How Far Can We Reach? Breaking Masked AES Smartcard Implementation Using One Trace

Wei Cheng[1], Chao Zheng[1], Yuchen Cao[1,2], Yongbin Zhou[1,2], Hailong Zhang[1], Sylvain Guilley[3,4], Laurent Sauvage[3,4]

[1]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]LTCI, Télécom ParisTech, Université Paris-Saclay, Paris, France
[4]Secure-IC S.A.S., Cesson-Sévigné, France
{wei.cheng,zhouyongbin}@iie.ac.cn,
{sylvain.guilley}@telecom-paristech.fr

Abstract. Rotating Sbox Masking (RSM) scheme is a highly efficient masking scheme proposed to protect cryptographic implementations from side channel attacks. It is a Low Entropy Masking Scheme and has attracted special attention for its low overhead but high performance. The two public targets of international academic competition DPA Contest v4 are both RSM-masked AES implementations, specifically, RSM-AES-256 for v4.1 and RSM-AES-128 for v4.2 respectively. The side channel security of RSM-AES-256 was intensively studied by researchers worldwide under the framework of DPA Contest and several flaws were identified, while the security of RSM-AES-128 is still not thoroughly studied. In this paper, we focus on analyzing the practical security of RSM-AES-128 from a profiling attack point of view. Specifically, we firstly present a Multivariate Template Attack (MTA) to maximize the success rates of key recovery attack. Next, we propose a new Depth-First Key Enumeration Algorithm (DFKEA) that could be applied to find the correct key efficiently after a side channel attack. By integrating the DFKEA to our MTA, we propose a novel multivariate profiling attack which could recover the whole secret key of RSM-AES-128 with over 95% possibility only using one electromagnetic trace. It is the best attack among all attacks submitted to DPA Contest Official up to now. Finally, we present one proposal to further improve the practical security of RSM-AES-128 at an acceptable overhead.

Keywords: Side Channel Attacks · Template Attack · DPA Contest · Countermeasures · Rotating Sbox Masking Scheme · Shuffling Scheme.

1 Introduction

Side Channel Attacks (SCA) have been proven to be a serious threat on practical security of cryptographic implementations, in which an adversary always extracts the sensitive information like secret key by statistic analysis on "physical observable" leakages [30,31,33,35]. These threats tend to get much worse with the advent of the Internet of Things (IoTs), since on one hand the IoT devices are typically too constrained with resources to deploy complex countermeasures for achieving a high security level. On the other hand, the adversary always has full control on these devices to carry out certain very intensive analysis and powerful attacks, especially including some profiling attacks. Particularly, recent attacks Spectre [6] and Meltdown [5] are essentially two Cache-based S-CA exploiting architectural vulnerabilities in CPU-level, which severely affect almost all modern CPUs. Although huge differences exist between IoT devices and CPU-based devices (like PC and cloud server) from architectures to low-level implementations, they all perform physically observable computations [12] and share the great threat induced by SCA. Therefore, countermeasures are the integrant parts of the security-critical systems.

In order to protect cryptographic implementations (devices) against SCA, many countermeasures have been proposed including masking, shuffling and hiding. Specifically, masking schemes [13,24,34,27] randomize the dependency between sensitive data and leakages by dividing each sensitive variable into several random shares to thwart SCA, while Shuffling schemes [25,26] randomize the order of operations during the executions. Quite differently, by circuit-level alteration, hiding-based countermeasures [32,34,14] make the leakages uniformly independent to the data processed. Among them, masking schemes are a class of the most attractive and frequently used techniques against SCA, since they provide formally provable security and could be implemented on algorithmic-level with no hardware alteration. Despite the improvements with respect to SCA security, almost all countermeasures always cause a significant overhead and performance loss on cryptographic implementation compared to an unprotected one. As a consequence, lightweight and efficient solutions for SCA countermeasures are very attractive for researchers and designers.

Rotating Sbox Masking (RSM) [7,11] scheme emerges as a very efficient countermeasure to provide 1st-order SCA security (actually immune to 1st- and 2nd-order zero-offset CPA [3]), which is a typical implementation of Low Entropy Masking Scheme (LEMS) [20,10,2]. It is the core protection scheme used in the latest edition of DPA Contest, namely DPA Contest v4 [16]. DPA Contest is an international open framework held for worldwide participants and researchers to evaluate and compare their attacks under a common setting [1]. Particularly, the forth edition of DPA Contest (both v4.1 and v4.2) are launched to evaluate the practical security of protected AES implementations running on an Atmel ATMega-163 smart-card. Specifically, the DPA Contest v4.1 (DPACv41)[15] has closed, while the DPA Contest v4.2 (DPACv42) is the latest version and still open. The target of DPACv41 is a RSM-masked AES-256 implementation, namely RSM-AES-256. During the DPACv41, the practical security of RSM-AES-256 was thoroughly studied from both non-profiling attacks and profiling attacks points of view, and several flaws were identified, especially the pitfalls in the RSM scheme [1,18,19] like **constant difference** in the RSM mask set, which could be exploited to break RSM scheme and then recover the secret key only using 14 traces. The RSM-AES-128 is the improved one of RSM-AES-256, in which several pitfalls were fixed [1] and it is the open target of DPACv42. The improved RSM scheme in RSM-AES-128 is also called RSM scheme, since we only focus on the improved one, there is no ambiguity in denotation.

Intuitively, the combination of different countermeasures could improve the practical security of the cryptographic implementations only if they are carefully implemented. This strategy was adopted in DPA Contest v4.2, in which both masking and shuffling are applied to upgrade the original RSM in terms of SCA security. Since several implementation flaws have been fixed (mainly by reprogramming in assembly code, precharging and the improved RSM scheme) and a new shuffling scheme is adopted [1], RSM-AES-128 is expected to achieve a high SCA security level. However, its practical security is still not intensively studied, especially from a viewpoint of profiling setting. In this paper we focus on security analysis of RSM-AES-128 using profiling attacks [23]. More concretely, although in presence of these security-oriented improvements in RSM-AES-128, our final question is that, can we recover the secret key of RSM-AES-128 only using one trace? If possible, what's the most efficient way to find the secret key?

To answer these two questions, two techniques are utilized in this paper to carry out our attack against RSM-AES-128. One is *Template Attack*, which is the strongest form of SCA from a perspective of information theory [28]. It's a natural better choice compared to other profiling attacks like Linear-Regression based attacks [4,9]. Furthermore, the multivariate attack which exploits leakages of consecutive intermediates [36] (similar to Multi-target DPA [29]) is more powerful compared to univariate counterpart. The other one is *Key Enumeration Algorithm (KEA)*. KEA is a post-processing technique to find the correct key effectively after an attack. Recently, several KEA [37,39,38] have been proposed to optimize the effectiveness, efficiency and memory overhead. Particularly, the optimal KEA [37] provides an optimal order to enumerate the results after an attack but with large memory overhead in terms of "key trails", while the Histograms-based KEA [38] leads to straightforward parallelization with simple bounds of rounding errors. These KEA methods generally assume that different subkey candidates are similarly distributed, therefore they can be viewed as breadth-first methods and adopt a combine-then-verify route. However, if the distributions of different chunks of key candidates vary from each other, existing KEA method would be less efficient since extra computational costs are induced and more key verifications are inevitable for finding the secret key.

Our Contributions. Since the goal is to carry out attacks against RSM-AES-128 with only one trace, our contributions are threefold as follows.

