Towards Optimal Leakage Exploitation Rate 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
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 Attack is widely accepted to be one of the most powerful side-channel attacks. However, the question of whether Template Attack is 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 Attack. Specifically, our contributions are two folds. On the theoretical side, we prove that Normalized Template Attack is better in terms of the leakage exploitation rate than Template Attack; on the practical side, we evaluate the key-recovery efficiency of Normalized Template Attack and Template Attack in the same attacking scenario. Evaluation results show that, compared with Template Attack, Normalized Template Attack is 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 Attack as a necessary step, so that one can better understand practical threats of Template Attack.

Keywords: Side-Channel Attacks, Power Analysis Attack, Template Attack, 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 Power Analysis Attack method, Template Attack was 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 Attack is widely accepted to be the strongest side-channel attack from an information theoretic

point of view [1]. While such an assumption is a bit of strong, it holds in many cases and has been used in other side-channel attacks [2, 3].

Principally, Template Attack is a two-stage attack method. 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 Attack is an important tool 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 [4]. For convenience, we call 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-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 [5]. For convenience, we call classical Template Attack in this case as "Template Attack 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 or asymmetric 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 Attack, 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 that of classical Template Attack? In this paper, we try to answer this important question.

It is well known thatleakage exploitation rate of Template Attack 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

Attack 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 Attack.

1.2 Contributions

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

1.3 Related Work

Template Attack was firstly introduced in [1]. In [5], 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 [5]. Principal subspace-based Template Attacks were investigated in [6, 7]. However, due to their high computational requirements, principal subspace-based Template Attacks are not used widely [5]. In [8], Template Attack was used to attack a masking protected implementation of a block cipher. In [9], an efficient leakage characterization method was introduced to efficiently characterize power leakages of the target device. Recently, a simple pre-processing technique of Template Attack, normalizing the sample values using the means and variances was evaluated for various sizes of test data [10]. In [11], the assumption of Template based DPA was relaxed with machine learning techniques. Also, the paper [12] relaxed the assumption made in Template Attack 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 works.

1.4 Organization of This Paper

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

2 Preliminaries

In this section, we briefly review Template Attack. We will introduce the two stages of Template Attack 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 [5, 13] 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:

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

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

2.2 The Extraction Stage

In the extraction stage, one tries to classify the correct (sub)key with one or a limited 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 [4], 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 [5], 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 $Pr(S_j|key_i) = p_{f(key_i)}(n_{f(key_i)}(S_j))$. 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 Attack

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

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 Attack. 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 Attack for Case 1. We call this method as "Normalized Template Attack for Case 1", which is summarized as Algorithm 1. The profiling stage of Normalized Template Attack for Case 1 is the same as classical Template Attack. 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 Attack has a higher leakage exploitation rate compared with classical Template Attack.

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

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

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}.

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 every trace 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.

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

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

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 Attack for Case 1 will return the correct key key_{ck}. However, when noises are large and/or templates are not very accurate, classical Template Attack may return a wrong key $key_{wk}, wk \in \{0, 1, \ldots, K-1\}\backslash ck$. We will show the reasons why classical Template Attack returns 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 $\mathsf{Set1}$, 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 $\mathsf{Set2}$, 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 Attack (Template Attack for Case 1) returns 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 of function $f(x) = e^x$ induces inequality (5) and inequality (6) hold simultaneously and will reduce leakage exploitation rate of classical Template

Attack when one directly computes conditional probabilities and applies maximum likelihood approach on the produce or the sum of conditional probabilities in the extraction stage. 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

$$maxln(S_1) = -4, H(ck, 1) = -4, H(wk, 1) = -6.4,$$
$$maxln(S_2) = -6, H(ck, 1) = -6, H(wk, 1) = -9.23.$$

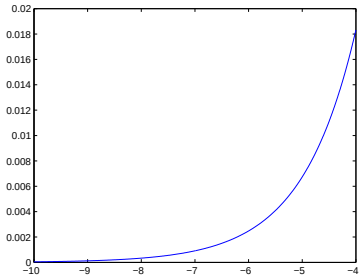


Fig. 1. The Function $f(x) = e^x$ for $x \in [-10, -4]$

Normalized Template Attack for Case 1 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, classical Template Attack will return wk as the answer because

$$exp(H(ck, 1))exp(H(ck, 2)) = exp(-4)exp(-9.23)$$
$$< exp(H(wk, 1))exp(H(wk, 2)) = exp(-6.4)exp(-6).$$

To sum up, Normalized Template Attack exploits normalized conditional probability which is more effective and accurate than traditional conditional probability of classical Template Attack. Therefore, we expect Normalized Template Attack to have a higher leakage exploitation rate than Template Attack.

