \in \{0, 1, \ldots, K-1\}$

  Step 1 Computes the natural logarithm of each conditional probability $P(i,j)$, i.e.
      $H(i,j) = lnP(i,j), i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t.$
  Step 2 Computes
      $maxln(j) = max\{H(0,j), H(1,j), \ldots, H(K-1,j)\}$
  for each traces $S_j, j = 1, 2, \ldots, t$.
  Step 3 Computes normalized conditional probability $V(i,j)$ for each traces:
      $V(i,j) = exp\Big(\frac{maxln(j)}{H(i,j)}\Big), i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t.$
  Step 4 Applying maximum likelihood approach on $\prod_{j=1}^{t} V(i,j)$. Let
      $key_{ck} = argmax_i\{\prod_{j=1}^{t} V(i,j), i = 0, 1, \ldots, K-1\}.$
  Step 5 Return $key_{ck}$.
```

Step 1-Step 3 in Algorithm 1 and Algorithm 2 is called normalization process. Note that, the normalization process is of extremely low computation cost. For example, we assume that there are K keys and the adversary obtains t power traces in the extraction stage. Then the normalization process only need to compute tK times of real division. In the following, we explain why our new way has higher leakage exploitation rate.

For every trace S_j $(j = 1, 2, \ldots, t)$, we compute

$$maxln(j) = max\{H(0,j), \ldots, H(K-1,j)\}.$$

Clearly, for each conditional probability $P(i,j), i = 0, 1, \ldots, K-1$ obtained from a single trace S_j $(j = 1, 2, \ldots, t)$, there exists a real number $\alpha_{(i,j)} \in [0,1], i = 0, 1, \ldots, K-1$ such that $\alpha_{(i,j)} = maxln(j)/H(i,j)$. Note that the number $\alpha_{(i,j)}$ is proportional to $P(i,j)$ for a fixed j. Let

$$V(i,j) = e^{\frac{maxln(j)}{H(i,j)}} = e^{\alpha_{(i,j)}}, i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t.$$

We call $V(i,j)$ the normalized conditional probability. Assume that, for the correct key key_{ck}, we have

$$\prod_{j=1}^{t} V(ck,j) > \prod_{j=1}^{t} V(i,j), i \in \{0, 1, \ldots, K-1\} \backslash ck.$$

Algorithm 2 Normalized Template Attack for Case 2

```
**Input:** $P(i,j), i = 0,1,\ldots,K-1, j = 1,2,\ldots,t$
**Output:** a candidate key $key_{ck}, ck \in \{0,1,\ldots,K-1\}$

**Step 1** Computes the natural logarithm of each conditional probability $P(i,j)$, i.e.
    $H(i,j) = lnP(i,j), i = 0,1,\ldots,K-1, j = 1,2,\ldots,t.$
**Step 2** Computes
    $maxln(j) = max\{H(0,j), H(1,j),\ldots,H(K-1,j)\}$
for each traces $S_j, j = 1,2,\ldots,t$.
**Step 3** Computes normalized conditional probability $V(i,j)$ for each traces:
    $V(i,j) = exp\Big(\frac{maxln(j)}{H(i,j)}\Big), i = 0,1,\ldots,K-1, j = 1,2,\ldots,t.$
**Step 4** Applying maximum likelihood approach on $\sum_{j=1}^{t} V(i,j)$. Let
    $key_{ck} = argmax_i\{\sum_{j=1}^{t} V(i,j), i = 0,1,\ldots,K-1\}.$
**Step 5** Return $key_{ck}$.
```

This assumption is reasonable. Because for the correct key key_{ck} and every trace S_j $(j = 1,2,\ldots,t)$, $\alpha_{(ck,j)}$ is much closer to 1 than $\alpha_{(i,j)}$ for the wrong keys $key_i, i \in \{0,1,\ldots,K-1\}\backslash ck$ with high probability. If

$$\prod_{j=1}^{t} P(ck,j) > \prod_{j=1}^{t} P(i,j), i \in \{0,1,\ldots,K-1\}\backslash ck,$$

Template Attack for Case 1 will return the correct key key_{ck}. However, when noises are large and/or the templates are not very accurate, the classical Template Attack may return a wrong key $key_{wk}, wk \in \{0,1,\ldots,K-1\}\backslash ck$. We will show one of the reasons about why the classical Template Attack return a wrong key in the following.

We divide the t traces $\{S_1, S_2,\ldots,S_t\}$ into two sets. In the first set $Set1$, there are u samples $\{S_{i_1},\ldots,S_{i_u}\}$ satisfy

$$P(ck,i_1) > P(wk,i_1),\ldots,P(ck,i_u) > P(wk,i_u).$$

In the second set $Set2$, there are $t-u$ samples $\{S_{j_1},\ldots,S_{j_{t-u}}\}$ satisfy

$$P(ck,j_1) \le P(wk,j_1),\ldots,P(ck,j_{t-u}) \le P(wk,j_{t-u}).$$

Let $P1_{ck} = \prod_{k=1}^{u} P(ck,i_k)$, $P2_{ck} = \prod_{k=1}^{t-u} P(ck,j_k)$, $P1_{wk} = \prod_{k=1}^{u} P(wk,i_k)$, and $P2_{wk} = \prod_{k=1}^{t-u} P(wk,j_k)$.

