Towards Leakage Exploitation Optimality in Profiled Side-Channel 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,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 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 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 end effective. Interestingly enough, this normalization is of extremely low computation cost, and thus is very easy-to-use in practice. Therefore, we argue that this normalization should be integrated into profiled attacks as one necessary step in the future, so that one could better understand the practical threats of them.
Keywords: Template Attack, leakage exploitation rate, normalization process, profiled side-channel attacks.

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 on the operation it performs. As an important attack method in Power Analysis Attack, S.Chari et al. presented Template Attack in [8]. Template Attack is widely accepted to be the strongest side channel attack possible 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 limiting, 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 power 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 a device.

In many real world settings, one can not classify the correct key with only a single power trace in the extraction stage due to noise and the accuracy of templates. Therefore, one needs more than one power 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 power traces and the templates in the extraction stage to classify the correct key. Let's show the two cases in the following.

Case 1: When the power 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".

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

Let's show some specific examples for the two cases respectively in the following.

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

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 obtains more than one power traces corresponding to the same key (key-dependent operation) in the extraction stage. Usually, this situation may be caused by the following two reasons. Reason one is that the adversary can not control 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 power trace due to noise 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 power traces with the same key-dependent operation and apply 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 in every invocation. When sliding-window exponentiation [19] is used to compute g^x, one could build templates for the window and applies Template Attack for Case 2.

Motivations An important method to improve the effectiveness of a side-channel attack method is to maximize leakage exploitation rate of the attack method. Usually, a side-channel attack method with higher leakage exploitation rate will be more powerful (In most cases, the more powerful attack method has higher Success Rate [1] [10] than the classical attack method under identical scenarios). In the above two attack scenarios of Template Attack (Case 1 and Case 2), the adversary applies maximum likelihood approach on the product or the sum of conditional probabilities obtained from power traces and templates to classify the correct key in the extraction stage. However, it is unknown that whether this way of exploiting power traces and templates maximize leakage exploitation rate and result in the highest Success Rate. Therefore, a natural question is that does there exist a better way to exploit power traces and templates which has higher leakage exploitation rate in the extraction stage and achieves higher Success Rate of attack than the classical way in Template Attack? In this paper, we try to answer this important question.

Contributions In this paper, we present a new way of exploiting power traces and templates. This new way has higher leakage exploitation rate and achieves higher Success Rate than the classical way. Moreover, the new way can be used in both Case 1 and Case 2. The new way introduces a normalization process into the classical Template Attack and exploits normalized conditional probability (See section 3 for more details) instead of the classical conditional probability. Our experiments verify that the new way has higher leakage exploitation rate because the Success Rate of both Case 1 and Case 2 will be improved a lot when the new way is used.

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,

[1] Assume that the adversary executes *tot* times attack in the same attack scenario. In the *tot* times attack, the adversary successes *suc* times (For example, the secret key is recovered successfully *suc* times.). Then Success Rate is defined to be suc/tot.

such as how to select points of interest in an efficient way, and how to preprocess noisy data. Template Attack for Case 2 was also presented in [9]. Subspace-based template attacks were investigated in [15,16]. In [17], Template Attack on an implementation of a block cipher that uses a masking scheme is introduced that retrieves the secret key. In [23], an efficient leakage characterization method in the profiling stage for Template Attack is introduced. A simple pre-processing technique of Template Attack, normalizing the trace means and variances from the training and targeted devices is evaluated for various test data set sizes in [20]. However, our important discovery is not considered or neglected in these previous work.

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 new attack methods. The two new 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 device which is identical or similar to the targeted device. One derives some power traces from this device. These power traces are used to build templates for each key-dependent operation. In the extraction stage, one uses these templates to classify the correct key.