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 Attack.

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

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

in Step 4.

4 Experiments

An attack method with a higher leakage exploitation rate will have a higher success rate [14]. Therefore, in this section, we will experimentally evaluate the success rate of Normalized Template Attack and classical Template Attack for both Case 1 and Case 2.

For implementations of cryptographic algorithms with countermeasures, one usually first use some method to delete the countermeasures and then tries to attack the implementation using classical attacks (Such as Template Attack). For example, if one has traces with random delays. He may first use the method proposed in [15] to remove the random delays and then use classical attack to recover the correct key. Therefore, we take unprotected AES-128 implementation as example for Case 1 and Case 2 of Template Attack. The real power traces of all the practical experiments were sampled from PowerSuite 4.0. PowerSuite 4.0 is a software benchmark evaluation board designed and developed by ourselves. The CPU of PowerSuite 4.0 is an 8-bit microcontroller STC89C58RD+, which is typical and is used widely in practice. 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 the Hamming Weight leakage function. In all the practical experiments, we chose the interesting points using the classic DPA method [16] with the properties introduced in [5].

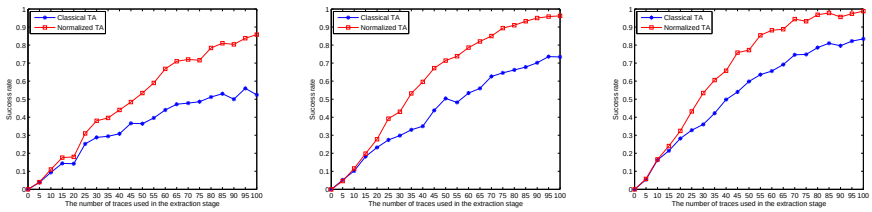
For simplicity, let np denote the number of traces used in the profiling stage and let ne denote the number of traces used in the extraction stage. Different numbers of traces used in the profiling stage means that the templates have different level of accuracy. Different numbers of traces used in the extraction stage represents different amounts of information can be exploited. In each figure, Normalized Template Attack (such as Normalized Template Attack for Case 1 or Normalized Template Attack for Case 2) is denoted by "Normalized TA" and classical Template Attack (such as Template Attack for Case 1 or Template Attack for Case 2) is 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 5000, 6000, and 7000 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 30000 real power traces with another fixed main key and random plaintext inputs. The 30000 real power traces were used in the extraction stage. We tested the success rate of Normalized Template Attack for Case 1 (denoted by $SR_{(ne,NTA1)}$) and the success rate of

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 30000 real 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 subkey (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. 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 in Appendix A shows the success rate of Normalized Template Attack for Case 1 and Template Attack 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 =5000 (b) np =6000 (c) np =7000

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

It can be seen from Figure 2 and Table 2 that, Normalized Template Attack for Case 1 has obviously higher success rate (higher leakage exploitation rate) than Template Attack 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 Attack for Case 1 is not lower than that of Template Attack for Case 1. Hence, we only show the case 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

