Towards Optimal Leakage Exploitation Rate in Template Attacks

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
`guangjunfan@163.com` , `feng@tca.iscas.ac.cn`
[2] State Key Laboratory of Information Security,
Institute of Information Engineering, Chinese Academy of Sciences
`{zhouyongbin,zhanghailong}@iie.ac.cn`

Abstract. Under the assumption that one has a reference device identical or similar to the target device, and thus be well capable of characterizing power leakages of the target device, Template Attacks are widely accepted to be the most powerful side-channel attacks. However, the question of whether Template Attacks are really optimal in terms of the 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 Attacks. Specifically, our contributions are two folds. On the theoretical side, we prove that Normalized Template Attacks are better in terms of the leakage exploitation rate than Template Attacks; on the practical side, we evaluate the key-recovery efficiency of Normalized Template Attacks and Template Attacks in the same attacking scenario. Evaluation results show that, compared with Template Attacks, Normalized Template Attacks are more effective. We note that, the computational price of the normalization process is of extremely low, and thus it is very easy-to-implement in practice. Therefore, the normalization process should be integrated into Template Attacks as a necessary step, so that one can better understand practical threats of Template Attacks.

Keywords: Side-Channel Attacks, Power Analysis Attacks, Template Attacks, Leakage Exploitation Rate.

1 Introduction

Power Analysis Attacks are the most widely used side-channel attacks. Power Analysis Attacks exploit the fact that the instantaneous power consumption of a device depends on the data it processes and the operations it performs. As an important method of Power Analysis Attacks, Template Attacks were firstly proposed by S. Chari et al. in 2002 [1]. Under the assumption that one has a reference device identical or similar to the target device, and thus be well capable of characterizing power leakages of the target device, Template Attacks

are widely accepted to be the strongest side-channel attacks from an information theoretic point of view [1].

Principally, Template Attacks consist of two stages. The first stage is the profiling stage and the second stage is the extraction stage. In the profiling stage, one can accurately characterize signals and noises in different time samples and builds templates for each key-dependent operation with the reference device. In the extraction stage, one can exploit a small number of power traces measured from the target device and the templates to classify the correct (sub)key. We note that, Template Attacks are important tools to evaluate the physical security of a cryptographic device.

In many real world settings, one can not classify the correct (sub)key with only a single trace in the extraction stage due to noises and accuracy of templates. Therefore, one needs more than one trace to classify the correct (sub)key. According to the attack scenarios, one may apply maximum likelihood approach on the product or the sum of the conditional probabilities to classify the correct (sub)key. Below are the two cases and some specific examples.

Case 1: When the traces are statistically independent, one will apply maximum likelihood approach on the product of conditional probabilities [2]. For convenience, we call classical Template Attacks in this case as "Template Attacks for Case 1". *Example for Case 1*: When one can attack the output of the S-boxes in the first round of AES-128 with random message inputs chosen by himself, he will apply maximum likelihood approach on the product of conditional probabilities. Because the outputs of the S-boxes are random, the traces are statistically independent.

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 [3]. For convenience, we call classical Template Attacks in this case as "Template Attacks for Case 2". *Example for Case 2*: If one tries to recover a fixed secret value, he will obtain a number of traces corresponding to the fixed secret value. Since every symmetric cipher contains some sort of key scheduling mechanism which processes the secret key, this case is likely to exist.

1.1 Motivations

In the extraction stage of Template Attacks, one *directly* applies maximum likelihood approach on the product or the sum of the conditional probabilities to classify the correct (sub)key. However, it is *unknown* that whether this simple way of exploiting traces and templates optimizes the leakage exploitation rate even if templates are accurately built in the profiling stage. Therefore, an important question is whether or not there exists a more powerful way in the extraction stage which has a higher leakage exploitation rate than classical Template Attacks? In this paper, we try to answer this important question.