Let us assume there exist K different key $key_i, i = 1, 2, \ldots, K$ which need to be classified. There also exist K different key-dependent operations O_i with $i = 1, 2, \ldots, K$. 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 leakage. It is the average signal M_i for each of the operations. The second part in a template estimates the probability density of the noise in the side channel leakage. 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 noise in the side channel leakage approximately has 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 power 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 noise occurring under key-dependent operation O_i is then 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 power traces obtained from the targeted device. Usually, due to noise and the accuracy of templates, one can not recover the correct key with only a single power trace. When one obtains more than one power traces 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 power traces and templates.

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

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

$$key_{ck} = argmax_{key_i} \Big\{ \prod_{j=1}^{t} Pr(S_j | key_i), i = 1, 2, \ldots, K \Big\},$$

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 of 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 power traces are not statistically independent, in many real world settings, one can only obtains more than one power 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 $key_i, i = 1, 2, \ldots, K$ and apply maximum likelihood approach on the sum of conditional probabilities, i.e.

$$key_{ck} = argmax_{key_i} \Big\{ \sum_{j=1}^{t} Pr(S_j | key_i), i = 1, 2, \ldots, K \Big\},$$

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 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, $f(key_i) = key_i$.

3 Our New Way to Increase Leakage Exploitation Rate for Template Attack

In this section, we first introduce our new way which has higher leakage exploitation rate (with Case 1 as the example) and explain why the new way has the higher leakage exploitation rate. Then, we introduce the Normalized Template Attacks for Case 1 and Case 2 in which the new way is used formally. Note that Our new way is different from the classical Template Attack only in the extraction stage. The profiling stage remains unchanged. Therefore, we only introduce the extraction stage in this section.

For simplicity, we rewrite $Pr(S_j|key_i)$ as $P(i,j), i = 1,2,\ldots,K, j = 1,2,\ldots,t$. Denote the natural logarithm of each conditional probability $P(i,j)$ as $H(i,j)$, i.e. $H(i,j) = lnP(i,j), i = 1,2,\ldots,K, j = 1,2,\ldots,t$. For every power trace S_j $(j = 1,2,\ldots,t)$, we compute $maxln(j) = max\{H(1,j), H(2,j), \ldots, H(K,j)\}$. Clearly, for each conditional probability $P(i,j), i = 1,2,\ldots,K$ obtained from a single power trace S_j $(j = 1,2,\ldots,t)$, there exists a real number $\alpha_{ij} \in [0,1], i = 1,2,\ldots,K$ such that $\alpha_{ij} = maxln(j)/H(i,j)$. Note that α_{ij} is proportional to the conditional probability $P(i,j)$ for a fixed j. Let

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

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

In the following, we will show a new way with higher leakage exploitation rate than the way in the classical Template Attack. We consider Case 1 as an example. For Case 2, we have similar result.

Assume that, for the correct key key_{ck}, we have

$$\prod_{j=1}^{t} V(ck,j) > \prod_{j=1}^{t} V(i,j), i \in \{1,2,\ldots,K\}\backslash ck.$$

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

If

$$\prod_{j=1}^{t} P(ck,j) > \prod_{j=1}^{t} P(i,j), i \in \{1,2,\ldots,K\}\backslash ck,$$

the classical Template Attack will return the correct key key_{ck}. However, when noise is large and/or the templates are not very accurate, the classical Template Attack may return a wrong key $key_{wk}, wk \in \{1,2,\ldots,K\}\backslash ck$.

We divide the t power traces $\{S_1, S_2, \ldots, S_t\}$ into two sets. In the first set $Set1$, there are u samples $\{S_{i_1}, \ldots, S_{i_u}\}$ and $P(ck,i_1) > P(wk,i_1), \ldots, P(ck,i_u) > P(wk,i_u)$. In the second set $Set2$, there are $t-u$ samples $\{S_{j_1}, \ldots, S_{j_{t-u}}\}$ and $P(ck,j_1) \le P(wk,j_1), \ldots, P(ck,j_{t-u}) \le P(wk,j_{t-u})$. 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 return 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

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

Clearly, we have

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

We know that $V(i,j) \in [e^0, e]$. However, we can not bound the range of $H^2(i,j)/maxln(j), i = 1,2,\ldots,K, j = 1,2,\ldots,t$. Note that, if inequality (2) and inequality (3) hold simultaneously, we can say that $H^2(wk,j)/maxln(j)$ are much larger than $H^2(ck,j)/maxln(j)$, for some $S_j \in Set2$ with high probability. This case occurs due to the noise in each power trace and depends on the templates. Our new way exploits the normalized conditional probability $V(i,j)$ so that it is more accurate and effective.