Let $V1_{ck} = \prod_{k=1}^{u} V(ck,i_k)$, $V2_{ck} = \prod_{k=1}^{t-u} V(ck,j_k)$, $V1_{wk} = \prod_{k=1}^{u} V(wk,i_k)$, and $V2_{wk} = \prod_{k=1}^{t-u} V(wk,j_k)$.

According to the definition, we have $P1_{ck} > P1_{wk}$, $P2_{ck} \le P2_{wk}$, $V1_{ck} > V1_{wk}$, and $V2_{ck} \le V2_{wk}$. Due to

$$\prod_{j=1}^{t} V(ck,j) = V1_{ck}V2_{ck} > V1_{wk}V2_{wk} = \prod_{j=1}^{t} V(wk,j),$$

we have

$$\frac{V1_{ck}}{V1_{wk}} > \frac{V2_{wk}}{V2_{ck}}. \tag{2}$$

However, if Template Attack for Case 1 returns key_{wk} as the output, thus

$$\prod_{j=1}^{t} P(ck, j) = P1_{ck}P2_{ck} < P1_{wk}P2_{wk} = \prod_{j=1}^{t} P(wk, j).$$

Therefore, we have

$$\frac{P1_{ck}}{P1_{wk}} < \frac{P2_{wk}}{P2_{ck}}. \tag{3}$$

We know that $maxln(j)$ belongs to the interval $(-\infty, 0)$. The values $maxln(j)$ are different for different traces with same operation and data due to noises in the traces and accuracy of the templates. For different traces with the same operation and different data, the above fact still holds. For example, we tried to attack the output of the first S-box in the first round of AES-128 with simulated traces[1]. Table 1 shows the variance of $maxln(j)$ of 200 simulated traces with the same operation and data[2].

Table 1. the variance of $maxln(j)$

np	5000	7000	9000
Variance	2.3481	2.0230	1.9404

In Table 1, the number of simulated traces used to build the templates is denoted by np. The three groups of templates built by different number of simulated traces represent different level of accuracy of templates. From table 1, we can see that the variance of $maxln(j)$ is not small and reduces with the increase of of np. Now, we assume that $V(ck, S)$ and $V(wk, S)$ are two fixed values and $V(ck, S) > V(wk, S)$ for some trace S. When $maxln(S)$ is large (i.e. $maxln(S)$ is close to 0.), the value of $exp(H(ck, S)) - exp(H(wk, S))$ is large. When $maxln(S)$ is small (i.e. $maxln(S)$ is far to 0.), the value of $exp(H(ck, S)) - exp(H(wk, S))$ is small even if $V(ck, S)$ and $V(wk, S)$ are two fixed values. This property of function $f(x) = e^x$ may cause equation (2) and equation (3) hold simultaneously and will reduce the success rate of classical Template Attack when one computes the produce or the sum of the conditional probabilities.

For example, Figure 1 shows the function $f(x) = e^x$ for $x \in [-10, -4]$. We assume the adversary obtains two traces S_1 and S_2 in the extraction stage. The adversary computes

[1] Please see Section 4.1 for more details about how the simulated traces are generated. The standard deviation of simulated Gaussian noise is 4.

[2] In each trace of the 200 simulated traces, the subkey and the input of S-box keep unchanged.

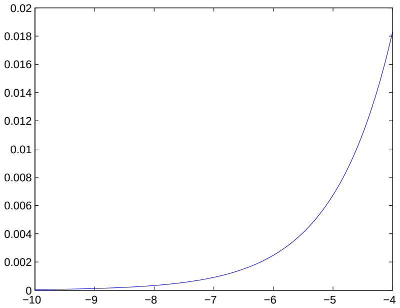


Fig. 1. the function $f(x) = e^x$ for $x \in [-10, -4]$

$$maxln(S_1) = -4, H(ck, 1) = -4, H(wk, 1) = -6.4,$$

$$maxln(S_2) = -6, H(ck, 1) = -6, H(wk, 1) = -9.23.$$

Normalized Template Attack will return the correct key ck because

$$exp(V(ck, 1))exp(V(ck, 2)) = exp(1)exp(0.65)$$

$$> exp(V(wk, 1))exp(V(wk, 2)) = exp(1)exp(0.625).$$

However, the classical Template Attack will return wk as the answer because

$$exp(H(ck, 1))exp(H(ck, 2)) = exp(-4)exp(-9.23)$$

$$< exp(H(wk, 1))exp(H(wk, 2)) = exp(-6.4)exp(-6).$$

Normalized Template Attack exploits normalized conditional probability which is more effective than traditional conditional probability of classical Template Attack. Therefore, we believe that Normalized Template Attack has success rate (leakage exploitation rate) than classical Template Attack. We know that $V(i, j) \in [e^0, e]$ and $P(i, j) = e^{H(i,j)} = V(i, j)^{\frac{H^2(i,j)}{maxln(j)}}, i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t$. Therefore, the value $H^2(i, j)/maxln(j)$ will affect the effectiveness of classical Template Attack for both Case 1 and Case 2. In the next section, we will verify our observation by experiments.