It is well known that leakage exploitation rate of Template Attacks is mainly affected by noises and accuracy of templates in power traces. However, how noises and accuracy of templates affect leakage exploitation rate of Template

Attacks has not been established. In this paper, we try to find some *quantitative* factors in the extraction stage which reduce leakage exploitation rate of Template Attacks.

1.2 Contributions

The main contributions of this paper are two-fold as follows. Although Template Attacks are widely accepted to be the strongest side-channel attacks from an information theoretic point of view, we first prove that the leakage exploitation rate of classical Template Attacks is not optimal. This observation is obtained by introducing a normalization process into classical Template Attacks yielding Normalized Template Attacks. In the extraction stage, Normalized Template Attacks have higher leakage exploitation rate than classical Template Attacks. Second, we find a quantitative factor which affects the leakage exploitation rate of classical Template Attacks. This quantitative factor gives us a better understanding of Template Attacks in theory.

1.3 Related Work

Template Attacks were firstly introduced in [1]. In [3], C. Rechberger et al. provided answers to some basic and practical issues of Template Attacks, such as how to select interesting points in an efficient way and how to preprocess noisy data. Template Attacks for Case 2 were also presented in [3] (Amplified Template Attacks). PCA-Based Template Attacks were investigated in [4]. However, this kind of Template Attacks is inefficient due to its high computational requirements [3] and may not improve the classification performance [5]. Therefore, PCA-Based Template Attacks are not used widely in practice. In [6], Template Attacks were used to attack a masking protected implementation of a block cipher. In [7], an efficient leakage characterization method was introduced to efficiently characterize power leakages of the target device. Recently, a simple pre-processing technique of Template Attacks, normalizing the sample values using the means and variances was evaluated for various sizes of test data [5]. In [8], the assumption of Template based DPA was relaxed with machine learning techniques. Also, the paper [9] relaxed the assumption made in Template Attacks by using a method based on a semi-supervised learning strategy. However, the important discoveries in this paper are not considered or neglected in the previous papers.

1.4 Organization of This Paper

The rest of this paper is organized as follows. In section 2, we review Template Attacks. In section 3, we introduce Normalized Template Attacks and prove that they have higher higher leakage exploitation rate than classical Template Attacks. Normalized Template Attacks are verified by experiments in section 4. In section 5, we conclude the whole paper.

2 Preliminaries

In this section, we briefly review Template Attacks. We will introduce the two stages of Template Attacks in the following.

2.1 The Profiling Stage

In the profiling stage, one has a reference device identical or similar to the target device. One can use power traces measured from the reference device to characterize power leakages of the target device.

Let us assume that there exist K different (sub)keys $key_i, i = 0, 1, \ldots, K-1$ which need to be classified. Also, there exist K different key-dependent operations O_i, $i = 0, 1, \ldots, K-1$. Usually, one will generate K templates, one for each key-dependent operation O_i. One can exploit advanced techniques [3, 10] to choose N interesting points $(P_1, P_2, \ldots, P_N)$. Each template is composed of a mean vector and a covariance matrix. Specifically, the mean vector is used to estimate the data-dependent portion of side-channel leakages. It is the average signal $M_i = (M_i[P_1], \ldots, M_i[P_N])$ for each one of the key-dependent operations. The covariance matrix is used to estimate the probability density of the noises at different interesting points. It is assumed that noises at different interesting points approximately follow the multivariate normal distribution. A N dimensional noise vector $n_i(S)$ is extracted from each trace $S = (S[P_1], \ldots, S[P_N])$ 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:

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

In equation (1), the symbol $|C_i|$ denotes the determinant of C_i and the symbol C_i^{-1} denotes its inverse.

2.2 The Extraction Stage

In the extraction stage, one tries to classify the correct (sub)key with a small number of traces obtained from the target device. Usually, due to noises and accuracy of templates, one can not recover the correct (sub)key with only one single trace. When one can obtain more than one trace in the extraction stage, one can classify the correct (sub)key by applying the maximum likelihood approach on the product or the sum of conditional probabilities.