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

Step 1-3 in Algorithm 1 and Algorithm 2 are called normalization process. Note that, the normalization process is of extremely low computation cost.

4 Experiments

In this section, we evaluate the key-recovery effectiveness of our improved Template Attacks and the classical Template Attacks 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. On one hand, we will evaluate the performance of our improved Template Attacks and the classical Template Attacks under different noise level using simulated power traces. On the other hand, we will perform our improved Template Attacks and the classical Template Attacks against real power traces.

Algorithm 1 Normalized Template Attack for Case 1

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

Step 1 Computes the natural logarithm of each conditional probability $P(i,j)$, namely

$$H(i,j) = lnP(i,j), i = 1,2,\ldots,K, j = 1,2,\ldots,t.$$

Step 2 Computes

$$maxln(j) = max\{H(1,j), H(2,j),\ldots,H(K,j)\}$$

for each power traces $S_j, j = 1,2,\ldots,t$.
Step 3 Computes normalized conditional probability $V(i,j)$ for each power traces:

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

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

$$key_{ck} = argmax_i\{\prod_{j=1}^{t} V(i,j), i = 1,2,\ldots,K\}.$$

Step 5 Return key_{ck}.

Algorithm 2 Normalized Template Attack for Case 2

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

Step 1 Computes the natural logarithm of each conditional probability $P(i,j)$, namely

$$H(i,j) = lnP(i,j), i = 1,2,\ldots,K, j = 1,2,\ldots,t.$$

Step 2 Computes

$$maxln(j) = max\{H(1,j), H(2,j),\ldots,H(K,j)\}$$

for each power traces $S_j, j = 1,2,\ldots,t$.
Step 3 Computes normalized conditional probability $V(i,j)$ for each power traces:

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

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

$$key_{ck} = argmax_i\{\sum_{j=1}^{t} V(i,j), i = 1,2,\ldots,K\}.$$

Step 5 Return key_{ck}.

For simplicity, let np denotes the number of simulated power traces or real power traces used in the profiling stage and let ne denotes the number of simulated power traces or real power traces used in the extraction stage. In each figure showing an experiment result, our improved Template Attack (such as Normalized Template Attack for Case 1 or Normalized Template Attack for Case 2) is denoted by "Our Method" and the classical Template Attack (such as Template Attack for Case 1 or Template Attack for Case 2) is denoted by "Classical Method".

In both the simulated experiments and practical experiments, we use different numbers of simulated or real power traces to build the templates. Different numbers of simulated or real power traces used in the process of building the templates mean that the templates have different levels of accuracy.

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 Gaussian noise is denoted by σ.