Multivariate Template Attack (MTA). The strong capacity on key recovering of TA comes from the accurate characterization of the data-dependent leakages, which is also called templates. In this paper, we propose a high-dimensional Multivariate Template Attack (MTA). Particularly, multivariate means our attacks targeted on multiple sensitive variables. We also use Principle Component Analysis (PCA) [40,41] as a leakage-feature extraction tool to reduce the dimensionality of our MTA, thus to dramatically decrease the data complexity of templates. By applying several optimization, our MTA can break the RSM scheme and shuffling scheme with 100% probability using one trace, meanwhile the partial success rates (PSR [8], corresponding to recovery of each subkey) of attack varies in a range from 93% to 98%, and the global success rate (GSR [8], corresponding to the recovery of entire secret key) is increased from 55% to 83% which increased 50.91% (also using one trace).

Depth-First Key Enumeration Algorithm (DFKEA). We propose a new key enumeration algorithm DFKEA featured with the high efficiency in finding the secret key after SCA. Compared to the state of the art KEA method, on one hand, DFKEA doesn't need to combine the possibilities of subkey candidates in different chunks, by which reduces the extra computations to speed up the key-finding process. On the other hand, the ranks of correct subkey hypothesis after side channel attacks could be very various in different chunks after a practical attack. Actually, our DFKEA adopts a (bounded) depth-first approach to efficiently traverse the most possible subkey hypothesis. By integrating DFKEA into our MTA scheme, its GSR is significantly increased from 83% to 95% (increased 14.46%, evaluation from DPA Contest Official) with an acceptable computation overhead. These results distinctly indicated that RSM-AES-128 is vulnerable to profiling attacks, especially for our MTA.

Shuffled Offset. Countermeasures are requisite for leakage-preventing to thwart SCA. Since the shuffling scheme and RSM scheme are independently applied in RSM-AES-128 [1], they can be compromised separately. To address this problem, by encoding the nonce used in shuffling scheme into offset, we propose a method to improve the security of RSM scheme by eliminating the mask-dependent leakage existing in full rounds of encryption process.

The rest of the paper is organized as follows: Section 2 introduces the details of RSM-AES-128 and template attacks. Section 3 explains the rationale of our MTA and its application to break the RSM scheme and shuffling scheme, and then to conduct the key-recovery attack. In section 4, the new key enumeration algorithm DFKEA is proposed with necessary validation experiments, and one proposal to further improve the practical security of RSM-AES-128 is described in section 5. Finally, conclusions are drawn in section 6 and some evaluation results of our attack from DPA Contest Official are in Appendix.

2 Preliminary and Notations

2.1 RSM-AES-128 Implementation

RSM-AES-128 [1] is a software implementation of AES-128 algorithm protected by both RSM scheme and shuffling scheme. Specifically, RSM scheme is applied to full rounds of encryption to protect all intermediate variables, while shuffling is only adopted in the first and last rounds of transformation to protect the commonly "vulnerable" part of implementation [42].

RSM Scheme. The well-designed masking and unmasking methodse make RSM very efficient. Essentially, RSM scheme is a low entropy boolean masking scheme, the mask set only contains 16 fixed mask values as follows.

$$\begin{aligned} M = \{\ & 0x03, 0x0c, 0x35, 0x3a, 0x50, 0x5f, 0x66, 0x69, \\ & 0x96, 0x99, 0xa0, 0xaf, 0xc5, 0xca, 0xf3, 0xfc\ \} \end{aligned} \tag{1}$$

where m_i and M_i both denotes the i-th mask value ($i = 0, 1, \ldots, 15$).

In a masked cryptographic implementation, all sensitive variables are required to be masked, and the nonlinear layer of the cipher is the most critical part for designers [22]. In AES, its round function consists of four subfunction, namely *KeyAdd (AK)*, *SubBytes (SB)*, *ShiftRows (SR)* and *MixColumns (MC)*. The *SubBytes* is the only nonlinear part in AES, while the other three are linear transformations, in which the RSM could be applied straightforwardly.

Let SB, MSB denote the original and masked *SubBytes* respectively, $x \in \mathbb{F}_2^n$ ($n = 8$ in AES) is an intermediate variable. Then the RSM-masked Sbox is

$$MSB_i(x) = SB(x \oplus m_i) \oplus m_{i+1}, \quad i = 0, ..., 15 \tag{2}$$

where the m_i, m_{i+1} denotes the input and output mask of Sbox respectively, which are consecutive in mask set M. The index i varies for all sixteen Sboxes, which determined by an offset vector with 16 elements, namely $\overrightarrow{offset}$, to guarantee that all Sboxes are masked independently. Finally, the mask compensations *MaskComp* are applied at the end of each round to ensure correctness.

$$MaskComp_{i,r} = \begin{cases} M_{\overrightarrow{offset}+r} \oplus MC(SR(M_{\overrightarrow{offset}+r})), \ for\ r = 1, ..., 9 \\ SR(M_{\overrightarrow{offset}+r}), \ for\ r = 10 \end{cases} \tag{3}$$

where r is the round index of AES and all indexes are increasing with *mod(16)*.
Shuffling Scheme. In RSM-AES-128, the shuffling scheme only adopted to protect the Sbox layer (*SubBytes*) in the first and last round by using *Shuffle0* and *Shuffle10* respectively. It is a 4-bit based permutation applied to change the order of sixteen Sboxes.

$$\textit{Shuffle0},\ \textit{Shuffle10} : \{0, 1, ..., 15\} \longrightarrow \{0, 1, ..., 15\} \tag{4}$$

2.2 Template Attacks

TA is the strongest form of SCA in an information theoretic sense, which assume that an adversary can fully characterize the leakage features of target device [28,23]. It's a two-phase attack consisting of profiling phase and attacking phase.

Let k^*, k denote the secret key and any possible key hypothesises respectively, and T is the public parameters like plaintext or ciphertext. We suppose that all intermediates x are elements of $\mathbb{F}_2^n$. Let $f(\cdot)$ be a mapping from T to a sensitive variable X, the measured leakages L can then be written as

$$L = \mathcal{L}(X) + Noise = \mathcal{L}(f(k^*, T)) + Noise \tag{5}$$

where $\mathcal{L}(\cdot)$ denotes the data-dependent leakage function and $Noise$ is the independent noise (as commonly assumed [28]). Particularly, $L = [L_1, L_2, \ldots, L_D]$

denotes D PoIs (points-of-interest) in a leakage trace, and a set of N measured traces are $\mathbf{L} = \{L^1, L^2, \ldots, L^N\}$.
Profiling Phase. In this phase, templates are built for each value of targeted intermediate $v \in \mathcal{V}$ in a cryptographic implementation. A template is parameterized by the mean vector and covariance matrix of the leakages corresponding to v, namely (μ_v, Σ_v). In practice, the tuple (μ_v, Σ_v) are estimated by empirical mean and covariance matrix $(\widehat{\mu}_v, \widehat{\Sigma}_v)$ as follows.

$$\begin{aligned} \widehat{\mu}_v &= \frac{1}{N_v}\sum_{i=1}^{N_v} L^{v,i} \\ \widehat{\Sigma}_v &= \frac{1}{N_v - 1}\sum_{i=1}^{N_v}(L^{v,i} - \widehat{\mu}_v)^T (L^{v,i} - \widehat{\mu}_v) \end{aligned} \tag{6}$$

where $L^{v,i}$ denotes the i-th trace in divided groups associated to intermediate variable v, while N_v is the number of v-th group of traces.
Attacking Phase. Assume that all $|\mathcal{V}|$ templates have constructed for each value of $v \in \mathcal{V}$. Let the $\mathbf{L}' = \{L^1, L^2, \ldots, L^Q\}$ denotes the Q attacking traces. In order to determine which value of v was used in L^i, the matching probabilities are computed for each template (μ_v, Σ_v) as follows.

$$p_{i,v}(L^i; (\mu_v, \Sigma_v)) = \frac{exp(-\frac{1}{2}(L^i - \mu_v)^T \Sigma_v^{-1}(L^i - \mu_v))}{\sqrt{(2\pi)^D \cdot det(\Sigma_v)}} \tag{7}$$

where again D is the dimensionality of L^i. Finally, with Maximum Likelihood (ML) principle the value of sensitive intermediate v is indicated by maximal $p_{i,v}$.

$$v* = \underset{v \in \mathbb{F}_2^n}{\arg\max}\, p_{i,v}(L^i; (\mu_v, \Sigma_v)) \tag{8}$$

Note that v is key-dependent intermediate, resulting that the secret key could be inferred after all chunks of v are recovered. Under the independence assumption among different traces, attacking results of Q traces are usually integrated for high confidence of correct key hypothesis.