Assume one obtains t traces (denoted by $S_1, S_2, \ldots, S_t$) in the extraction stage. When the traces are statistically independent, one will apply maximum likelihood approach on the product of conditional probabilities [2], 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 (sub)key. The output of the function $f(S_j, key_i)$ is the index of a key-dependent operation. For example, when the output of the first S-box (denoted by $Sbox$) in the first round of AES-128 is chosen as the target intermediate value, one builds templates for each 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 corresponding to the power trace S_j.

When the traces are not statistically independent, one may applies maximum likelihood approach on the sum of conditional probabilities [3], i.e.

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

where $Pr(S_j|key_i) = p_{f(key_i)}(n_{f(key_i)}(S_j))$. The key_{ck} is considered to be the correct (sub)key. The output of the function $f(key_i)$ is the index of a key-dependent operation. In this case, the output of $f(key_i)$ only depends on key_i. For example, when the output of a S-box in the key expansion algorithm of AES-128, $f(key_i) = Sbox(key_i)$ is chosen as the target intermediate value.

3 Normalized Template Attacks

In this section, we first introduce the main idea of Normalized Template Attacks. Then, we explain why Normalized Template Attacks have higher leakage exploitation rate compared with Template Attacks.

Main Idea: We introduces a normalization process in the extraction stage and does not change the profiling stage. The normalization process helps improve the leakage exploitation rate of Template Attacks. The reason is that it reduces the effects of noises in each trace and inaccuracy of templates by exploiting normalized conditional probability instead of conditional probability.

Now, we show Normalized Template Attacks for Case 1. We call this method as "Normalized Template Attacks for Case 1", which is summarized as Algorithm 1. The profiling stage of Normalized Template Attacks for Case 1 is the same as classical Template Attacks. Therefore, we ignore the profiling stage here and only show the extraction stage. For simplicity, we rewrite $Pr(S_j|key_i)$ as $P(i,j), i = 0, 1, \ldots, K - 1, j = 1, 2, \ldots, t$.

Step 1-Step 3 in Algorithm 1 are called the normalization process. We note that, the computational cost of the normalization process is extremely low. In the following, we explain why Normalized Template Attacks have a higher leakage exploitation rate compared with classical Template Attacks.

For each trace S_j $(j = 1, 2, \ldots, t)$, we compute $maxln(j)$. 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 trace S_j. Let

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

5

Algorithm 1 Normalized Template Attacks 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:

$$V(i, j) = exp\left(\frac{maxln(j)}{H(i, j)}\right), i = 0, 1, \ldots, K - 1, j = 1, 2, \ldots, t.$$

Step 4 Applying maximum likelihood approach on $\prod_{j=1}^{t} V(i, j)$. Let

$$key_{ck} = argmax_i\left\{\prod_{j=1}^{t} V(i, j), i = 0, 1, \ldots, K - 1\right\}.$$

Step 5 Return key_{ck}.

We call $V(i, j)$ the normalized conditional probability.

We note that the probability that the following two inequalities (inequality (3) and inequality (4)) happen simultaneously is extremely low. The reason is that for the correct key key_{ck} and most traces S_j ($j = 1, 2, \ldots, t$), the value $\alpha_{(ck,j)}$ is much closer to 1 than $\alpha_{(i,j)}$ for all the wrong keys $key_i, \forall i \in \{0, 1, \ldots, K - 1\}\backslash ck$ with high probability.

$$\prod_{j=1}^{t} V(ck, j) \leq \prod_{j=1}^{t} V(i, j), \exists i \in \{0, 1, \ldots, K - 1\}\backslash ck. \tag{3}$$

$$\prod_{j=1}^{t} P(ck, j) > \prod_{j=1}^{t} P(i, j), \forall i \in \{0, 1, \ldots, K - 1\}\backslash ck. \tag{4}$$

Now, we assume for the correct key key_{ck}, it holds that

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

If

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

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

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

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 $P1_{ck} > P1_{wk}$, $P2_{ck} \leq P2_{wk}$, $V1_{ck} > V1_{wk}$, and $V2_{ck} \leq V2_{wk}$. 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 further have

$$\frac{V1_{ck}}{V1_{wk}} > \frac{V2_{wk}}{V2_{ck}}. \tag{5}$$

However, if classical Template Attacks (Template Attacks for Case 1) return key_{wk} as the output, it holds that