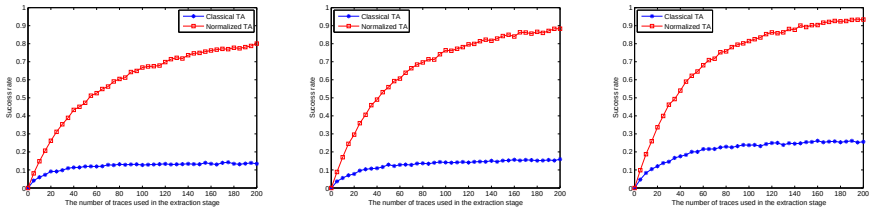
means that an input word [b0,b1,b2,b3] is transformed into [b1,b2,b3,b0]. SubWord in Algorithm 2 performs a byte substitution on each byte of its input word, using the S-box. If the adversary can recover w[3],w[7],w[11], and w[15], then he can recover the main key key[0],key[1],...,key[15] using the 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 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 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, the success rate of both the two attacks 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 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 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 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 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 $sr_{(ne,i,NTA2)}$ ($sr_{(ne,i,NTA2)} = num_{(ne,i,NTA2)}/128$) and $sr_{(ne,i,TA2)}$ ($sr_{(ne,i,TA2)} = num_{(ne,i,TA2)}/128$) to denote the success rate. The success 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. Table 3 in Appendix A shows the success rate of Normalized Template Attack for Case

2 and Template Attack 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 =80000 (b) np =120000 (c) np =160000

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

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

Although there exist other side-channel attacks [17–19] against AES-128 with better attack results than Normalized Template Attack, the significance of our work is not to find the best attack against AES-128. Furthermore, Template Attack can be exploited to attack many cryptographic protocols not only AES. Therefore, our work is of great significance.

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 attack [20], reduced Template Attack [21], and principal subspace-based Template Attacks [6, 7] etc. We believe that the two questions are worth researching. It is also very necessary to further verify Normalized Template Attack and classical Template Attack for 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] Fahn, P.N., Pearson, P.K.: IPA: A New Class of Power Attacks. CHES1999, LNCS 1717, pp.173-186, 1999.

[3] Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Power Analysis Attacks of Modular Exponentiation in Smart Cards. CHES1999, LNCS1717, pp.144-157, 1999.

[4] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.156, Springer (2007).

[5] Rechberger, C., Oswald, E.: Practical Template Attacks. WISA2004, LNCS 3325, PP.440-456, 2004.

[6] Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.:Template Attacks in Principal Subspaces. CHES2006, LNCS 4249, pp.1-14, 2006.

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

[8] Oswald, E., Mangard, S.: Template Attacks on Masking—Resistance Is Futile. CT-RSA2007, LNCS 4377, pp.243-256, 2007.

[9] Zhang, H., Zhou, Y., Feng, D.: An Efficient Leakage Characterization Method for Profiled Power Analysis Attacks. ICISC2011, LNCS 7259, pp.61-73, 2011.

[10] 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.

[11] Lerman, L., Bontempi, G., Markowitch, O.: Side Channel Attack: An Approach Based On Machine Learning. COSADE2011, pp.29-41, 2011.

[12] Lerman, L. Medeiros, S.F., Veshchikov, N., Meuter, C., Bontempi, G., Markowitch, O.: Semi-Supervised Template Attack. COSADE2013, LNCS 7864, pp.184-199, 2013.

[13] Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods A Performance Analysis for Side Channel Cryptanalysis. CHES2006, LNCS4249, pp.15-29, 2006.

[14] 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.

[15] 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.

[16] Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. CRYPTO1999, LNCS 1666, pp.388-397, 1999.
[17] Mangard, S.: A Simple Power-Analysis (SPA) Attack on Implementations of The AES Key Expansion, ICISC2002, LNCS 2587, pp.343-358, 2003.
[18] Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic Side-Channel Attacks on The AES: Why Time Also Matters in DPA, CHES2009, LNCS 5747, pp.97-111, 2009.
[19] Oren, Y., Renauld, M., Standaert, F.-X., Wool, A.: Algebraic Side-Channel Attacks Beyond the Hamming Weight Leakage Model, CHES2012, LNCS 7428, pp.140-154, 2012.
[20] Schindler, W., Lemke, K., Paar, C.: A Stochastic Model for Differential Side Channel Cryptanalysis. CHES2005, LNCS 3659, pp.30-46, 2005.
[21] Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. pp.108, Springer (2007).

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	5000	6000	7000
variance	2.7356	2.5252	2.1740

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

ne \ np	5000	6000	7000
25	0.31/0.25	0.39/0.27	0.43/0.32
50	0.53/0.36	0.71/0.50	0.77/0.59
75	0.72/0.48	0.89/0.64	0.93/0.74

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

ne \ np	80000	120000	160000
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

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.