4 Experiments

In this section, we evaluate the key-recovery effectiveness of Normalized Template Attack and the classical Template Attack for Case 1 and Case 2 by experiments from which show the leakage exploitation rate of our new way and the classical way. We analyze the effectiveness by using success rate [10] as a metric.

For implementation of a cryptographic algorithm with countermeasures, the adversary may first use some method to delete the countermeasures and then tries to attack the implementation with classical attacks (Such as Template Attack). For example, if an adversary has traces with random delays. He may first use the method in paper [31] to remove the random delays and then execute classical attacks. Therefore, we take unprotected AES-128 implementation as example for Case 1 and Case 2 of Template Attack.

On one hand, we will evaluate the performance of Normalized Template Attack and the classical Template Attack under different noise level using simulated power traces. On the other hand, we will perform Normalized Template Attack and the classical Template Attack against real power traces.

For simplicity, let np denote the number of traces used in the profiling stage and let ne denote the number of traces used in the extraction stage. Different number of traces used in the process of building the templates means that the templates have different level of accuracy. Different number of traces used in the extraction stage represents different amount of information can be exploited.

In each figure showing experiment results, Normalized Template Attack (such as Normalized Template Attack for Case 1 or Normalized Template Attack for Case 2) is denoted by "Normalized TA" and the classical Template Attack (such as Template Attack for Case 1 or Template Attack for Case 2) is denoted by "Classical TA".

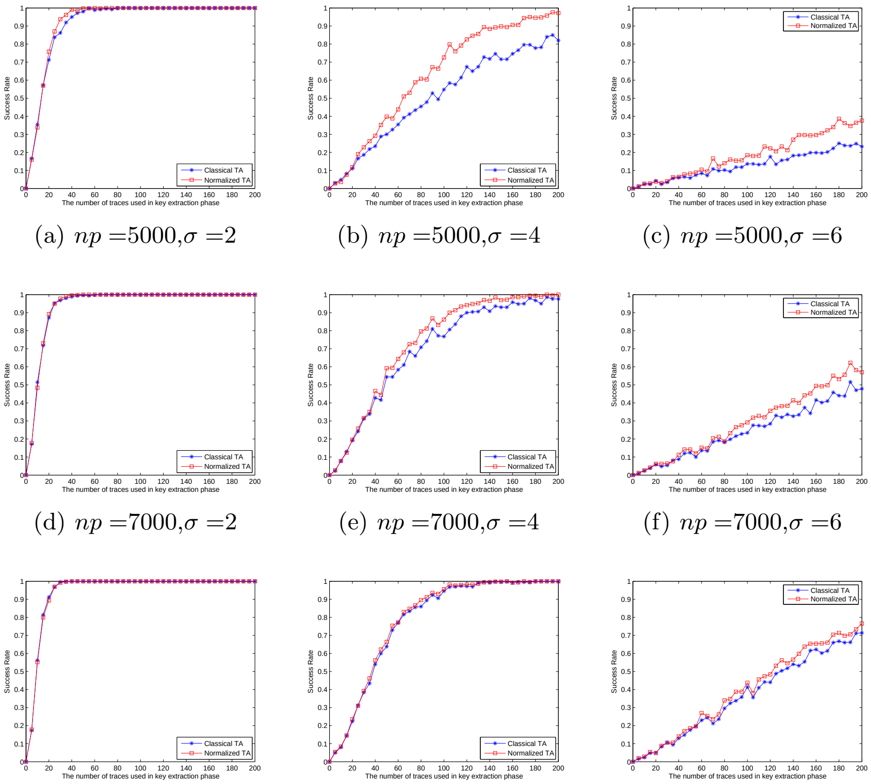
4.1 Simulated Experiments

We will introduce simulated experiments about Normalized Template Attack for Case 1 at first. Then, simulated experiments about Normalized Template Attack for Case 2 will be shown. In all simulated experiments, the standard deviation of simulated Gaussian noise is denoted by σ.

4.1.1 Case 1

In simulated scenarios, we chose the output of the first S-box in the first round of unprotected AES-128 as the target intermediate value. The Hamming Weight power model [12] was adopted to test the effectiveness of Normalized Template Attack for Case 1 and Template Attack for Case 1. We employed three different noise levels to test the influence of noises on the performance of the two attacks. The standard deviation of simulated Gaussian noise for the three noise levels were 2, 4, and 6.