In practice, SCA are more likely to focus on single sensitive intermediate, especially true when against protected implementations. However, under the context of profiling attacks like TAs, an attacker always has capacity to carry out attacks against several directly associated intermediates in sequence.

3 Multivariate Template Attack against RSM-AES-128

Under the open framework of DPA Contest v4.2, attackers are allowed to access all information and leakage datasets including cryptographic parameters and measured electromagnetic traces. In this section we propose a Multivariate Template Attack (MTA), in which high-dimensional leakages are exploited simultaneously. More importantly, our attack provide a new practical perspective of

TA integrated with PCA against protected cryptographic implementation with minimal number of traces, while the latter is mainly adopted for leakage-features extraction on physical leakages rather than data-dimension reduction.

The Onion-peeling strategy is the core to our attack against RSM-AES-128. Firstly, TAs are applied against masking scheme and shuffling scheme by recovering $\overrightarrow{offset}$ and *Shuffle0*, respectively. Once the combined countermeasure is compromised, a MTA is applied to recover the secret key with as high success rates as possible. Finally, post-processing methods like our DFKEA in Sec.4 are employed to dramatically improve the success rates at a practical tractable cost.

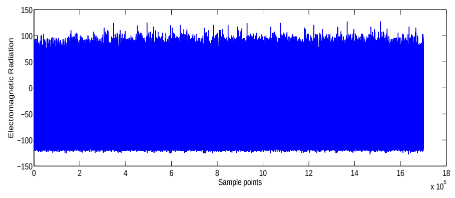
Let $V_i, i \in [1, 2, 3, 4, 5]$ denote the sensitive intermediates related to masks, shuffles, inputs of Sbox (Sboxin), Outputs of Sbox (Sboxout) and inputs to the MixColumns (MCin) respectively, thus $V_{i,j}, j \in [0, 1, \ldots, 15]$ is j-th byte, and $v_{i,j}$ is the instance of $V_{i,j}$.

$$L_i = \mathcal{L}(V_i) + Noise = HW(V_i) + Noise, \quad for\ i = 1, ..., 5$$
$$V_i = \begin{cases} M_{\overrightarrow{offset}}, & i = 1 \\ Shuffle0, & i = 2 \\ M_{\overrightarrow{offset}} \oplus K \oplus T, & i = 3 \\ MSB(M_{\overrightarrow{offset}} \oplus K \oplus T), & i = 4 \\ SR(MSB(M_{\overrightarrow{offset}} \oplus K \oplus T)), & i = 5 \end{cases} \tag{9}$$

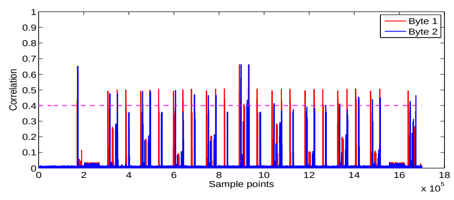
where $SR(\cdot)$ is *ShiftRows* in AES, and the same as usual, Hamming Weight (HW) model are used for leakage detection and PoI selection.

3.1 Breaking the Combined Countermeasure

Attacking RSM scheme. All masks in RSM scheme are determined by $\overrightarrow{offset} = \{s_0, s_1, \ldots, s_{15}\}$. In the first round of encryption, all masks are involved into the encryption process. As a consequence, our TA targets all M_{s_i} in the first round for simplicity (similar to carry out TA targeting at last round operations). The leakage features of the first two masks of RSM-AES-128 are depicted as Fig.1(b).



(a) Plotting one electromagnetic trace



(b) Leakage of the first two Masks.

Fig. 1. (a) Visualization of one trace, and (b) leakage of masks in the first two bytes of RSM-AES-128 (HW, Correlation).

Our proposed TA follows a typical two-step roadmap as follows. In profiling phase we build templates for all possible values of $V_{1,j}, j \in [0, 1, \ldots, 15]$. Apparently, recovering masks is equivalent to recover the $\overrightarrow{offset}$. Since PCA is utilized to extract device-specific leakage features, for each $V_{1,j}$, leakages selected from about 1200 PoIs are integrated, then only top 10 components corresponding to first 10 eigenvalue (data-dimensions are reduced from $D = 1200$ to $D = 10$) are chosen to build templates $(\mu_{V_{1,j}}, \Sigma_{V_{1,j}})$ as Equ.6. In attacking phase, Equ.7 and Equ.8 are applied to determine the most possible mask (or equivalently $s_j, j \in [0, 1, \ldots, 15]$) hypothesis after PCA.

Our experimental results validated the effectiveness of our PCA-based TA against RSM scheme. The success rates for recovering all masks are 100%. One main reason is that all masks involved almost all cryptographic operations throughout entire execution of RSM-AES-128, thus from an attacker's point of view, these operations leak sufficient information to recover all masks.

Attacking Shuffling scheme. The attack of shuffling scheme is very similar to compromise RSM scheme. The major difference and main difficulty is the very limited exploitable leakages only leaking from Sbox-shuffling in the first round of RSM-AES-128. As a consequence, in order to recover the *Shuffle0* by one trace, different number of PoIs are used for each element of *Shuffle0* (as Tab.1).

Table 1. Number of PoIs for PCA and selected number of components after PCA

Byte index	‖PoIs‖	‖Components‖	Byte index	‖PoIs‖	‖Components‖
0	571	85	8	419	85
1	436	85	9	507	85
2	493	85	10	421	85
3	427	85	11	506	85
4	458	85	12	425	85
5	453	85	13	523	85
6	477	85	14	424	85
7	492	85	15	406	85

Despite the variety existed in number of PoIs when attacking different elements of *Shuffle0*, the success rate of recovering *Shuffle0* is 100%.

Once the combined countermeasure is compromised, RSM-AES-128 becomes an unprotected implementation, thus the key-recovery attack can be carried out to recover the secret key. Particularly, our experiments also validate the effectiveness of applying PCA and a profiling attack to break countermeasures in a very efficient way (especially from a engineering perspective).

3.2 Our Multivariate Template Attack

We propose a MTA to retrieve the secret key in an effective way. Here multivariate means that multiple key-dependent variables are targeted to obtain subkeys. Specifically, our MTA targets at three sensitive variables depending on the same subkey. These three variables are $V_{3,j} = M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j$, $V_{4,j} = MSB(M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j)$ and $V_{5,j} = SR(MSB(M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j))$. Note that our MTA combines the results of three univariate TAs, rather than combining leakages in measured traces before the attack to keep low computational overheads (similar idea applied in [29] but with a non-profiling setting).