4.1.1 Simulated Experiments about Normalized Template Attack for Case 1

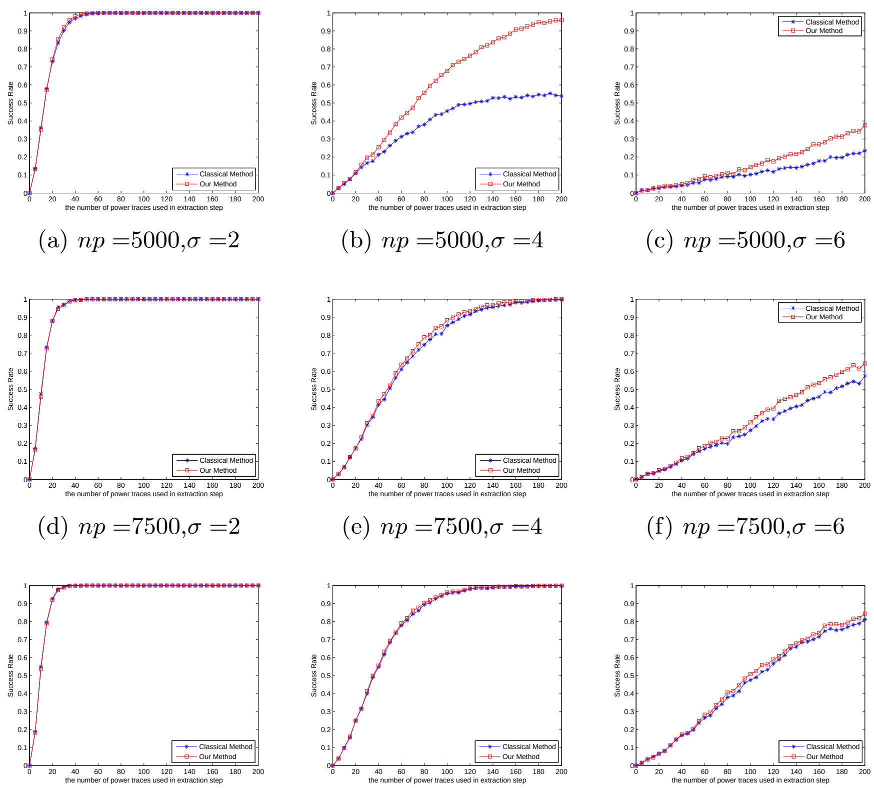
In simulated scenarios, we chose the output of the first S-box of the first round of unprotected AES-128 as the target intermediate value. The Hamming Weight power model [12] was adopted to test the effectiveness of Normalized Template Attack for Case 1 and Template Attack for Case 1. We employed three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

For each noise level, we did as follows. We used 5000, 7500, and 10000 simulated power traces to build the 256 templates respectively (Because the output of the first S-box of the first round of unprotected AES-128 is 8 bits long, we need to build 256 templates.). The three groups of simulated power traces were generated with a fixed key and random plaintext input. We generated additional 20000 simulated power traces which were used in the extraction stage with a fixed key and random plaintext input. 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 the extraction stage as follows. We executed 32 groups of the two attacks simultaneously. In each of the 32 groups, we repeated the two attacks 128 times. For each time, we chose ne traces chosen 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. For the ith group, we respectively recorded how many times the two attacks can successfully recover the correct key (denoted by

$num_{(ne,i,NTA1)}$ for Normalized Template Attack for Case 1 and $num_{(ne,i,TA1)}$ for Template Attack for Case 1). Then, for each group, we computed the Success Rate $sr_{(ne,i,NTA1)}$ ($sr_{(ne,i,NTA1)} = num_{(ne,i,NTA1)}/128$) and $sr_{(ne,i,TA1)}$ ($sr_{(ne,i,TA1)} = num_{(ne,i,TA1)}/128$). The Success Rate of the two attacks for the case one using ne traces in the extraction stage were computed by

$$SR_{(ne,NTA1)} = \frac{\sum_{i=1}^{32} sr_{(ne,i,NTA1)}}{32}, SR_{(ne,TA1)} = \frac{\sum_{i=1}^{32} sr_{(ne,i,TA1)}}{32}.$$

We will show $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ for different ne in Figure 1. Note that, in Figure 1, each subtitle represents that the two attacks (Normalized Template Attack for Case 1 and Template Attack for Case 1) are executed when the number of simulated power traces used in the profiling stage equals to np and the standard deviation of Gaussian noise equals to σ.