For each noise level, we did as follows. We used 5000, 7000, and 9000 simulated power traces to build the 256 templates respectively (Because the output of the S-box is 8 bits long, we needed to build a template for each output of the S-box.). The three groups of simulated power traces were generated with a fixed sub key and random plaintext inputs. We generated additional 20000 simulated power traces which were used in the extraction stage with another fixed sub key and random plaintext inputs. We tested the success rate of Normalized Template Attack for Case 1 (denoted by $SR_{(ne,NTA1)}$) and Template Attack for Case 1 (denoted by $SR_{(ne,TA1)}$) when one can use ne traces in the extraction stage as follows. We repeated the two attacks 500 times. For each time, we chose ne traces from the 20000 simulated power traces uniformly at random. Both Normalized Template Attack for Case 1 and Template Attack for Case 1 used the same templates and the same ne traces in the extraction stage. We respectively recorded how many times the two attacks can successfully recover the correct key (denoted by $num_{(ne,NTA1)}$ for Normalized Template Attack for Case 1 and $num_{(ne,TA1)}$ for Template Attack for Case 1). Then, We computed the success rate $SR_{(ne,NTA1)}$ ($SR_{(ne,NTA1)} = num_{(ne,NTA1)}/500$) and $SR_{(ne,TA1)}$ ($SR_{(ne,TA1)} = num_{(ne,TA1)}/500$). We will show $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ for different ne in Figure 2.



(a) np =5000,σ =2

(b) np =5000,σ =4

(c) np =5000,σ =6

(d) np =7000,σ =2

(e) np =7000,σ =4

(f) np =7000,σ =6

(g) np =9000,σ =2

(h) np =9000,σ =4

(i) np =9000,σ =6

Fig. 2. Simulated Experiments Results of Normalized Template Attack for Case 1 and Template Attack for Case 1 ($SR_{(ne,NTA1)}$,$SR_{(ne,TA1)}$)

From Figure 2, we can see that Normalized Template Attack for Case 1 is more effective than Template Attack for Case 1 when the templates are not very accurate (The templates are built with less simulated power traces.) and the noise level is high. When one uses more simulated power traces to build the templates in the profiling stage, the success rate of Normalized Template Attack for Case 1 are not lower than that of Template Attack for Case 1. Hence, we only consider the case that one can only use less simulated power traces to build the templates in the profiling stage.

4.1.2 Case 2

To verify the Normalized Template Attack for Case 2, we attacked the key expansion algorithm of unprotected AES-128 as an example. Algorithm 3 in Appendix A describes the key expansion algorithm of unprotected AES-128.

RotWord in Algorithm 3 performs a one-byte circular left shift on a word. This means that an input word [b0,b1,b2,b3] is transformed into [b1,b2,b3,b0]. SubWord in Algorithm 3 performs a byte substitution on each byte of its input word, using the S-box. If the adversary can recover w[3],w[7],w[11], and w[15], then he can recover the main key key[0],key[1],. . .,key[15] using the internal structure of the key expansion algorithm easily. Note that w[3],w[7],w[11], and w[15] are the input of RotWord. And the output of RotWord is the input of SubWord. Therefore, one can try to attack the outputs of the S-boxes in SubWord and to recover w[3],w[7],w[11], and w[15] completely if he obtains the output of every S-box in SubWord successfully. In all of our simulated experiments for Normalized Template Attack for Case 2 and Template Attack for Case 2, we attacked an output of a S-box in SubWord and tried to recover key[15] in w[3] as an example. The processes of attacking other key bytes in w[3],w[7],w[11], and w[15] are similar.

We adopted ID power model [11] to test the effectiveness of Normalized Template Attack for Case 2 and Template Attack for Case 2. If the intermediate value computed by the device is *mid*, the ID power model will leak *mid* itself as the simulated power consumption. For Hamming Weight power model [12], different intermediate values with the same Hamming Weight will have the same simulated power consumption. Therefore, it is very difficult to classify a specific intermediate value accurately when the intermediate values of different power traces are fixed. Hence, we used ID power model. We used 20000, 40000, and 60000 simulated power traces to build the 256 templates respectively[1]. The value of key[15] of each simulated power trace was generated randomly. We employed three different noise levels to test the influence of noises on the performance of the two attacks. The standard deviation of simulated Gaussian noise for the three noise levels were 2, 4, and 6.

In our simulated experiments, we chose 32 random values for key[15]. For each one of the 32 values of key[15], we generated 600 simulated power traces.

[1] If we use less traces in the profiling stage, the success rate of both the two attack methods are low though the success rate of Normalized Template Attack for Case 2 will also much higher than that of Template Attack for Case 2. Therefore, we used 20000, 40000, and 60000 simulated power traces to build 256 templates in order to give out a clearer situation.