Similarly, in profiling phase, all templates are built for each possible value of $V_{3,j}, V_{4,j}$ and $V_{5,j}, j \in [0, 1, \ldots, 15]$ using Equ.6. The PCA is also used to extract features of data-dependent leakages and to reduce the data complexity. Concretely, leakages selected from 500 PoIs ($D = 500$) are feeded into PCA and various number of components are chosen to build templates, which the latter differs from the component-choosing method in recovering *Shuffle0*. The number of selected components are listed in Tab.2.

Table 2. Number of chosen components after PCA for $V_{3,j}, V_{4,j}$ and $V_{5,j}$

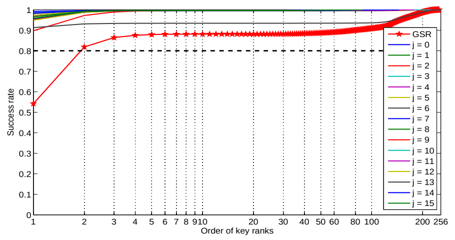
Byte index j	$\|V_{3,j}\|$	$\|V_{4,j}\|$	$\|V_{5,j}\|$	Byte index j	$\|V_{3,j}\|$	$\|V_{4,j}\|$	$\|V_{5,j}\|$
0	70	50	55	8	60	50	55
1	75	50	60	9	50	50	50
2	60	50	55	10	60	50	50
3	75	50	50	11	70	50	50
4	60	50	65	12	60	50	50
5	60	50	50	13	70	50	60
6	65	50	50	14	60	50	55
7	55	50	50	15	55	50	55

Subsequently, in attacking phase, leakages from selected PoIs firstly feed into PCA and then Equ.7 is applied to obtain the ranked subkey candidates, which sorted by their possibilities. Since the combined protection scheme has been compromised, we focus on key-dependent variables by assuming that $\overrightarrow{offset}$ and *Shuffle0* are known for following analysis.

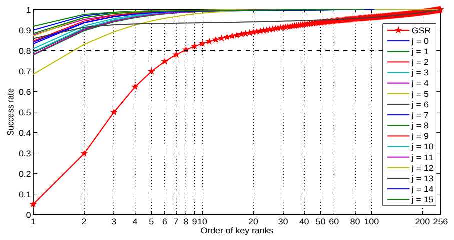
The d-th order partial and global success rates $V_{3,j}, V_{4,j}, V_{5,j}$ $(j \in [0, \ldots, 15])$ and our MTA are plotted in Fig.2. Particularly, for our MTA, Fig.2(d) shows that the first-order PSR of all subkey bytes are over 98%, except the fourteenth subkey byte for 93%. In addition, we can observe that:

1. PSR of all subkey candidates follow a extreme Pareto distribution, which means the roughly 10% of subkey candidates lead to almost 90% of PSR. The lower the rank (from 1 to 256 and the lowest one is 256-th) of subkey candidates, the lower possibility of it to be the correct subkey hypothesis.
2. More importantly, compared to univariate TA, MTA makes the correct subkey hypothesis approach to the highest ranks among all subkey candidates, even for the worst subkey. This approaching effect also validates the positive impact of our MTA on revealing the secret key. Consequently, the GSR of MTA is significantly higher than all three univariate TAs.
3. Different byte-positions (indexes) of subkeys have small effects on attacking results except the fourteenth subkey, which is the worst byte of all subkeys in all four cases. However, this "information" could be exploited to improve the success rates by post-processing techniques, especially to improve GSR. But in order to keep the generality of our attack, this "information" is not utilized in our MTA and following DFKEA.

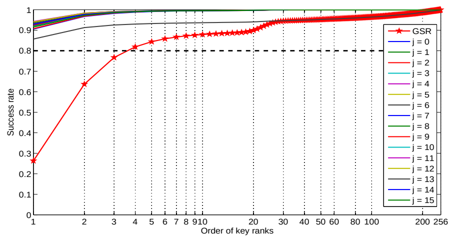
In practice, it's advantageous to make use of information about subkey rank distributions to maximize the GSR with constrained computational complexity.



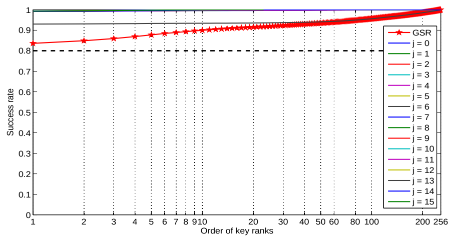
(a) Attacking $V_{3,j}$ (Sboxin).



(b) Attacking $V_{4,j}$ (Sboxout).



(c) Attacking $V_{5,j}$ (MCin).



(d) Our MTA.

Fig. 2. Partial Success Rates (PSR) and Global Success Rate (GSR) of key-recovery attacks targeted at (with logarithmic X-axis) (a) $V_{3,j}$, (b) $V_{4,j}$, (c) $V_{5,j}$ and (d) our MTA for each j-th subkey, $j \in [0, \ldots, 15]$, only one trace is used in all four cases.

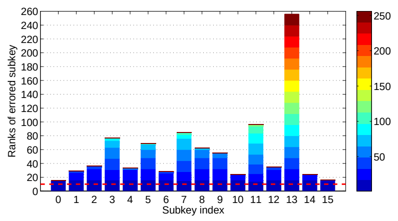
In next section, we propose the Depth-First Key Enumeration Algorithm to dramatically improve the GSR of MTA based on observation 1 and 2.

4 Depth-First Key Enumeration Strategy

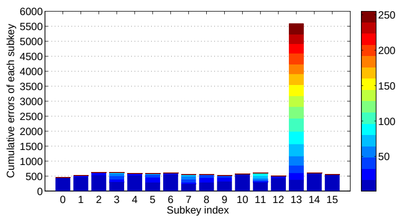
Our primary interest is to recover the secret key of RSM-AES-128 in a very efficient way, thus two main requirements for post-processing methods are the high efficiency of key-finding and the maximal success rates. The former requires the minimal number of key verifications and less extra computations which may be caused by combining all sixteen sorted subkey lists together to obtain global key sorting results [37,39,38]. While the latter requires a high coverage of the highly possible subkey candidates.

Keeping these requirements in mind, we firstly investigate the distribution of errored subkey candidates. Here "errored" subkey means its correct candidate is not ranked first among all possible candidates. The ranks of errored subkeys and cumulative errors of each subkey are depicted as Fig.3. Unsurprisingly, same as observed in Sec.3.2, the fourteenth subkey is the worst case among all subkeys from an attacking point of view.

In order to improve the coverage of the most possible subkey candidates, we also inspect the number of errored subkeys N_{err} per attack (or per trace) as showed in Fig.4. By using 25,000 to 40,000 traces, two main observations are:



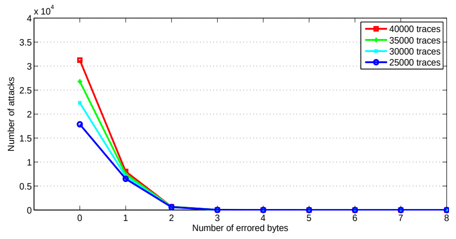
(a) The ranks of errored subkeys.



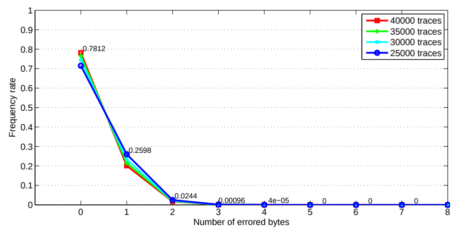
(b) Cumulative errors of each subkey.