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

Therefore, we have

$$\frac{P1_{ck}}{P1_{wk}} < \frac{P2_{wk}}{P2_{ck}}. \tag{6}$$

We know that the value of $maxln(j)$ falls in the interval $(-\infty, 0)$. The value of $maxln(j)$ is different for different traces even with the same operation and data due to noises in each trace and accuracy of templates. For different traces with the same operation and different data, the above fact still holds. For example, we try to attack the output of the first S-box in the first round of AES-128 in simulated attacking scenario (The signals are assumed to follow the Hamming Weight leakage model. The standard deviation of the Gauss noises added into the simulated power traces is 4.). Table 1 in Appendix A shows the variance of $maxln(j)$ of 100 simulated power traces with the same operation and data for different number of simulated power traces used in the profiling stage. Table 1 shows that, the variance of $maxln(j)$ is not small and reduces with the increase of the number of simulated power traces used in the profiling stage.

Now, we assume $V(ck,S)$ 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. The value $maxln(S)$ is close to 0.), the value of $exp(H(ck,S)) - exp(H(wk,S))$ is large. When $maxln(S)$ is small (i.e. The value $maxln(S)$ is far from 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 induces inequality (5) and inequality (6) hold simultaneously and will reduce leakage exploitation rate of classical Template Attacks when one directly computes conditional probabilities and applies maximum likelihood approach on the produce or the sum of conditional probabilities in the extraction stage.
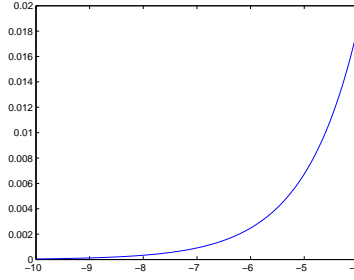
Fig. 1. The Function $f(x) = e^x$ for $x \in [-10, -4]$

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

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

$$maxln(S_2) = -6, H(wk,2) = -6, H(ck,2) = -9.23.$$

Normalized Template Attacks for Case 1 will return the correct key ck because

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

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

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

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

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

To sum up, Normalized Template Attacks exploit normalized conditional probabilities which are more effective and accurate than traditional condition-al probabilities of classical Template Attacks. Therefore, we expect Normalized

Template Attacks to have a higher leakage exploitation rate than classical Template Attacks.

We know that $V(i,j) \in [e^0, e]$ and $P(i,j) = e^{H(i,j)} = V(i,j)^{H^2(i,j)/maxln(j)}$, $i = 0, 1, \ldots, K-1, j = 1, 2, \ldots, t$. Consequently, the value $H^2(i,j)/maxln(j)$ will reduce leakage exploitation rate of classical Template Attacks.

Normalized Template Attacks for Case 2 are identical to Normalized Template Attacks for Case 1 except that they apply maximum likelihood approach on $\sum_{j=1}^{t} V(i,j)$ to determine the correct key, namely

$$key_{ck} = argmax_i \left\{ \sum_{j=1}^{t} V(i,j), i = 0, 1, \ldots, K-1 \right\}$$

in Step 4.

4 Experiments

An attack method with a higher leakage exploitation rate will has a higher success rate [11]. Therefore, in this section, we will experimentally evaluate the success rates of Normalized Template Attacks and classical Template Attacks for both Case 1 and Case 2.