(a) np =5000,σ =2 (b) np =5000,σ =4 (c) np =5000,σ =6

(d) np =7500,σ =2 (e) np =7500,σ =4 (f) np =7500,σ =6

(g) np =10000,σ =2 (h) np =10000,σ =4 (i) np =10000,σ =6

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

From Figure 1, 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. For example, when $np = 5000$, $\sigma = 4$, and $ne = 200$, the $SR_{(200,NTA1)}$ equals to 0.96 while $SR_{(200,TA1)}$ equals to 0.54. In all the cases, the Success Rate of Normalized Template Attack for Case 1 are not lower than that of Template Attack for Case 1. 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 Simulated Experiments about Normalized Template Attack for 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 algorithm 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 output of S-box in SubWord and recover w[3],w[7],w[11], and w[15] completely if he obtains the output of S-box successfully. In all of our experiments for Normalized Template Attack for Case 2 and Template Attack for Case 2, we attacked the output of S-box in SubWord and tried to recover key[15] in w[3] as an example. The processes of attacking other byte 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 for different power traces are fixed. Hence, we used ID power model. We used 10000, 15000, and 20000 simulated power traces to build 256 templates respectively. The value of key[15] of each simulated power trace was generated randomly. We employed three different noise levels (Gaussian noise) to test the influence of noise on the performance of Normalized Template Attack for Case 2 and Template Attack for Case 2. The standard deviation of the three noise levels were 2, 4, and 8.

In our simulated experiments, we chose 32 random value for key[15]. For each one of the 32 values of key[15], we generated 400 simulated power traces. 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 400 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

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

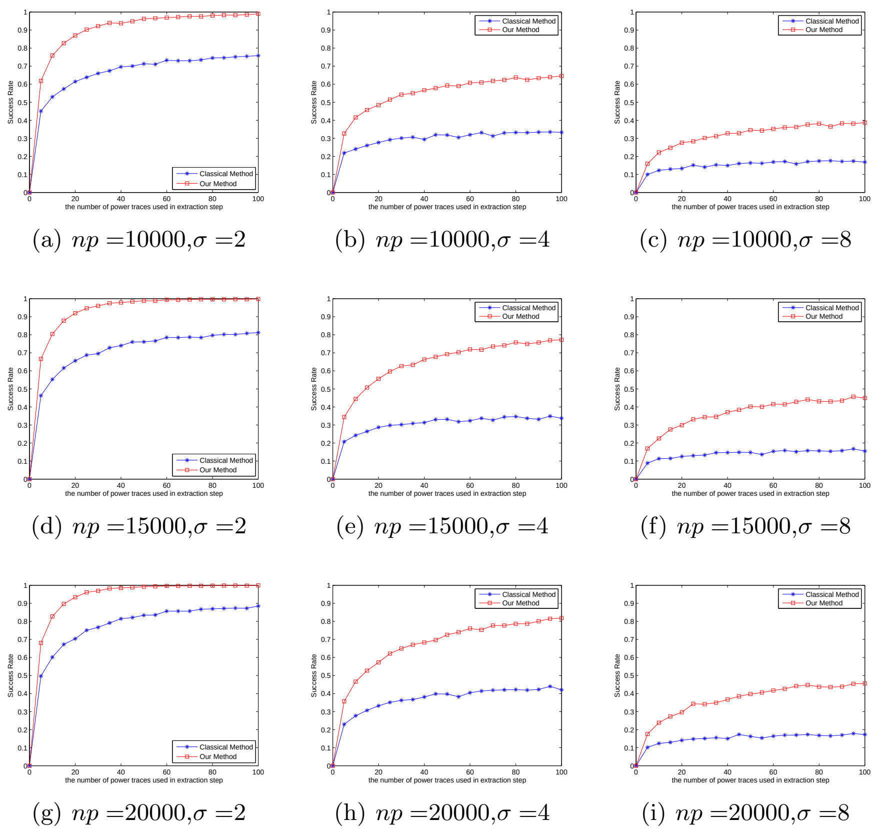
We will show $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$ for different ne in Figure 2. Note that, in Figure 2, each subtitle represents that the two attacks (Normalized Template Attack for Case 2 and Template Attack for Case 2) are executed when the number of simulated power traces used in the profiling stage equals to np and the standard deviation of Gaussian noise equals to σ.