We tested the success rate of Normalized Template Attack for Case 2 (denoted by $SR_{(ne,NTA2)}$) and the success rate of Template Attack for Case 2 (denoted by $SR_{(ne,TA2)}$) when one can use ne traces in the the extraction stage as follows. For the ith ($i = 1, 2, \ldots, 32$) value of key[15], we repeated the two attacks 128 times. For each time, we chose ne simulated power traces uniformly at random from the corresponding 600 simulated power traces. Both Normalized Template Attack for Case 2 and Template Attack for Case 2 used the same templates and the same ne traces in the extraction stage. For the ith ($i = 1, 2, \ldots, 32$) value of key[15], we respectively recorded how many times the two attacks can recover the output of S-box successfully (denoted by $num_{(ne,i,NTA2)}$ for Normalized Template Attack for Case 2 and $num_{(ne,i,TA2)}$ for Template Attack for Case 2). For the ith ($i = 1, 2, \ldots, 32$) value of key[15], we use $sr_{(ne,i,NTA2)}$ and $sr_{(ne,i,TA2)}$ to denote the success rate (i.e. $sr_{(ne,i,NTA2)} = num_{(ne,i,NTA2)}/128$ and $sr_{(ne,i,TA2)} = num_{(ne,i,TA2)}/128$). The success rate of the two attacks for the case one using ne traces in the extraction stage were computed by

$$SR_{(ne,NTA2)} = \frac{\sum_{i=1}^{32} sr_{(ne,i,NTA2)}}{32}, SR_{(ne,TA2)} = \frac{\sum_{i=1}^{32} sr_{(ne,i,TA2)}}{32}.$$

We will show $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$ for different ne in Figure 3. From Figure 3, we can see that Normalized Template Attack for Case 2 is much more effective than Template Attack for Case 2.

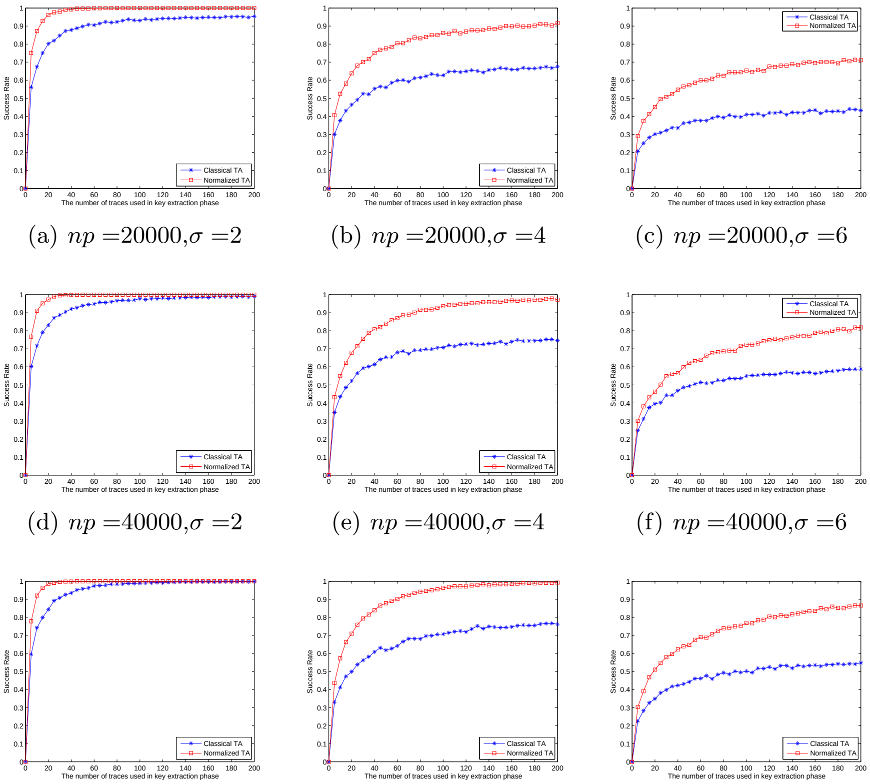
4.2 Practical Experiments

We performed our two new attack methods against real power traces. The real power traces of all the practical experiments were sampled from PowerSuite 4.0. PowerSuite 4.0 is a software benchmark evaluation board we designed and developed by ourselves, and its CPU is an 8-bit microcontroller STC89C58RD+. The real power traces were acquired with a sampling rate of 50M sample/s from PowerSuite 4.0 board. The average number of real power traces during the sampling process was ten times. The leakage function of our device approximates Hamming Weight leakage function. In all the practical experiments, we chose the interesting points using DPA method [4] and based on the properties introduced in [9].

We will introduce practical experiments about Normalized Template Attack for Case 1 at first. Then, practical experiments about Normalized Template Attack for Case 2 will be shown.

4.2.1 Case 1

We tried to attack the output of the first S-box in the first round of an unprotected AES-128 software implementation over PowerSuite 4.0 as an example. Similarly to the simulated experiments of Normalized Template Attack for Case 1 and Template Attack for Case 1, we used 5000, 7000, and 9000 real power traces to build the 256 templates respectively. The three groups of real power



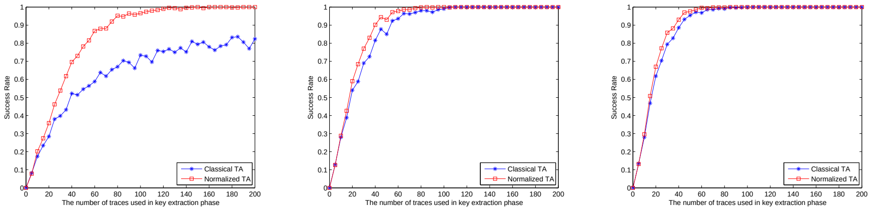
(a) np =20000,σ =2 (b) np =20000,σ =4 (c) np =20000,σ =6

(d) np =40000,σ =2 (e) np =40000,σ =4 (f) np =40000,σ =6

(g) np =60000,σ =2 (h) np =60000,σ =4 (i) np =60000,σ =6

Fig. 3. Simulated Experiments Results of Normalized Template Attack for Case 2 and Template Attack for Case 2 ($SR_{(ne,NTA2)}$,$SR_{(ne,TA2)}$)

traces were generated with a fixed main key and random plaintext inputs. We generated additional 20000 real power traces with another fixed main key and random plaintext inputs. The 20000 real power traces were used in the extraction stage. We also computed $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ similarly to that in the simulated experiments in Section 4.1.1 but with real power traces. We will show $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ for different ne in Figure 4. In Figure 5, we show the success rate of recovering the whole main key of the unprotected AES-128 software implementation over PowerSuite 4.0 for Normalized Template Attack for Case 1 and Template Attack for Case 1 (denoted by $GSR_{(ne,NTA1)}$ and $GSR_{(ne,TA1)}$). For every S-box in the first round of AES-128, Normalized Template Attack for Case 1 is more effective than Template Attack for Case 1. Table 2 shows the ratio of $SR_{(ne,NTA1)}$ to $SR_{(ne,TA1)}$ in the practical experiments for different number of np and ne.

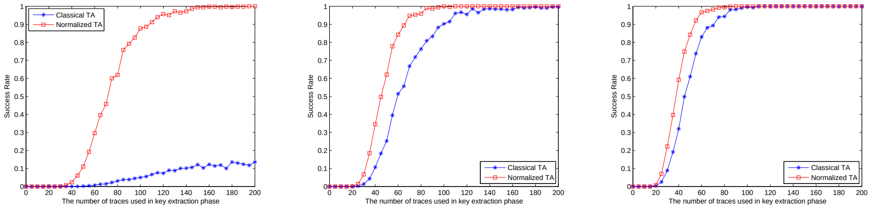


(a) np =5000

(b) np =7000

(c) np =9000

Fig. 4. Practical Experiments Results of Normalized Template Attack for Case 1 and Template Attack for Case 1 ($SR_{(ne,NTA1)}$,$SR_{(ne,TA1)}$)



(a) np =5000

(b) np =7000

(c) np =9000

Fig. 5. Practical Experiments Results of Normalized Template Attack for Case 1 and Template Attack for Case 1 ($GSR_{(ne,NTA1)}$,$GSR_{(ne,TA1)}$)

From Figure 4, Figure 5 and Table 2, we can see that Normalized Template Attack for Case 1 is more effective than Template Attack for Case 1. Similar to the simulated experiments, when we uses more real power traces to build the

Table 2. The ratio of $SR_{(ne,NTA1)}$ to $SR_{(ne,TA1)}$ ($SR_{(ne,NTA1)}/SR_{(ne,TA1)}$) in The Practical Experiments

ne \ np	5000	7000	9000
50	1.43	1.09	1.02
100	1.31	1.01	1
150	1.26	1	1

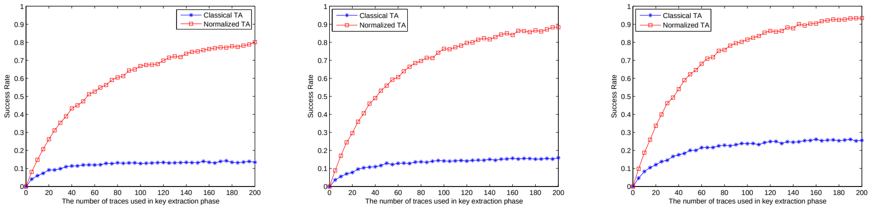
templates in the profiling stage, the success rate of Normalized Template Attack for Case 1 are not lower than that of Template Attack for Case 1. Hence, we only consider the case that one can only use less real power traces to build the templates in the profiling stage.

4.2.2 Case 2

We attacked the same intermediate value as the simulated experiments of Case 2 in the key expansion algorithm of an unprotected AES-128 software implementation over PowerSuite 4.0 as an example.

Similarly to the simulated experiments, we used 80000, 120000, and 160000 real power traces to build the 256 templates respectively. If we use less real power traces in the profiling stage (such as 20000, 40000, and 60000 simulated power traces like the simulated experiments in Section 4.1.2), the success rate of both the two attack methods are low though the success rate of Normalized Template Attack for Case 2 will also much higher than that of Template Attack for Case 2. The reason of this situation is the leakage function of our device approximates Hamming Weight leakage function and it is very difficult to distinguish two different intermediate values (the output of the S-box) which have the same Hamming Weight. Therefore, we used 80000, 120000, and 160000 real power traces to build 256 templates in order to give out a clearer situation. The main key of each of these real power traces was chosen uniformly at random. In our practical experiments, we also chose 32 random main key (Thus there are 32 random values of key[15].). For each main key, we generated 600 real power traces with the fixed main key. Similarly to the simulated experiments, we computed $sr_{(ne,i,NTA2)}$ and $sr_{(ne,i,TA2)}, i = 1, 2, \ldots, 32$ for the 32 random values of key[15]. Then we computed $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$. The value of $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$ are shown in Figure 6. Table 3 shows the ratio of $SR_{(ne,NTA2)}$ to $SR_{(ne,TA2)}$ for different number of np and ne.

From Figure 6 and Table 3, we can see that Normalized Template Attack for Case 2 is more effective than Template Attack for Case 2. The success rate of Normalized Template Attack for Case 2 is close to 1. We attacked all the S-boxes in w[3], w[7], w[11] and w[15] similarly. For each S-box, Normalized Template Attack for Case 2 is more effective than Template Attack for Case 2. Furthermore, the situation of success rate for each S-box are very similar. Due to the fact that $SR_{(ne,NTA2)}$ is much higher than $SR_{(ne,TA2)}$ for all the S-boxes, the success rate of Normalized Template Attack for Case 2 of recovering the



(a) np =80000 (b) np =120000 (c) np =160000

Fig. 6. Practical Experiments Results of Normalized Template Attack for Case 2 and Template Attack for Case 2 ($SR_{(ne,NTA2)}$,$SR_{(ne,TA2)}$)

Table 3. The Ratio of $SR_{(ne,NTA2)}$ to $SR_{(ne,TA2)}$ ($SR_{(ne,NTA2)}/SR_{(ne,TA2)}$) in Practical Experiments

np ne	80000	120000	160000
50	3.97	4.32	3.10
100	5.24	5.39	3.43
150	5.69	5.56	3.51

whole main key is close to 1 and is much higher than that of Template Attack for Case 2 which is close to 0. Therefore, it is not necessary to show the success rate of recovering the whole main key here.

5 Conclusion and Future Work

In this paper, we prove that the leakage exploitation rate of the classical Template Attack is not optimal by introducing a normalization process. The normalization process can be used in both Case 1 and Case 2 yielding Normalized Template Attack. We verified Normalized Template Attack by simulated and practical experiments. Remarkably enough, the normalization process is of extremely low computation cost. Therefore, we argue that this normalization process should be integrated into Template Attack as one necessary step in order to better understand practical threats of this kind of attack. Additionally, we find a quantitative factor in the extraction stage of the classical Template Attack which affects the effectiveness of it.

Although there exit other side channel attacks [26,27,28] against AES with better attack result than Normalized Template Attack, the significance of our work is not to find the best attack against AES. Furthermore, Template Attack can be exploited to attack many cryptographic protocols not only AES.

Our work inspire us to think about the following two questions. First, is the leakage exploitation rate of other profiled side-channel attacks optimal? Second, what are the quantitative factors in the extraction stages of other profiled side-channel attacks which affect the effectiveness of the attacks? Other profiled side-channel attacks include the stochastic model based attack method [24], reduced Template Attack [25], and principal subspace-based Template Attacks [15,16] etc. We believe that the two questions are worth researching. Additionally, in this paper, we use success rate to show leakage exploitation rate. How to quantitatively compute leakage exploitation rate of an attack method is another interesting problem.

References

1. K. Gandol, C. Mourtel, and F. Olivier.: Electromagnetic Analysis: Concrete Results. CHES2001, LNCS 2162, pp.251-261, 2001.
2. Paul C. Kocher.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO1996, LNCS 1109, pp.104-113, 1996.
3. D. Boneh, R.A. DeMillo, and R.J. Lipton.: On the Importance of Checking Cryptographic Protocols for Faults. EUROCRYPT1997, LNCS 1233, pp.37-51, 1997.
4. P. Kocher, J. Jaffe, and B. Jun.: Differential Power Analysis. CRYPTO1999, LNCS 1666, pp.388-397, 1999.
5. D. Boneh, G. Durfee, and Y. Frankel.: An Attack on RSA Given a Fraction of the Private Key Bits. ASIACRYPT1998, LNCS 1514, pp.25-34, 1998.
6. European Network of Excellence (ECRYPT). The side channel cryptanalysis lounge, http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html (retrieved on 29.03.2008)
7. J.-J. Quisquater, F. Koene.: Side channel attacks:State of the art, October 2002.[6].
8. S. Chari, J.R. Rao, and P. Rohatgi.: Template Attacks. CHES2002, LNCS 2523, pp.13-28, 2003.
9. C. Rechberger, E. Oswald.: Practical Template Attacks. WISA2004, LNCS 3325, PP.440-456, 2004.
10. F.-X. Standaert, T.G. Malkin, and M. Yung.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. EUROCRYPT2009, LNCS 5479, pp.443-461, 2009.
11. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel.: Mutual Information Analysis. CHES2008, LNCS 5154, pp.426-442, 2008.
12. S. Mangard, E. Oswald, and T. Popp.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.40-42, Springer (2007).
13. S. Mangard, E. Oswald, and T. Popp.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.156, Springer (2007).
14. B. Gierlichs, K. Lemke-Rust, and C. Paar.: Templates vs. Stochastic Methods A Performance Analysis for Side Channel Cryptanalysis. CHES2006, LNCS4249, pp.15-29, 2006.
15. C. Archambeau, E. Peeters, F.-X. Standaert, and J.-J. Quisquater.:Template Attacks in Principal Subspaces. CHES2006, LNCS 4249, pp.1-14, 2006.
16. F.-X. Standaert, C. Archambeau.: Using Subspace-Based Template Attacks to Compare and Combin Power and Electromagnetic Information Leakages. CHES2008, LNCS 5154, pp.411-425, 2008.
17. E. Oswald, S. Mangard.: Template Attacks on Masking—Resistance Is Futile. CT-RSA2007, LNCS 4377, pp.243-256, 2007.
18. N. Hanley, M. Tunstall, and W.P. Marnane.: Unknown Plaintext Template Attacks. WISA2009, LNCS 5932, pp.148-162, 2009.

19. A. Menezes, P. van Oorschot, and S. Vanstone.: Handbook of Applied Cryptography, CRC Press, Chapter 14, Algorithm 14.85, pp.616, 1996.
20. D.P. Montminy, R.O. Baldwin, M.A. Temple, and E.D. Laspe.: Improving cross-device attacks using zero-mean unit-variance mormalization. Journal of Cryptographic Engineering, Volume 3, Issue 2, pp.99-110, June 2013.
21. P.N. Fahn, P.K. Pearson.: IPA: A New Class of Power Attacks. CHES1999, LNCS 1717, pp.173-186, 1999.
22. T.S. Messerges, E.A. Dabbish, and R.H. Sloan.: Power Analysis Attacks of Modular Exponentiation in Smart Cards. CHES1999, LNCS1717, pp.144-157, 1999.
23. H.L. Zhang, Y.B. Zhou, and D.G. Feng.: An Efficient Leakage Characterization Method for Profiled Power Analysis Attacks. ICISC2011, LNCS 7259, pp.61-73, 2011.
24. W. Schindler, K. Lemke, and C. Paar.: A Stochastic Model for Differential Side Channel Cryptanalysis. CHES2005, LNCS 3659, pp.30-46, 2005.
25. S. Mangard, E. Oswald, and T. Popp.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.108, Springer (2007).
26. S. Mangard.: A Simple Power-Analysis (SPA) Attack on Implementations of The AES Key Expansion, ICISC2002, LNCS 2587, pp.343-358, 2003.
27. M. Renauld, F.-X. Standaert, and N. Veyrat-Charvillon.: Algebraic Side-channel Attacks on The AES: Why Time Also Matters in DPA, CHES2009, LNCS 5747, pp.97-111, 2009.
28. Y. Oren, M. Renauld, F.-X. Standaert, and A. Wool.: Algebraic Side-Channel Attacks Beyond the Hamming Weight Leakage Model, CHES2012, LNCS 7428, pp.140-154, 2012.
29. L. Lerman, G. Bontempi, and O. Markowitch.: Side Channel Attack: An Approach Based On Machine Learning. COSADE2011, pp.29-41, 2011.
30. L. Lerman, S.F. Medeiros, N. Veshchikov, C. Meuter, G. Bontempi, and O. Markowitch.: Semi-Supervised Template Attack. COSADE2013, LNCS 7864, pp.184-199, 2013.
31. F. Durvaux, M. Renauld, F.-X. Standaert, Loic van Oldeneel tot Oldenzeel, Nicolas Veyrat-Charvillon.: Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models. CARDIS2012, LNCS 7771, pp. 123-140, 2013.

Appendix A: The Key Expansion Algorithm of Unprotected AES-128

In this section, we introduce the key expansion algorithm of unprotected AES-128 in Algorithm 3.

Algorithm 3 Key Expansion Algorithm of AES-128

```
KeyExpansion (byte key [16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)
        w[i] = (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
        temp = w[i − 1];
        if (i mod 4 = 0)
            temp = SubWord(RotWord(temp)) ⊕ Rcon[i/4];
        w[i] = w[i − 4] ⊕ temp;
    }
}
```

The AES-128 key expansion algorithm takes as input a 4-word (16-byte) key (main key) and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.