For implementation of a cryptographic algorithm with countermeasures, one usually first uses some methods to delete the countermeasures and then tries to attack the implementation using classical attacks against unprotected implementation. For example, if one has traces with random delays [16]. He may first use the method proposed in [12] to remove the random delays and then uses classical attacks to recover the correct (sub)key. The methods of deleting countermeasures are beyond the scope of this paper. Therefore, we take unprotected AES-128 implementation as example. We tried to attack the output of the first S-box in the first round of an unprotected AES-128 software implementation on an typical 8-bit microcontroller STC89C58RD+ (This 8-bit microcontroller was also exploited by other papers [7,17].). The real power traces were acquired with a sampling rate of 50MS/s. The average number of real power traces during the sampling process was 10 times. The leakage function of our device approximates the Hamming Weight leakage function. In all the practical experiments, we chose the interesting points using CPA based method [14] with the properties introduced in [3].

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 numbers of traces used in the profiling stage means that the templates have different level of accuracy. Different numbers of traces used in the extraction stage represents different amounts of information can be exploited. In each figure, Normalized Template Attacks (such as Normalized Template Attacks for Case 1 or Normalized Template Attacks for Case 2) are denoted by "Normalized TA" and classical Template Attacks (such as Template Attacks for Case 1 or Template Attacks for Case 2) are denoted by "Classical TA". We will introduce practical

experiments about Case 1 at first. Then, practical experiments about Case 2 will be shown. In fact, we also executed simulated experiments and the observations obtained from simulated experiments are similar to those of practical experiments.

4.1 Case 1

We tried to attack the output of the first S-box in the first round of unprotected AES-128 software implementation over PowerSuite 4.0 as an example. We used 5,000, 6,000, and 7,000 real power traces to build the 256 templates respectively. The three groups of real power traces were generated with a fixed main key and random plaintext inputs. We generated additional 100,000 real power traces with another fixed main key and random plaintext inputs. The 100,000 real power traces were used in the extraction stage. We tested the success rate of Normalized Template Attacks for Case 1 (denoted by $SR_{(ne,NTA1)}$) and the success rate of Template Attacks 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 100,000 real power traces uniformly at random. Both Normalized Template Attacks for Case 1 and Template Attacks for Case 1 used the same templates and the same ne traces in the extraction stages. We respectively recorded how many times the two attacks can successfully recover the correct subkey (denoted by $num_{(ne,NTA1)}$ for Normalized Template Attacks for Case 1 and $num_{(ne,TA1)}$ for Template Attacks 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. For every S-box in the first round of AES-128, Normalized Template Attacks for Case 1 are more effective than Template Attacks for Case 1. Table 2 in Appendix A shows the success rate of Normalized Template Attacks for Case 1 and the success rate of Template Attacks for Case 1 in the practical experiments for different np and ne. In each entry "A/B" of Table 2, "A" represents $SR_{(ne,NTA1)}$ and "B" represents $SR_{(ne,TA1)}$.

(a) $np =5,000$ (b) $np =6,000$ (c) $np =7,000$
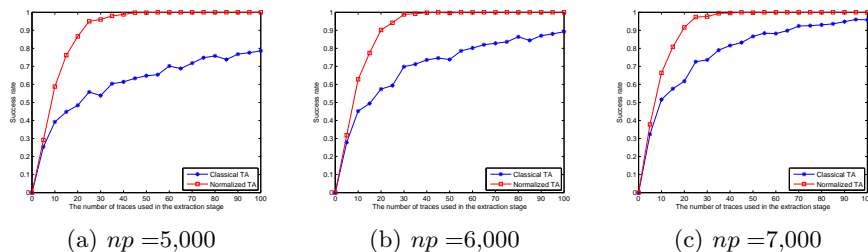
Fig. 2. Practical Experiments Results of Normalized Template Attacks for Case 1 and Template Attacks for Case 1 ($SR_{(ne,NTA1)}$,$SR_{(ne,TA1)}$)

It can be seen from Figure 2 and Table 2 that, Normalized Template Attacks for Case 1 have obviously higher success rate (higher leakage exploitation rate) than Template Attacks for Case 1. When we use more real power traces to build the templates in the profiling stage and use more real power traces in the extraction stage, the success rate of Normalized Template Attacks for Case 1 is not lower than the success rate of Template Attacks for Case 1. Hence, we only show the cases that one uses less real power traces here.

4.2 Case 2

To verify Case 2, we attacked the key expansion algorithm of unprotected AES-128 software implementation over PowerSuite 4.0 as an example. Algorithm 2 in Appendix B describes the key expansion algorithm of unprotected AES-128. RotWord in Algorithm 2 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 2 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 mathematical 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 outputs of every S-box in SubWord successfully. In all of our practical experiments for Normalized Template Attacks for Case 2 and Template Attacks 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 used 80,000, 120,000, and 160,000 real power traces to build the 256 templates respectively. If we use less real power traces in the profiling stage, the success rates of both the two attacks are low though the success rate of Normalized Template Attacks for Case 2 will also be much higher than the success rate of Template Attacks 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 80,000, 120,000, and 160,000 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 chose 32 random main key (Thus there were 32 random values of key[15].). For each main key, we generated 600 real power traces with the fixed main key. We tested the success rate of Normalized Template Attacks for Case 2 (denoted by $SR_{(ne,NTA2)}$) and the success rate of Template Attacks for Case 2 (denoted by $SR_{(ne,TA2)}$) when one can use ne traces in 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 real power traces uniformly at random from the corresponding 600 simulated power traces. Both Normalized Template Attacks for Case 2 and Template Attacks

11

for Case 2 used the same templates and the same ne traces in the extraction stages. 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 Attacks for Case 2 and $num_{(ne,i,TA2)}$ for Template Attacks for Case 2). For the ith $(i = 1, 2, \ldots, 32)$ value of key[15], we use $sr_{(ne,i,NTA2)}$ $(sr_{(ne,i,NTA2)} = num_{(ne,i,NTA2)}/128)$ and $sr_{(ne,i,TA2)}$ $(sr_{(ne,i,TA2)} = num_{(ne,i,TA2)}/128)$ to denote the success rate. The success rates 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. Table 3 in Appendix A shows the success rates of Normalized Template Attacks for Case 2 and Template Attacks for Case 2 in the practical experiments for different np and ne. In each entry "A/B" of Table 3, "A" represents $SR_{(ne,NTA2)}$ and "B" represents $SR_{(ne,TA2)}$.

(a) $np =$80,000   (b) $np =$120,000   (c) $np =$160,000
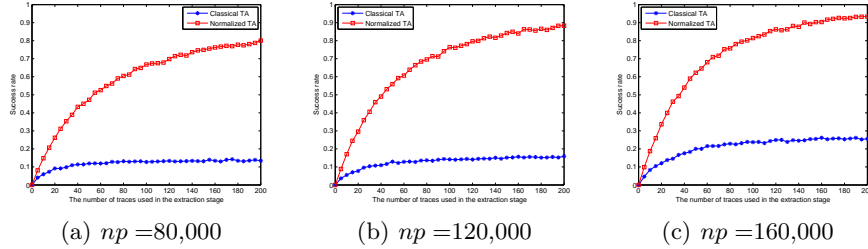
Fig. 3. Practical Experiments Results of Normalized Template Attacks for Case 2 and Template Attacks for Case 2 $(SR_{(ne,NTA2)}, SR_{(ne,TA2)})$

It can be seen from Table 3 that, Normalized Template Attacks for Case 2 have much higher success rate (higher leakage exploitation rate) than Template Attacks for Case 2. The success rate of Normalized Template Attacks 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 Attacks for Case 2 have much higher success rate than Template Attacks for Case 2. Moreover, the situations of the success rate for each S-box are very similar. Therefore, the success rate of Normalized Template Attacks for Case 2 of recovering the whole main key is close to 1 and is much higher than the success rate of Template Attacks for Case 2 which is close to 0.

5 Conclusion and Future Work

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

Our work inspire us to think about the following two questions. First, whether the leakage exploitation rates of other profiled side-channel attacks (especially in the extraction stages) is optimal? Second, what are the quantitative factors in the extraction stages of other profiled side-channel attacks affect their effectiveness? Other profiled side-channel attacks include the stochastic model based attacks [13], reduced Template Attacks [14], PCA-Based Template Attacks [4], and LDA-Based Template Attacks [15] etc. We believe that the two questions are worth researching. It is also very necessary to further verify Normalized Template Attacks in other devices such as FPGA and ASIC. Moreover, it would be interesting to have a formula that expresses leakage exploitation rate of a side-channel attack in terms of the number of traces in future work.