From Figure 2, we can see that Normalized Template Attack for Case 2 is much more effective than Template Attack for Case 2 when the noise level is high. For example, when $np = 15000$, $\sigma = 4$, and $ne = 100$, the Success Rate of Normalized Template Attack for Case 2 equals to 0.77 while the Success Rate of Template Attack for Case 2 equals to 0.34. When $np = 15000$, $\sigma = 8$, and $ne = 100$, the Success Rate of Normalized Template Attack for Case 2 equals to 0.45 while the Success Rate of Template Attack for Case 2 equals to 0.16.

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.

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.



(a) np =10000,σ =2 (b) np =10000,σ =4 (c) np =10000,σ =8

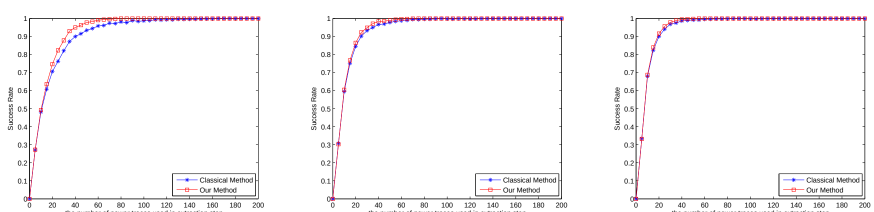
(d) np =15000,σ =2 (e) np =15000,σ =4 (f) np =15000,σ =8

(g) np =20000,σ =2 (h) np =20000,σ =4 (i) np =20000,σ =8

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

4.2.1 Practical Experiments about Normalized Template Attack for Case 1

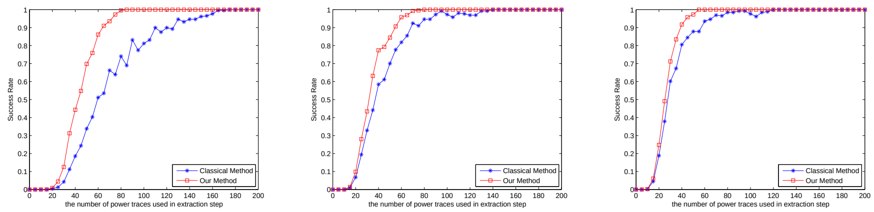
We tried to attack the output of the first S-box of the first round of an unprotected software AES-128 implementation over PowerSuite 4.0 as an example. Similarly to the simulated experiments of Normalized Template Attack for Case 1 and Template Attack for Case 1, we used 5000, 7500, and 10000 real power traces to build the 256 templates respectively. The three groups of real power traces were generated with a fixed main key and random plaintext input. We generated additional 20000 real power traces with a fixed main key and random plaintext input. 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 of Normalized Template Attack for Case 1 and Template Attack for Case 1 but with real power traces. We will show $SR_{(ne,NTA1)}$ and $SR_{(ne,TA1)}$ for different ne in Figure 3. In Figure 4, we show the Success Rate of recovering the whole main key in the first round of the unprotected software AES-128 implementation over PowerSuite 4.0 for Normalized Template Attack for Case 1 and Template Attack for Case 1 (denoted by $GSR_{(ne,NTA1)}$ and $GSR_{(ne,TA1)}$). Note that, in Figure 3 and Figure 4, each subtitle represents that the two attacks (Normalized Template Attack for Case 1 and Template Attack for Case 1) are executed when the number of real power traces used in the profiling stage equals to np. Table 1 shows the minimum number of the real power traces needed by one in the extraction stage of our practical experiments to recover the correct key successfully with probability 1 for Normalized Template Attack for Case 1 and Template Attack for Case 1 respectively.