Fig. 3. The distribution of errored subkey candidates, "errored" means failed to recover a subkey by first-order rank (80,000 traces from DPA Contest v4.2 [16] are used).

1) $N_{err} \leq 4$, and 2) the frequency of each N_{err} significantly decreases with the increase of N_{err}. Hence, if we take N_{err} into consideration, and enumerate all possible subkey candidates along with the increase of N_{err}, the most possible subkey could be covered. Thus key-recovery could be very efficient. Hereafter, we propose a Depth-First Key Enumeration Algorithm (DFKEA) to utilize these N_{err}-related information to find the secret key. To keep portability to other attack scenarios, we insist on only utilizing statistics of N_{err} in our DFKEA.



(a) Number of Errored Subkeys N_{err}



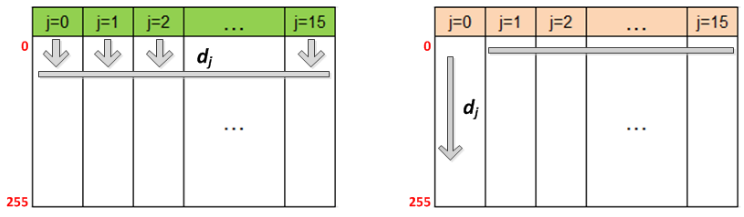
(b) Frequency

Fig. 4. The distribution of number of errored subkeys (N_{err}) with number of traces from 25,000 to 40,000 (the first eight datasets from DPA Contest v4.2 [16]).

4.1 Proposal of DFKEA

Main idea behind the DFKEA is to enumerate subkey candidates along with increase of N_{err} in a depth-first way. Specifically, we firstly verify the subkey candidates with $N_{err} = 0$, which means to enumerate candidates with the highest rank. Next, subkey candidates with $N_{err} = 1$ are enumerated, that is to test all candidates in which only one subkey is not ranked first (while other fifteen correct candidates ranked first). Similarly, candidates with N_{err} from 2 to 15 can be enumerated and then verified. Particularly, based on aforementioned observations, our key enumerations could be efficiently done by restricting $N_{err} \leq 4$.

Let d_j denotes the enumeration depth of j-th subkey byte in our DFKEA for $j \in [0, \ldots, 15]$, and $d_{j,n_{err}}$ for d_j with $N_{err} = n_{err}$. Note that we have different $d_{j,n_{err}}$ according to varied N_{err}. On the basis of observations from Fig.4, the schematic of our DFKEA with $N_{err} = 1$ is depicted as Fig.5(b). In fact, principles behind of the state-of-the-art KEA methods are the breadth-first strategy as illustrated in Fig.5(a), in which all sixteen subkeys candidates are enumerated with an approximately equal depth. On the contrary, our DFKEA adopts a depth-first strategy, which treat each byte of subkey differently, resulting with different enumeration depth d_j for each byte of subkey.



(a) A breadth-first KEA (b) Our DFKEA with $N_{err} = 1$

Fig. 5. Main enumeration strategies of the breadth-first KEA vs our DFKEA.

Our algorithm of DFKEA is described as Alg.1. Note that convertKey(c_i) in line 10 converts each combination c_i to the errored subkey indexes by which these errored subkeys will be enumerated later.

Algorithm 1 Our Depth-First Key Enumeration Algorithm (DFKEA)

```
Input  : Sixteen lists of possible subkey candidates, $skCandi[256][16]$,
         $N_{err}$, $d_{j,n_{err}}[]$, plaintexts $P[16]$ and ciphertexts $C[16]$.
Output: Secret key or the most possible subkey candidates $sk[16]$.
 1: $Flag = false$
 2: if $true ==$ KeyVerification($P[], C[], skCandi[1][]$) then
 3:     Flag $= true$
 4:     return $sk[] = skCandi[1][]$
 5: else
 6:     for $n_{err} \in [0, N_{err}]$ do
 7:         $comb[n_{err}] = \{C_{16}^{n_{err}}$ elements$\}$ /* $C_m^k$ is the combination formula */
 8:         for $c_i \in comb[n_{err}]$ do
 9:             /*convertKey($c_i$) converts $c_i$ to subkey indexes for enumeration */
10:             $keyInd$ = convertKey($c_i$)
11:             for $key \in skCandi[d_{j,n_{err}}[]][keyInd]$ do
12:                 $Flag =$ KeyVerification($P[], C[]$, key)
13:                 if $true == Flag$ then
14:                     return $sk[] = key$
15:                 end if
16:             end for
17:         end for
18:     end for
19: end if
20: if $false == Flag$ then
21:     return $sk[16] = skCandi[1][]$ /*Deal with failures on finding the correct key */
22: end if
```

In practice, the efficiency is straightforwardly determined by the number of key verifications. For our DFKEA, we assume that the enumeration depths $d_{j,n_{err}}$ keep the same for each subkey, while change with different n_{err}. Note that this setting is for the sake of simplicity but not necessary, since if an attacker knows that attacks on some subkeys are always worse than others, their enumeration depth $d_{j,n_{err}}$ could be larger to improve the coverage of possible candidates. Let EC denotes the number of enumerations, $EC_{n_{err}}$ denotes EC for $N_{err} = n_{err}$, and the first-order success rate of each subkey is p, thus we assume that n_{err} obeys the binomial distribution, $n_{err} \sim B(16, p)$. The total number of enumerations EC_{total} can be computed as follows.

$$\begin{aligned} EC_{n_{err}} &\leqslant C_{16}^{n_{err}} * (d_{j,n_{err}} - 1)^{n_{err}} \\ EC_{total} &= \sum_{n_{err}=0}^{N_{err}} Pr(n_{err}) * EC_{n_{err}} \\ &\leqslant \sum_{n_{err}=0}^{N_{err}} C_{16}^{n_{err}} * (1-p)^{n_{err}} * p^{16-n_{err}} * (d_{j,n_{err}} - 1)^{n_{err}} \end{aligned} \tag{10}$$

where $\leqslant$ is used because our DFKEA will stop (Early Stopping) once the correct key is found, and $d_{j,n_{err}}$ keep the same for $j \in [0, \ldots, 15]$. Typically, let $n_{err} = 1$ and $d_{j,n_{err}} = d_{j,1} = 256$, thus the maximal number of enumerations is $EC_{n_{err}} = EC_1 = C_{16}^1 * 255 \approx 2^{12}$. For increased $n_{err} = 3$, we set $d_{j,n_{err}} = d_{j,3} = 10$, thus $EC_3 = C_{16}^3 * (10-1)^3 \approx 2^{18.64}$. Apparently, n_{err} (or more precisely, N_{err}) and $d_{j,n_{err}}$ directly affect the number of enumerations for our DFKEA and the coverage of possible subkey candidates.

4.2 Analysis and Experimental Results

The efficiency of an algorithm is critical in practice. Here, we primarily take the number of enumerations EC into account, which is also the number of key-verifications. The computational complexity comparison of our DFKEA with the simple KEA is tabled as Tab.3. Importantly note that the state-of-the-art KEA adopts the same breadth-first strategy with this simple KEA.

Table 3. Comparison of our DFKEA and a simple KEA with EC as evaluation criteria, *total* is $max(EC_{total})$ computed with $N_{err} = 3$

Depth $d_{j,n_{err}}$	DFKEA using Equ.10				Simple KEA
	$n_{err} = 1$	$n_{err} = 2$	$n_{err} = 3$	*total* ($N_{err} = 3$)	
2	$2^{4.00}$	$2^{6.91}$	$2^{9.13}$	$2^{9.44}$	$2^{16.00}$
3	$2^{5.00}$	$2^{8.91}$	$2^{12.13}$	$2^{12.29}$	$3^{16} \approx 2^{25.36}$
4	$2^{5.58}$	$2^{10.08}$	$2^{13.88}$	$2^{13.99}$	$4^{16} = 2^{32.00}$
5	$2^{6.00}$	$2^{10.91}$	$2^{15.13}$	$2^{15.21}$	$5^{16} \approx 2^{37.15}$
10	$2^{7.17}$	$2^{13.25}$	$2^{18.64}$	$2^{18.67}$	$10^{16} \approx 2^{53.15}$
20	$2^{8.25}$	$2^{15.40}$	$2^{21.87}$	$2^{21.89}$	$20^{16} \approx 2^{69.15}$
256	$2^{11.99}$	$2^{22.90}$	$2^{33.11}$	$2^{33.11}$	$256^{16} = 2^{128.00}$

In Tab.3, it is divided into two parts by the table-cells colored with green. The upper half part is featured with $EC \leqslant 2^{30}$, while the lower-right part is on the contrary. Obviously, compared to this breadth-first KEA, our DFKEA are much powerful on enumerating candidates with high enumeration depths. Although we always theoretically assume that attacks against different subkeys would obtain similar distribution for different candidates, practical results of different subkeys always varies from each other, especially when using electromagnetic traces. Hence, our DFKEA is very suitable for these attacking scenarios.

With the same 40,000 traces as in Fig.4, we practical evaluate the effectiveness of our DFKEA. Considering the high efficiency requirement, we restrict the enumeration depthes with $EC \leqslant 2^{20}$, which means that roughly $d_{j,1} = 256$, $d_{j,2} = 20$, $d_{j,3} = 10$ and $d_{j,4} = 5$ for $n_{err} = 1, 2, 3, 4$, respectively. For purpose of comparison, the enumeration depth of simple KEA is set to $d_1 = 2$ and $d_2 = 5$ which means $EC_1 = 2^{16}$ and $EC_2 = 2^{37.15}$, respectively. The experimental results are depicted as Fig.6. Quite clearly, our DFKEA is significantly better than the simple KEA, and the success rates increase by 24.06% and 20.05% compared to the breadth-first KEA for $d_1 = 2$ and $d_2 = 5$, respectively. These results are also in very accordance with our observations from Fig.4 that the number of errored subkeys dropped off sharply with the increase of n_{err}, thus it's very advantageous to adopt the depth-first strategy as in DFKEA. However, as a post-processing technique, DFKEA also improves the error-tolerant capability of side channel attacks, since not only the first-ranked subkey candidates, but also some of the most possible subkey candidates would be verified to obtain a high success rate.

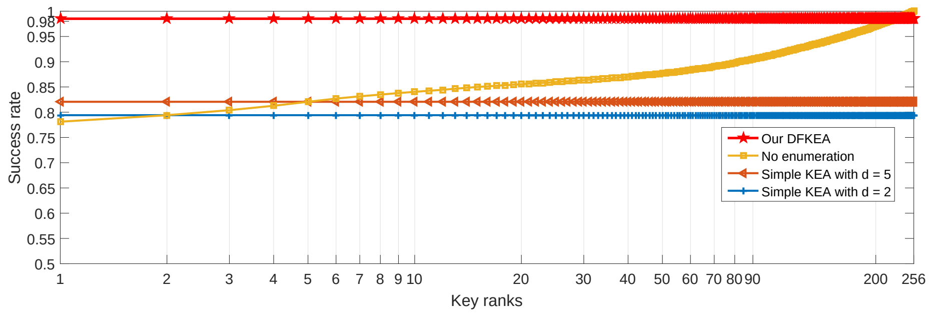


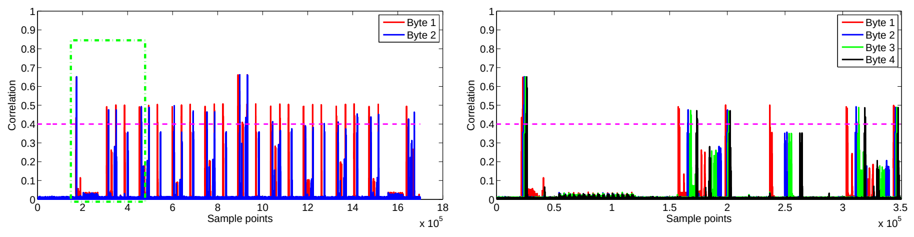
Fig. 6. Comparison between our DFKEA and the breadth-first KEA with respect to different enumeration depth (with logarithmic X-axis).

To summarize up, experimental results strongly validate the effectiveness of our DFKEA in terms of both key-recovery efficiency and attacking success rates (also equivalent to guessing entropy [8]). More importantly, compared to the simple KEA, our DFKEA doesn't require to combine all sixteen lists of candidates to get a global key-ranking list [38], which in return reduces computational burden and makes our DFKEA much more efficient than the breadth-first KEAs. Furthermore, our DFKEA is also generally applicable to both profiling and non-profiling attacks in SCA. Assembling our MTA with DFKEA, our attacking scheme can recover the entire secret key of RSM-AES-128 by using only one electromagnetic trace with a probability of more than 95% (more details refer to official evaluation report released by DPA Contest v4.2[17]).

5 Proposal for Further Improving RSM-AES-128

RSM scheme is a first-order masking scheme which featured with high efficiency and low overhead, even integrated with shuffling scheme as in RSM-AES-128. Practically, it's hard to retrieve masks (offsets) and shuffling vectors used in protected cryptographic implementations (e.g. RSM-AES-128) with non-profiling attacks, but these sensitive parameters aren't immune to profiling attacks like TA and leakage "fingerprints" matching methods [43]. This is also evidently validated by our attacks in Sec.3. In fact, our attack exploits the distinct leakage features to differentiate and recover the masks and shuffle vectors. Therefore, the core to improve the practical security of RSM-AES-128 (or other RSM-masked implementations) is to reduce or even eliminate these distinguishable leakage features. Based on our observations and results of MTA and DFKEA, we hereafter present one proposal to improve the practical security of RSM-AES-128.

Shuffled Offset Method. Considering the RSM scheme applied in RSM-AES-128, all $16 \times 11 = 176$ masks are determined by sixteen offsets denoted as $\overrightarrow{offset}$. In fact, each element of $\overrightarrow{offset}$ determines all eleven masks used in entire ten rounds. From attacking point of view, it's a deterministic relation between each offset and corresponding eleven masks. As a result, the leakages of every eleven masks could be exploited to recover corresponding offset[43]. For illustration, leakages of the first two and four bytes of masks are showed in Fig.7.



(a) One trace with 1,700,000 samples (b) Zoomed in from 150,000 to 500,000.

Fig. 7. Leakages of (a) masks used in the first two byte positions and (b) masks used in the first four byte positions respectively (HW, Correlation, with 40,000 traces).

In Fig.7(a), it's obvious that each of eleven masks corresponding to the first two bytes leaked the mask-dependent information during whole encryption process, which could be exploited to reveal the first two offsets. Similarly, all sixteen offsets could be revealed. By zoomed in view in Fig.7(b), it's clear that leakages of masks in the first four bytes occur in a deterministic sequential order.

In order to randomize the leakages of offsets during whole encryption (decryption) process, we propose a new masking and unmasking method for RSM scheme. The rationale of new $MaskComp(\cdot)$ is to update all offsets for every rounds of encryption. Specifically, we only change the offsets and mask compensation adopted in RSM scheme of RSM-AES-128. Namely, the new offset vector is determined by $\overrightarrow{offset}_{in}$ and round index r as follows.

$$MSB'(X) = SB(X \oplus M_{\overrightarrow{offset}_{in}+(r-1)}) \oplus M_{\overrightarrow{offset}_{in}+r} \tag{11}$$

where $MSB'(\cdot)$ is the new masked sboxes. In order to improve efficiency, we assign $\overrightarrow{offset}_{in} = \overrightarrow{offset} + sh[r-1]$, where $sh[r]$ is the r-th element of *Shuffle0*. Then the masking and compensation could be done easily as follows.

$$\begin{aligned} MSB'(X) &= SB(X \oplus M_{\overrightarrow{offset}+sh[r-1]+(r-1)}) \oplus M_{\overrightarrow{offset}+sh[r-1]+r} \\ MaskComp'_{i,r} &= \begin{cases} M_{\overrightarrow{offset}+sh[r]+r} \oplus MC(SR(M_{\overrightarrow{offset}+sh[r-1]+r})), \\ \qquad for\ r = 1,...,9 \\ SR(M_{\overrightarrow{offset}+sh[r-1]+r}),\ for\ r = 10 \end{cases} \end{aligned} \tag{12}$$

More importantly, our new improved RSM scheme is more secure than RSM scheme used in RSM-AES-128, since it would be degraded to the original RSM scheme if the *Shuffle0* is compromised or deactivated by adversaries (in the worst case). By using independent $\overrightarrow{offset}$ in adjacent rounds, attacks exploiting multi-round leakages[43] to against RSM scheme can be effectively hindered.

Apparently, our proposal is compatible with other proposals like new mask set[21] to provide a high level of practical security for cryptographic devices. In addition, it could be implemented on the other platforms like FPGA integrated with typical hardware-oriented countermeasures (e.g. random delays[34]).

6 Conclusions and Perspectives

As a highly efficient and lightweight masking scheme, RSM is proposed to protect cryptographic implementations like AES from side channel attacks. In this paper, by means of profiling attacks, we have thoroughly analyzed the practical security of public target of DPA Contest v4.2, namely RSM-AES-128 in an extreme condition (using only one trace). Specifically, under the framework of DPA Contest v4.2, we firstly propose a Multivariate Template Attack, which recovers the secret key of RSM-AES-128 with global success rate increased from 55% to 83% (significantly increased by 50.91%). Secondly, based on several attack-oriented observations, we propose a new Depth-First Key Enumeration Algorithm (DFKEA) to further improve the global success rates of our attack. After integrated with DFKEA, the global success rate of our MTA soars to about 95% (evaluated by DPA Contest Official), with a increase of 14.46% compared to original MTA. Furthermore, theoretical analysis and experiments shows that our DFKEA is definitely more effective than the breadth-first KEA. Finally, we propose a *shuffled offset* method to improve the practical security of RSM-AES-128, in which the leakages directly associated to consecutive offsets are eliminated.

However, although our DFKEA is very efficient than a simple breadth-first KEA, there are still some theoretical and practical issues need to be studied, like the quantitative relation between the global success rate and enumeration depths. From a more practical point of view, parallel computing is very suitable for key enumeration algorithm, especially using GPU platforms. Therefore, it's worthwhile to transform our DFKEA to a paralleled fashion in future work.

Acknowledgments. This work was supported in part by National Natural Science Foundation of China (Grant No.61632020, No.61472416 and No.61602468).

References

1. Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, Zakaria Najm. Analysis and Improvements of the DPA Contest v4 Implementation. SPACE 2014: 201-218 (2014)
2. Houssem Maghrebi, Sylvain Guilley, Jean-Luc Danger. Leakage Squeezing Countermeasure against High-Order Attacks. WISTP 2011: 208-223 (2011)
3. Eric Brier, Christophe Clavier, Francis Olivier. Correlation Power Analysis with a Leakage Model. CHES 2004: 16-29 (2004)
4. Julien Doget, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert. Univariate side channel attacks and leakage modeling. J. Cryptographic Engineering 1(2): 123-144 (2011)
5. Lipp Moritz, Schwarz Michael, Gruss Daniel, Prescher Thomas, Haas Werner, Mangard Stefan, Kocher Paul, Genkin Daniel, Yarom Yuval, Hamburg Mike. Meltdown. ArXiv e-prints, arXiv 2018:1801.01207 (2018)
6. Kocher Paul, Genkin Daniel, Gruss Daniel, Haas Werner, Hamburg Mike, Lipp Moritz, Mangard Stefan, Prescher Thomas, Schwarz Michael, Yarom Yuval. Spectre Attacks: Exploiting Speculative Execution. ArXiv e-prints, arXiv 2018:1801.01203 (2018)
7. Shivam Bhasin, Claude Carlet, Sylvain Guilley. Theory of masking with codewords in hardware: low-weight dth-order correlation-immune Boolean functions. IACR Cryptology ePrint Archive 2013: 303 (2013)
8. François-Xavier Standaert, Tal Malkin, Moti Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. EUROCRYPT 2009: 443-461 (2009)
9. Nicolas Bruneau, Claude Carlet, Sylvain Guilley, Annelie Heuser, Emmanuel Prouff, Olivier Rioul. Stochastic Collision Attack. IEEE Trans. Information Forensics and Security 12(9): 2090-2104 (2017)
10. Claude Carlet, Jean-Luc Danger, Sylvain Guilley, Houssem Maghrebi. Leakage Squeezing of Order Two. INDOCRYPT 2012: 120-139 (2012)
11. Maxime Nassar, Youssef Souissi, Sylvain Guilley, Jean-Luc Danger. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. DATE 2012: 1173-1178 (2012)
12. Silvio Micali, Leonid Reyzin. Physically Observable Cryptography (Extended Abstract). TCC 2004: 278-296 (2004)
13. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. CRYPTO 1999: 398-412 (1999)
14. Pablo Rauzy, Sylvain Guilley, Zakaria Najm. Formally Proved Security of Assembly Code Against Leakage. IACR Cryptology ePrint Archive 2013: 554 (2013)
15. TELECOM ParisTech SEN Research Group. The DPA Contest 4th Edition. 2013-2014, DPACv41, http://www.dpacontest.org/v4/index.php (2014)

16. TELECOM ParisTech SEN Research Group. The DPA Contest 4th Edition. 2014-2018, DPACv42, **http://www.dpacontest.org/v4/42_doc.php** (2018)
17. Evaluation Reports of the Submitted Attacking Scheme. Available at: **http://www.dpacontest.org/v4/evals/2016-01-25_anonymous.pdf**. (2016)
18. Amir Moradi, Sylvain Guilley, Annelie Heuser. Detecting Hidden Leakages. ACNS 2014: 324-342 (2014)
19. Xin Ye, Thomas Eisenbarth. On the Vulnerability of Low Entropy Masking Schemes. CARDIS 2013: 44-60 (2013)
20. Maxime Nassar, Sylvain Guilley, Jean-Luc Danger. Formal Analysis of the Entropy / Security Trade-off in First-Order Masking Countermeasures against Side-Channel Attacks. INDOCRYPT 2011: 22-39 (2011)
21. Nikita Veshchikov, Sylvain Guilley. Implementation flaws in the masking scheme of DPA Contest v4. IET Information Security 11(6): 356-362 (2017)
22. Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, Matthieu Rivain. Higher-Order Masking Schemes for S-Boxes. FSE 2012: 366-384 (2012)
23. François-Xavier Standaert, François Koeune, Werner Schindler. How to Compare Profiled Side-Channel Attacks?. ACNS 2009: 485-498 (2009)
24. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. CHES 2007: 28-44 (2007)
25. Christoph Herbst, Elisabeth Oswald, Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. ACNS 2006: 239-252 (2006)
26. Matthieu Rivain, Emmanuel Prouff, Julien Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. CHES 2009: 171-188 (2009)
27. Matthieu Rivain, Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. CHES 2010: 413-427 (2010)
28. Suresh Chari, Josyula R. Rao, Pankaj Rohatgi. Template Attacks. CHES 2002: 13-28 (2002)
29. Luke Mather, Elisabeth Oswald, Carolyn Whitnall. Multi-target DPA Attacks: Pushing DPA Beyond the Limits of a Desktop Computer. ASIACRYPT (1) 2014: 243-261 (2014)
30. Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. CRYPTO 1996: 104-113 (1996)
31. Paul C. Kocher, Joshua Jaffe, Benjamin Jun. Differential Power Analysis. CRYPTO 1999: 388-397 (1999)
32. Christophe Clavier, Jean-Sébastien Coron, Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. CHES 2000: 252-263 (2000)
33. Karine Gandolfi, Christophe Mourtel, Francis Olivier. Electromagnetic Analysis: Concrete Results. CHES 2001: 251-261 (2001)
34. Stefan Mangard, Elisabeth Oswald, Thomas Popp. Power analysis attacks - revealing the secrets of smart cards. Springer 2007, ISBN 978-0-387-30857-9, (2007)
35. Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, Pankaj Rohatgi. The EM Side-Channel(s). CHES 2002: 29-45 (2002)
36. Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, Yannick Teglia. Multi-variate High-Order Attacks of Shuffled Tables Recomputation. CHES 2015: 475-494 (2015)
37. Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. Selected Areas in Cryptography 2012: 390-406 (2012)
38. Romain Poussier, François-Xavier Standaert, Vincent Grosso. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. CHES 2016: 61-81 (2016)

39. Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, Marc Witteman. Fast and Memory-Efficient Key Recovery in Side-Channel Attacks. SAC 2015: 310-327 (2015)
40. François-Xavier Standaert, Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. CHES 2008: 411-425 (2008)
41. Youssef Souissi, Maxime Nassar, Sylvain Guilley, Jean-Luc Danger, Florent Flament. First Principal Components Analysis: A New Side Channel Distinguisher. ICISC 2010: 407-419 (2010)
42. Emmanuel Prouff. DPA Attacks and S-Boxes. FSE 2005: 424-441 (2005)
43. Zeyi Liu, Neng Gao, Chenyang Tu, Jian Zhou, Yuan Ma, Yuan Zhao. Leakage Fingerprints: A Non-negligible Vulnerability in Side-Channel Analysis. AsiaCCS 2016: 807-818 (2016)

A Appendix

A.1 Evaluation Results of Our attack from DPA Contest v4.2 [17]

For our experimental results using all sixteen public datasets (80,000 traces in total), the success rate is 100%, while the success rate is about 95% which is evaluated by DPA Contest Official with their private datasets. Although the success rate decreased, the evaluation results still validated the effectiveness of our Multivariate Template Attack (MTA) and Depth-First Key Enumeration Algorithm (DFKEA).

The Global Success Rate (GSR) and the Partial Success Rates (PSR) [8] released by DPA Contest Official are depicted as Fig.8 and Fig.9. For the purpose of comparison, the evaluation results of Partial Guessing Entropy (PGE) are also showed as Fig.10.

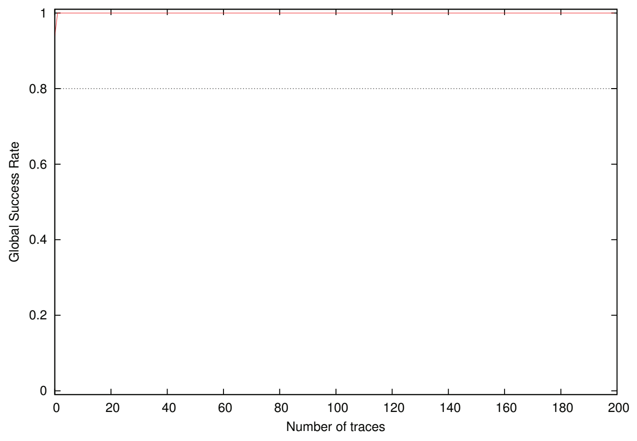
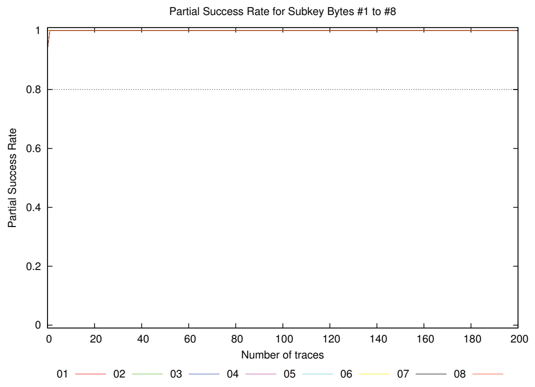
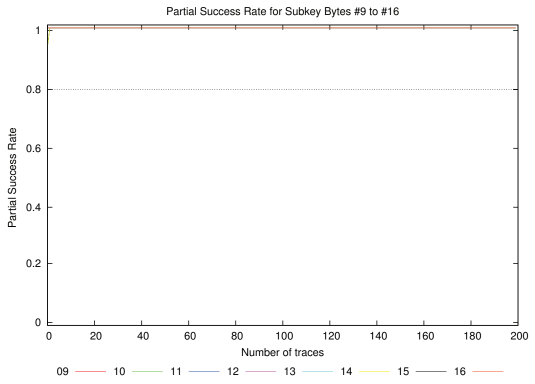


Fig. 8. The Global Success Rate (GSR) of our attack released by DPA Contest Official.

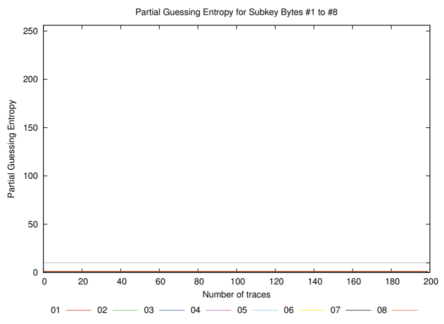


(a) Subkey index from #1 to #8

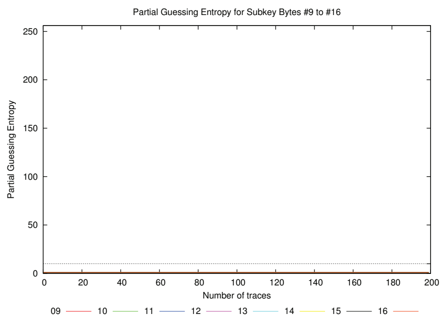


(b) Subkey index from #9 to #16

Fig. 9. The Partial Success Rate (PSR) of our proposed attack released by DPA Contest Official, for all sixteen subkeys.



(a) Subkey index from #1 to #8



(b) Subkey index from #9 to #16

Fig. 10. The Partial Guessing Entropy (PGE) of our proposed attack released by DPA Contest Official, for all sixteen subkeys.