References

[1]   Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. CHES2002, LNCS 2523, pp.13-28, 2003.
[2]   Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.156, Springer (2007).
[3]   Rechberger, C., Oswald, E.: Practical Template Attacks. WISA2004, LNCS 3325, pp.440-456, 2004.
[4]   Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.:Template Attacks in Principal Subspaces. CHES2006, LNCS 4249, pp.1-14, 2006.
[5]   Montminy, D.P., Baldwin, R.O., Temple, M.A., Laspe, E.D.: Improving cross-device attacks using zero-mean unit-variance mormalization. Journal of Cryptographic Engineering, Volume 3, Issue 2, pp.99-110, June 2013.
[6]   Oswald, E., Mangard, S.: Template Attacks on Masking—Resistance Is Futile. CT-RSA2007, LNCS 4377, pp.243-256, 2007.
[7]   Zhang, H., Zhou, Y., Feng, D.: An Efficient Leakage Characterization Method for Profiled Power Analysis Attacks. ICISC2011, LNCS 7259, pp.61-73, 2011.
[8]   Lerman, L., Bontempi, G., Markowitch, O.: Side Channel Attack: An Approach Based On Machine Learning. COSADE2011, pp.29-41, 2011.
[9]   Lerman, L. Medeiros, S.F., Veshchikov, N., Meuter, C., Bontempi, G., Markowitch, O.: Semi-Supervised Template Attack. COSADE2013, LNCS 7864, pp.184-199, 2013.
[10] Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods A Performance Analysis for Side Channel Cryptanalysis. CHES2006, LNCS4249, pp.15-29, 2006.

[11] Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. EUROCRYPT2009, LNCS 5479, pp.443-461, 2009.

[12] Durvaux, F., Renauld, M., Standaert, F.-X. et al.: Efficient Removal of Random Delays from Embedded Software Implementations Using Hidden Markov Models. CARDIS2012, LNCS 7771, pp. 123-140, 2013.

[13] Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. CHES2005, LNCS 3659, pp.30-46, 2005.

[14] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Chapter 6, Springer (2007).

[15] Standaert, F.-X., Archambeau, C.: Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. CHES2008, LNCS 5154, pp.411-425, 2008.

[16] Coron, J.-S., Kizhvatov, I.: Analysis and Improvement of the Random Delay Countermeasure of CHES 2009. CHES2010, LNCS 6225, pp.95–109, 2010.

[17] Feng, M., Zhou, Y., Yu, Z.: EMD-Based Denoising for Side-Channel Attacks and Relationships between the Noises Extracted with Diffierent Denoising Methods. ICICS2013, LNCS 8233, pp.259-274, 2013.

Appendix A: The Tables

In all the tables, the number of traces used to build the templates is denoted by np and the number of traces used in the extraction stage is denoted by ne.

Table 1. The Variance of $maxln(j)$

np	5,000	6,000	7,000
variance	2.7356	2.5252	2.1740

Table 2. The Success Rates $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ in Practical Experiments

ne \\ np	5,000	6,000	7,000
25	0.95/0.56	0.94/0.59	0.97/0.73
50	1.00/0.65	1.00/0.74	1.00/0.87
75	1.00/0.75	1.00/0.84	1.00/0.93

Table 3. The Success Rates $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$ in Practical Experiments

ne \\ np	80,000	120,000	160,000
50	0.47/0.12	0.56/0.13	0.62/0.20
100	0.67/0.13	0.76/0.14	0.81/0.24
150	0.75/0.13	0.84/0.15	0.89/0.25

15

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

In this section, we introduce the key expansion algorithm of unprotected AES-128 in Algorithm 2.

Algorithm 2 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.