(a) np =5000 (b) np =7500 (c) np =10000

Fig. 3. Practical Experiments Results of Normalized Template Attack for Case 1 and Template Attack for Case 1 ($SR_{(ne,NTA1)}$,$SR_{(ne,TA1)}$)

From Figure 3, Figure 4 and Table 1, we can see that Normalized Template Attack for Case 1 is more effective than Template Attack for Case 1. For example, when np = 10000, Normalized Template Attack for Case 1 only needs 55 real power traces to recover the correct key with probability 1 while Template Attack for Case 1 needs 120 real power traces.



(a) np =5000 (b) np =7500 (c) np =10000

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

Table 1. The Minimum Number of The Real Power Traces Needed by One to Recover The Correct Key Successfully With Probability 1 for Normalized Template Attack for Case 1 And Template Attack for Case 1

	np =5000	np =7500	np =10000
Our Method	85	80	55
Classical Method	175	140	120

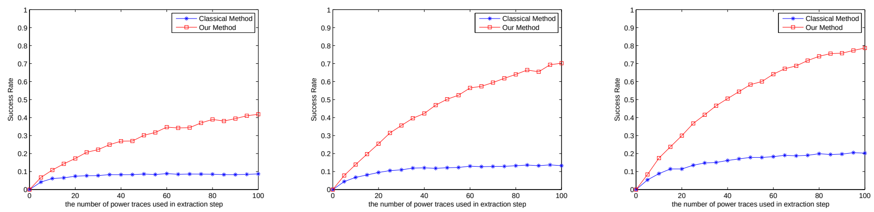
4.2.2 Practical Experiments about Normalized Template Attack for Case 2

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

Similarly to the simulated experiments, we used 20000, 40000, and 60000 real power traces with random main key to build the 256 templates respectively. In our practical experiments, we also chose 32 random main key (Thus there are 32 random key[15].). For each main key, we generated 250 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 key[15] but the ne traces used in the extraction stage were chosen from the 250 real power traces uniformly at random each time of the 128 times of attacks. Then we computed $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$. The value of $SR_{(ne,NTA2)}$ and $SR_{(ne,TA2)}$ are shown in Figure 5. In Figure 5, the subtitles represent how many real power traces are used in the profiling stage.

Table 2 shows the ratio of the Success Rate of Normalized Template Attack for Case 2 to the Success Rate of Template Attack for Case 2 for different number of real power traces used in the profiling stage and in the extraction stage.

From Figure 5 and Table 2, we can see that the Success Rate of Normalized Template Attack for Case 2 is much higher than the Success Rate of Template Attack for Case 2. For example, when np = 40000 and ne = 100, the Success Rate of Normalized Template Attack for Case 2 is 5.30 times higher than the Success Rate of Template Attack for Case 2.



(a) np =20000 (b) np =40000 (c) np =60000

Fig. 5. Practical Experiments Results of Normalized Template Attack for Case 2 and Template Attack for Case 2 ($SR_{(ne,NTA2)}$,$SR_{(ne,TA2)}$)

Table 2. The Ratio of The Success Rate of Normalized Template Attack for Case 2 to The Success Rate of Template Attack for Case 2 in Practical Experiments

ne \ np	20000	40000	60000
20	2.33	2.68	2.62
60	3.93	4.35	3.51
100	4.81	5.30	3.89

5 Conclusion and Future Work

In this paper, we find a new way to increase leakage exploitation rate in Template Attack. The new way can be used in both Template Attack for Case 1 and Template Attack for Case 2. We present Normalized Template Attack for Case 1 and Normalized Template Attack for Case 2 which have the normalization process and exploit normalized conditional probability instead of conditional probability. We also verified the two new attack methods by simulated and practical experiments. Remarkably enough, the normalization process in both our attack methods is of extremely low computation cost. Therefore, we argue that the normalization process should be integrated as a necessary part of profile attacks in order to better understand the practical threats of these attacks.

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. Z. Hailong, Z. Yongbin, and F. Dengguo.: An Efficient Leakage Characterization Method for Profiled Power Analysis Attacks. ICISC2011, LNCS 7259, pp.61-73, 2011.

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.

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

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;
   }
}
```

to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher.