How Far Can We Reach? Breaking RSM-Masked AES-128 Implementation Using Only One Trace

Wei Cheng[1], Chao Zheng[1], Yuchen Cao[1,2], Yongbin Zhou[1,2], Hailong Zhang[1], Sylvain Guilley[3,4] and 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
{chengwei,zhouyongbin}@iie.ac.cn,
{sylvain.guilley,laurent.sauvage}@telecom-paristech.fr

Abstract. Rotating Sbox Masking (RSM) scheme is a lightweight and highly efficient first-order masking scheme proposed to protect cryptographic implementations like AES from side channel attacks. It is a Low Entropy Masking Scheme (LEMS) and has attracted special attention from academia and industry with its low overhead and high performance. The two public targets of DPA Contest v4 are both RSM-masked AES implementations, specifically, AES-256 (namely RSM-AES-256) for v4.1 and AES-128 (namely RSM-AES-128) for v4.2 respectively. The security of RSM-AES-256 was intensively studied by researchers worldwide under the framework of DPA Contest and several flaws were identified. Its improved version is RSM-AES-128, in which several pitfalls of RSM-AES-256 were fixed. However, the practical 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. 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 combining the DFKEA to our MTA, we propose a novel multivariate profiling attack scheme which could recover the secret key of RSM-AES-128 with over 95% possibility only using one trace. It is the best attack among all attacks submitted to DPA Contest Official up to now. After thoroughly analyzed our attack scheme and RSM-AES-128, we finally present two proposals to improve the practical security of this implementation at an acceptable overhead and performance loss.

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

1 Introduction

For several years, Side Channel Attacks (SCA) have put a great threat on practical security of cryptographic implementations, in which an adversary always extracts the sensitive information like secret key by statistic analysis on data-dependent side channel leakages [Koc96, KJJ99, GMO01]. 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 to achieve a high security level. On the other hand, the adversary always has full control on these devices to carry out some very intensive analysis and powerful attacks, especially including some profiling attacks.

To protect cryptographic implementations (devices) against these attacks, many countermeasures and techniques have been proposed, including masking, shuffling and hiding. Specifically, masking schemes [CJRR99, CPR07, MOP07, RP10] eliminate the dependency between sensitive data and leakages by dividing each sensitive variable into several random shares to thwart SCAs. Shuffling schemes [HOM06, RPD09] randomize the order of operations during the execution of cryptographic implementations. Quite differently, by circuit-level alteration, hiding-based countermeasures [CCD00, MOP07, RGN13] 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, which means no requirement on hardware-level alteration. Despite the improvements with respect to SCA security, all the 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) [NSGD12] scheme emerges as a small and efficient countermeasure to provide first-order SCA security, which is a typical implementation of Low Entropy Masking Scheme (LEMS) [MGD11, NGD11, CDGM12]. It has already attracted extensive attentions of researchers from both academia and industry. RSM is also the core protection scheme used in the latest edition of DPA Contest, namely DPA Contest v4 [DPA17]. DPA Contest is an international open framework for worldwide researchers and participants to evaluate and compare their attacks under a common setting [BBD+14]. Particularly, the forth edition of DPA Contest (both v4.1 and v4.2) are launched to analyze and evaluate the practical security of protected software implementations of AES running on an Atmel ATMega-163 smart-card. Specifically, the DPA Contest v4.1 (DPACv4.1) [DPA14] has closed, while the DPA Contest v4.2 (DPACv4.2) is the latest version of contest and still open. The target of DPACv4.1 is a RSM-masked AES-256 implementation, namely RSM-AES-256. During the DPACv4.1, the practical security of RSM-AES-256 was thoroughly studied from both non-profiling attacks and profiling attacks, and several flaws were identified, especially the pitfalls in the RSM scheme [BBD+14, MGH14, YE13] 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 version of RSM-AES-256, in which these pitfalls were fixed [BBD+14] and it is the open target of DPA Contest v4.2 (DPACv4.2). Note that the improved version of RSM scheme in RSM-AES-128 is also called RSM scheme, since we only focus on the improved one, thus there is no ambiguity in denotation.

Intuitively, the combination of different countermeasures could obviously improve the practical security of the cryptographic implementations when they are carefully implemented. This strategy was adopted in DPA Contest v4.2, in which both masking and shuffling are applied to upgrade the first implementation in terms of SCA security. Since several implementation flaws have been fixed (mainly by reprogrammed using assembly language, register precharge and the improved RSM scheme) and a new shuffling scheme is added [BBD+14], RSM-AES-128 is expected to achieve a high SCA security level. Despite the significant improvements made from a perspective of non-profiling setting, the practical security of RSM-AES-128 is still not intensively studied from a perspective of profiling setting, which is worthwhile of further study and analysis. As a result, we focus on security analysis of RSM-AES-128 using profiling attacks [SKS09] in this paper. In fact, attack is always the most straightforward way to evaluate the practical security of cryptographic implementations. Considering the attacking results of RSM-AES-256 implementation during the DPA Contest v4.1 and security-oriented improvements of RSM-AES-128, our final question is that, can we recover the secret key of RSM-AES-128 only using one trace by profiling attacks? If possible, what's the most efficient way to find the secret key by

using only one trace?

From a attacker's perspective, the more exploitable information make it easier to carry out a successful attack. Under the framework of DPA Contest, all necessary information and parameters including details of implementation, parameters of countermeasures and open traces are available to global participants. For attacks against RSM-AES-128, three main tools and observations which could be exploited by attackers are as follows.

- **Onion-Peeling Strategy**. Onion-Peeling strategy is type of Divide-and-Conquer strategy widely applied in SCA. Although the combining of RSM scheme and shuffling scheme provide RSM-AES-128 a significant security improvement, the attackers still can use the onion-peeling strategy to break the masking scheme and shuffling scheme independently. This strategy also can be applied to attack against RSM-AES-256 implementation since the details of the implementation is open to researchers. Once the combined countermeasure is deactivated, the RSM-AES-128 implementation becomes an unprotected software implementation [NSGD12], which is much vulnerability to side channel attacks, especially for template attack.

- **Template Attack (TA)**. Template attack is broadly accepted as the strongest form of side channel attack from a perspective of information theory [CRR02]. The framework of DPA Contest v4.2 provides sufficient information and datasets for constructing accurate templates and cross-validations. Considering that our goal is to recover secret key of RSM-AES-128 using only one trace, TA is a natural better choice compared to other profiling attacks like Linear-Regression based attacks. Furthermore, a critical observation is that the sensitive variables always take part in the cryptographic operations more than once. For example with AES encryption, the inputs of Sbox layer come from the KeyAdd layer, meanwhile the outputs of Sbox layer feed into the ShiftRows layer, etc. Therefore the multivariate TA which exploits the leakages of multiple sensitive variables [BGNT15] (similarly the Multi-target DPA attacks [MOW14]) is even more powerful attacks compared to univariate TA.

- **Key Enumeration Algorithms (KEA)**. Considering the full procedure of side channel attacks, KEA is a powerful post-processing technique to find the correct key effectively after an attack. Recently, several KEA [VGRS12, BKM+15, PSG16] have been proposed to optimize their the effectiveness, efficiency, memory overhead and parallelization. Particularly, the optimal KEA [VGRS12] provides an optimal order to enumerate the results after an attack but with large memory overhead in terms of "key trals", while the Histograms-based KEA [PSG16] leads to straightforward parallelization with simple bounds of rounding errors. These KEA methods generally assume that different chunks of subkey candidates are similarly distributed, therefore they are breadth-first methods and adopt a combine-then-verify route. Specifically, all attack results of different key chunks are firstly combined according to their possibilities (or scores), and then to verify each combined key candidates ordered with joint possibility (or overall scores). However, if the distributions of different chunks of key candidates vary from each other, existing KEA method would be less efficient since more key verifications are inevitable for finding the secret key.

Considering all information and tools could obtain for an attacker under the framework of DPA Contest v4.2, we are especially interested in analyzing and evaluating the practical security of RSM-AES-128 by trying to recover its secret key only using one trace.

Our Contributions. Keeping in mind that our goal is to recover the secret key of RSM-AES-128 in a simple but efficient way with only one trace, our contributions mainly are threefold as follows.

1. **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 attacks against multiple sensitive variables corresponding to the same subkey chunk (byte), while the dimensionality is the number of interesting points used in our attack. We also use Principle Component Analysis (PCA) [SA08, SNG+10] as a leakage-feature extraction tool to reduce the dimensionality of our MTA, thus to dramatically decrease the data complexity of templates. By applying onion-peeling strategy and several optimization, our MTA can break the RSM scheme and shuffling scheme with 100% probability using one trace, meanwhile the partial success rates (PSR, corresponding to recovery of each key byte) of attack varies in a range from 93% to 98%, and the global success rate (GSR, corresponding to the recovery of entire secret key) is increased from 55% to 83% which increased 50.91% (using one trace).

2. **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 the side channel attacks. 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 eliminates 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, especially when the electromagnetic traces used as leakages. Based on some specific observations, DFKEA adopts a (bounded) depth-first approach to traverse the most possible subkey hypothesis quickly. By integrating DFKEA into our attack scheme, its success rate (SR) is significantly increased from 83% to 95% (increased 14.46%, evaluated by DPA Contest Official) with an acceptable computation overhead. These results distinctly indicated that RSM-AES-128 is vulnerable to profiling attacks, especially for MTA.

3. **Fully Shuffled Round Transformation (FSRT)**. Countermeasures are requisite to thwart SCA. Since the shuffling scheme of RSM-AES-128 is only applied to protect the Sbox layer [BBD+14], other three layers including *KeyAdd* layer, *Shiftrows* layer and *MixColumns* layer are not shuffled to hinder SCA. As a consequence, the shuffling scheme could be bypassed easily. One notable observation existed in the RSM-AES-128 is that all four layer's operation could be shuffled. Therefore, we propose the full-round shuffling scheme to further improve the SCA security of RSM-AES-128 with negligible overhead. We also propose an idea 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 implementation and then template attacks together with some notations used throughout this paper. Section 3 explains the rationale of multivariate template attack and its application to break the RSM scheme and shuffling scheme and to conduct the key-recovery attack. In section 4, the new key enumeration algorithm DFKEA is proposed with necessary validation experiments. Next, all ideas to further improve the practical security of RSM-AES-128 are described in section 5. Finally, conclusions are drawn in section 6 and some evaluation results from DPA Contest Official are in Appendix.

2 Preliminary and Notations

2.1 RSM-AES-128 Implementation

RSM-AES-128 [BBD+14] is a software implementation of AES-128 [DR02] simultaneously protected by 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

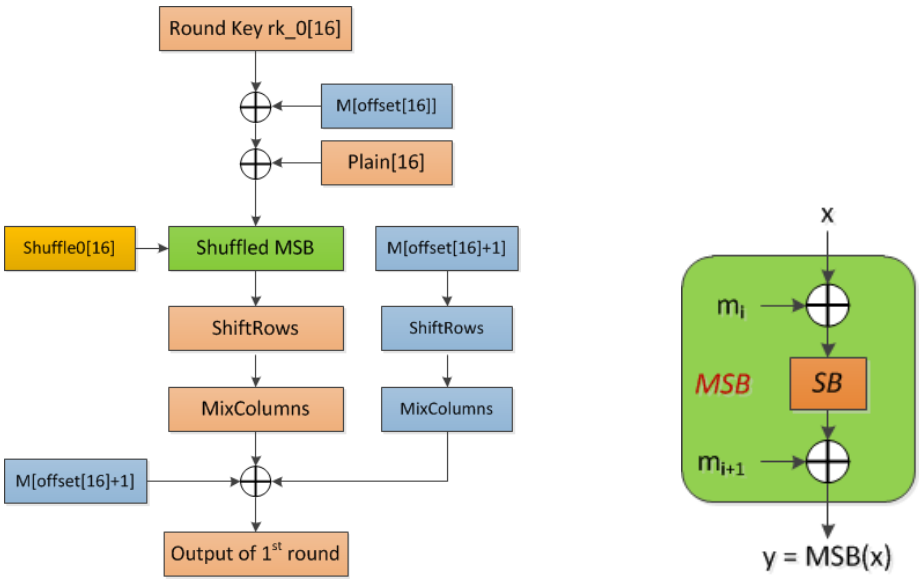
in the Sbox layer of the first and last rounds of transformation to protect the commonly "vulnerable" part of implementation [Pro05].

RSM Scheme. The well-designed masking and unmasking method makes RSM very efficient. Essentially, RSM scheme is a low entropy boolean masking scheme, the mask set only contains 16 fixed mask values which are selected in advance as follows.

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

where m_i and M_i 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. In AES, its round function consists of four subfunction, namely *KeyAdd (AK)*, *SubBytes (SB)*, *ShiftRows (SR)* and *MixColumns (MC)*. Therefore all these four subfunctions must be protected by RSM. Note that *SubBytes* is the only nonlinear part in AES, while the other three are linear transformations, in which the RSM could be applied straightforwardly. The first round structure of RSM-masked implementation RSM-AES-128 is depicted as Fig.1(a). Specifically, sixteen masks are Xored with first round key *rk_0* and then the plaintext are involved, followed by the *masked SubBytes* with shuffling. Next, *ShiftRows (SR)* and *MixColumns (MC)* operated on masked intermediates, finally the mask compensation *MaskComp* are used to unmasking and re-masking intermediates with proper masks.



(a) (b)

Figure 1: The overall structure of first round in RSM-AES-128 (a) and schematic of Masked SubBytes (MSB) (b)

Let SB, MSB denote the original and masked *SubBytes* in AES respectively, and $x \in \mathbb{F}_2^n$ ($n = 8$ in AES) is an intermediate variable. Note that the state matrix and operations in AES are byte-oriented, all intermediate values are elements from $\mathbb{F}_2^8$. Then the RSM-masked Sbox is (as depicted in Fig.1(b))

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

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 different Sbox and determined by an

offset vector, namely $\overrightarrow{offset}$, to guarantee that all sixteen Sboxes in RSM-AES-128 are masked independently.

Finally, the mask compensations *MaskComp* are applied to ensure that all intermediates are masked properly before getting into next round.

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

where r is the round index of AES.

Shuffling Scheme. The shuffling scheme only adopted to protect the Sbox layer (*SubBytes*) in the first and last rounds of RSM-AES-128, using $Shuffle0$ and $Shuffle10$ respectively. It is a 4-bit based permutation applied to change the order of sixteen Sboxes.

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

2.2 Template Attacks

TA are the strongest form of side channel attacks in an information theoretic sense, which assume that the adversary can fully characterize the leakage features from an reference device and then conduct the key-recovery attack on target device [CRR02, SKS09]. They are two-phase attacks and consist of profiling phase and attacking phase.

Let k^*, k denote the secret key and any possible key hypothesises respectively, and T is the plaintext or ciphertext. We suppose that all computations are done on n-bit chunks, namely all intermediates x are element 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

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

where $\mathcal{L}(\cdot)$ denotes the data-dependent leakage function and $Noise$ is the independent noise (as commonly assumed). Particularly, $L = [L_1, L_2, \ldots, L_D]$ denotes multiple points-of-interest (PoI) in a leakage trace and D is the data dimensionality (the number of PoIs). Thus a set of N measured traces denote as $\mathbf{L} = \{L^1, L^2, \ldots, L^N\}$.

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 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-grouped 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 traces measured from the targeted device and Q for number of 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, namely number of PoIs in this context.

Finally, with Maximum Likelihood (ML) principle, the actual value of sensitive intermediate v is indicated by maximal $p_{i,v}$.

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

Note that v is key-dependent intermediate, resulting that the secret key (bytes) could be inferred immediately 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, SCAs 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 connected intermediates in sequence.

3 Multivariate Template Attack against RSM-AES-128

Under the framework of DPA Contest v4.2, attackers are allowed to access all information and datasets including the implementation details, cryptographic parameters and measured electromagnetic traces. Therefore, TAs are practical and in order to recover the secret key of RSM-AES-128 by one trace, we propose a Multivariate Template Attack (MTA) in this section, 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 trace, while the latter is mainly adopted for leakage-features extraction on measured traces rather than data-dimension reduction. In practical, PCA works because data-dependent signals are the dominant part in the measured traces, especially true for selected PoIs.

The Onion-peeling strategy is the core to our entire attack against RSM-AES-128, since it is protected by combined countermeasure consisting of RSM scheme and shuffling scheme. Therefore, 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 of the attack at a practical tractable cost. Note that all operations in RSM-AES-128 are byte-oriented, therefore we follow a byte-oriented descriptions in this paper for the sake of brevity.

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 $V_{i,j}, j \in [0, 1, \ldots, 15]$ are j-th byte of these intermediates, and $v_{i,j}, j \in [0, 1, \ldots, 15]$ is the instance of $V_{i,j}$, similar denotation for other variables.

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

where the same as usual, the 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 $\overrightarrow{offset}$ consisting of sixteen elements, namely $\overrightarrow{offset} = \{s_0, s_1, \ldots, s_{15}\}$. In the first round of encryption, all masks (indexed by s_i) are involved into the encryption process. As a consequence, our TA targets all M_{s_i} in the first round for simplicity. However, it's similar to carry out TA targeting on last round of RSM-AES-128. The leakage features of masks in first two byte-positions of RSM-AES-128 are depicted as Fig.2.

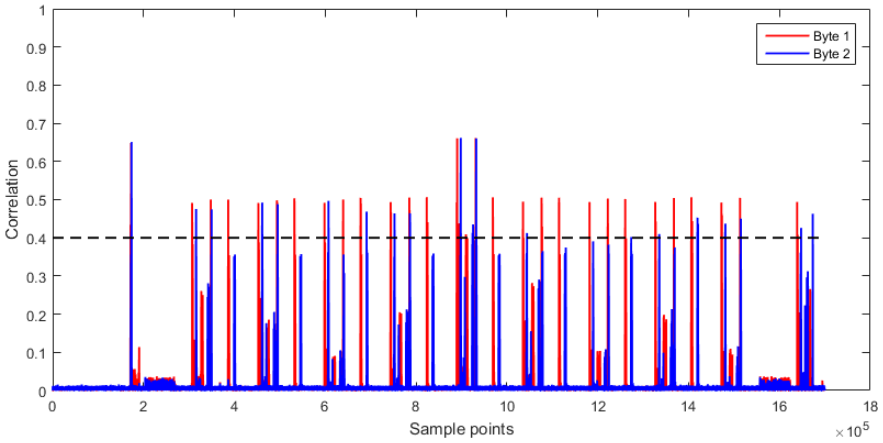


Figure 2: The leakage of masks in the first two bytes of RSM-AES-128 (HW, Correlation)

Therefore, 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 $\overrightarrow{offset}$. As mentioned, we use PCA to extract device-specific leakage features. Specifically, for each $V_{1,j}, j \in [0, 1, \ldots, 15]$, leakages selected from about 1200 PoIs are integrated together, and then only top 10 components corresponding to first 10 eigenvalue (data-dimensions are reduced from $D = 1200$ to $D = 10$) are further chosen to build each template $(\mu_{V_{1,j}}, \Sigma_{V_{1,j}})$ using Equ.6. In attacking phase, for each new trace treated by PCA, Equ.7 and Equ.8 are applied to determine the most possible mask (or $s_i, i \in [0, 1, \ldots, 15]$) hypothesis.

Our experimental results validated the effectiveness of our PCA-based TA against RSM scheme. The success rate for recovering all masks (or $s_i, i \in [0, 1, \ldots, 15]$) are 100%. One main reason is that all masks take part in almost all cryptographic operations including Xored with subkey bytes, then Xored with plains and other intermediates, thus from an attacker's point of view, these operations leak enough information to recover all masks.

Attacking Shuffling scheme. The attack procedure 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. As a consequence, in order to recover the shuffling vector $Shuffle0$ by one trace, different number of PoIs are used for each element of $Shuffle0$ (as Tab.1).

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

At this point, the combined countermeasure is compromised by our PCA-based TA, which validates the effectiveness of our attack. Particularly, our experiments also validate the effectiveness of applying PCA and high-dimension TA to break countermeasures in a very efficient way (especially from a engineering perspective). However, RSM-AES-128 has become an unprotected cryptographic implementation once both countermeasures are compromised, hence we can carry out the second step of Onion-Peeling strategy to recover

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

the secret key.

3.2 Our Multivariate Template Attack

We propose a Multivariate Template Attack (MTA) to obtain the secret key in an effective way. Here multivariate means that multiple key-dependent variables are targeted to co-contribute to obtain subkey candidate ranks. Specifically, our MTA targets three sensitive variables depending on each of key bytes. There three variables are input of Sbox ($V_{3,j} = M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j$), output of Sbox ($V_{4,j} = MSB(M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j)$) and input of Mixcolumns ($V_{5,j} = MSB(M_{\overrightarrow{offset}[j]} \oplus K_j \oplus T_j)$, the *ShiftRows* is ignored since it only changes the positions of different bytes while keep their values unchanged), all of which depend on $K_j, j \in [0, 1, \ldots, 15]$. Importantly note that our MTA combines the attacking results after three univariate-oriented TAs rather than combining leakages in measured traces before the specific attack (similar idea applied in [MOW14] named Multi-target attacks but in a non-profiling setting).

The leakage features of first byte of these two sensitive variables are exampled as follows (selected samples of the first 50,000 points in measured traces).

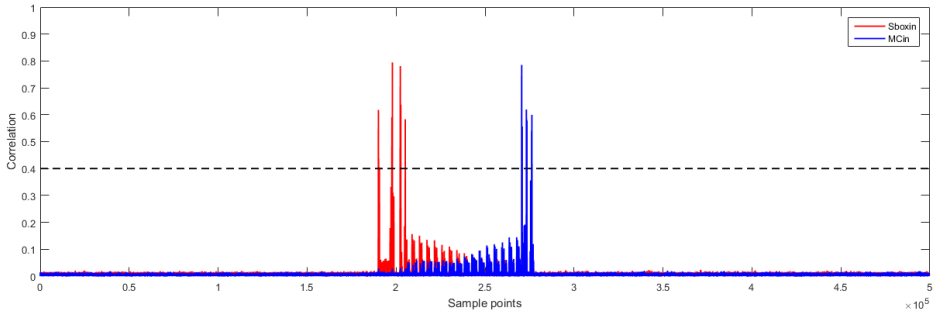


Figure 3: The leakage of Sboxin and MCin in first byte of RSM-AES-128 (HW, Correlation)

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. As aforementioned, 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 $Shuffle0$ to optimize the key-recovery success rates. The rough number of chosen components are listed in Tab.2.

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

Table 2: Number of chosen components after PCA for $V_{3,j}, V_{4,j}$ and $V_{5,j}$ $(j \in [0, \ldots, 15])$

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

their possibilities. Since the masks and shuffles have 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 success rates (PSR) of each byte are computed by cumulatively summed success rates. Therefore, the PSR of $V_{3,j}, V_{4,j}$, $V_{5,j}$ $(j \in [0, \ldots, 15])$ and our combined attack (MTA) are plotted in Fig.4 as follows.

Particularly, for our MTA, Fig.4(d) shows that the first-order PSR of all subkey bytes are over 98%, except fourteenth subkey byte for 93%. In addition, we can observe that,

1. PSR of all subkey candidates follow a extreme Pareto distribution (or "80-20 rule"), 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 one candidate to be the correct subkey hypothesis.

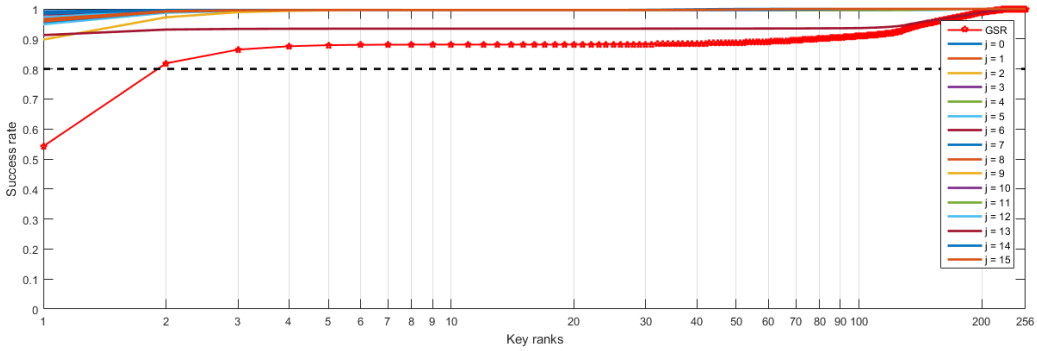
2. More importantly, compared to univariate TA, MTA makes the correct subkey hypothesis approach to the highest ranks among all subkey candidates, even for the worst subkey byte (fourteenth subkey byte, namely $j = 13$). This approaching effect also validates the positive impact of our MTA on revealing the secret key. Consequently, the GSR of MTA is significantly higher than all three univariate TAs.

3. Different positions (indexes) of bytes have small effect on attacking results except fourteenth subkeys, which is the worst byte of all subkey bytes, no matter which sensitive variables are targeted. However, this "information" could be exploited to improve the success rates by post-processing techniques, especially to improve GSR.

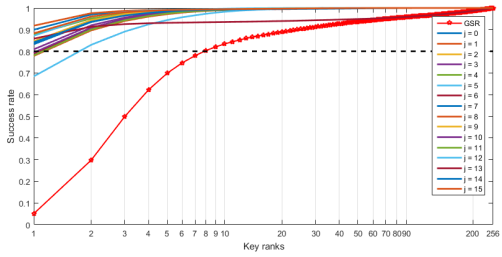
In practice, it's advantageous to make use of information related to subkey rank distributions when maximizing the global success rate (GSR) with constrained computational complexity. In next section, we propose the bounded Depth-First Key Enumeration Algorithm (DFKEA) to dramatically improve the GSR of MTA by exploiting these distribution information.

4 Depth-First Key Enumeration Algorithm

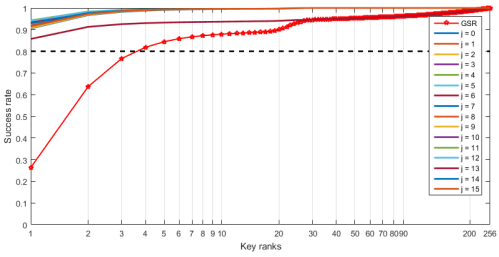
Our major interest is to recover the secret key of RSM-AES-128 in a very efficient way, thus two main requirements for post-processing methods are the high efficiency of key-finding and the maximal success rates. On one hand, the former requires the minimal number of key verifications and less extra computations, like computations caused by combining all sixteen sorted subkey lists together to obtain global key sorting results [VGRS12, BKM+15, PSG16]. On the other hand, the latter requires a high coverage of the most possible subkey candidates.



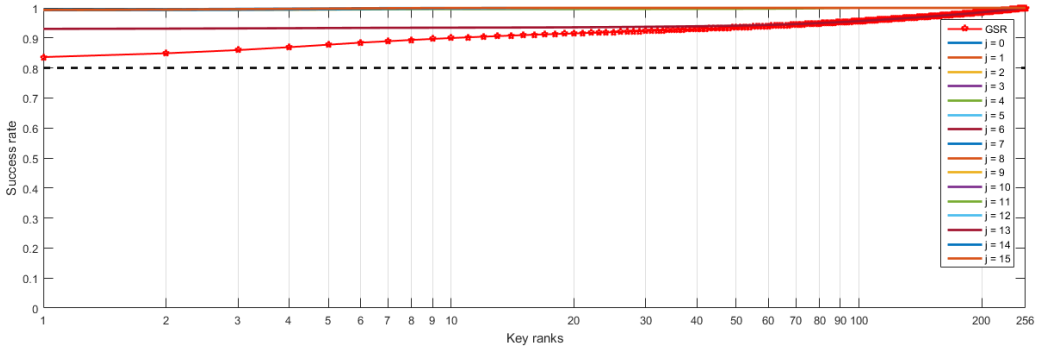
(a) $V_{3,j}$ (Sboxin).



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



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

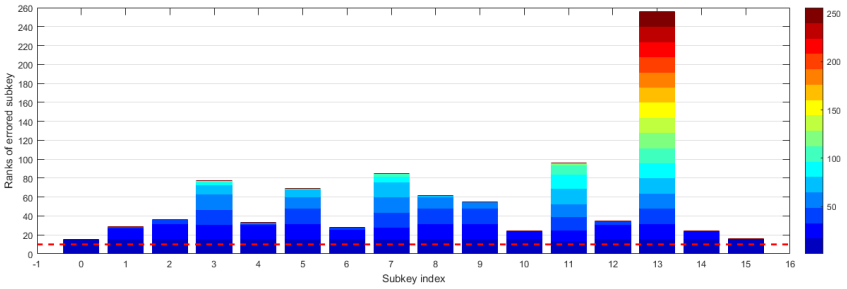


(d) Our combined TA (MTA).

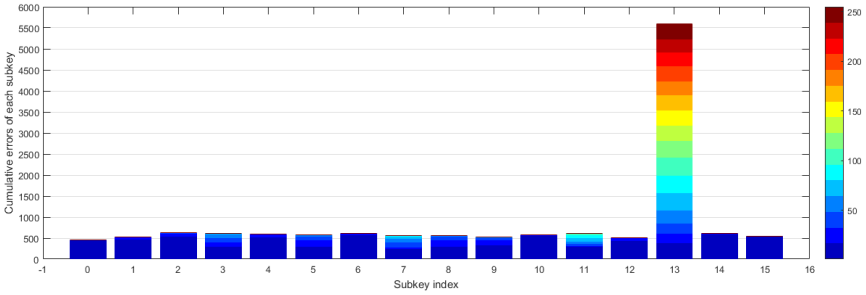
Figure 4: Partial Success Rates (PSR) and Global Success Rate (GSR) of key recoverys targeted on different variables (with logarithmic X-axis). Particularly, (a). $V_{3,j}$ (Sboxin), (b). $V_{4,j}$ (Sboxout), (c). $V_{5,j}$ (MCin) and (d). our combined TA (MTA) for each j-th subkey byte, $j \in [0, \ldots, 15]$, only one trace is used in all four cases.

Keeping these requirements in mind, we firstly investigate the distribution of errored subkey candidates. Here "errored" subkey means its correct subkey candidate is not ranked first among all possible candidates. The ranks of errored subkeys and cumulative errors of each subkey are depicted as Fig.5. Note that all 80,000 traces from DPA Contest v4.2 are used in our experiments unless explicit stated. Unsurprisingly, same as observed in Sec.3.2, 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 subkey bytes (N_{err}) per attack (or per trace) as showed in Fig.6. By using 25,000 to 40,000 traces, two main observations are $N_{err} \leq 4$ and number of attacks corresponding to N_{err} significantly decreases with the increase of N_{err}. Hence, if we take N_{err} into consideration, which means that if we enumerate all possible subkey



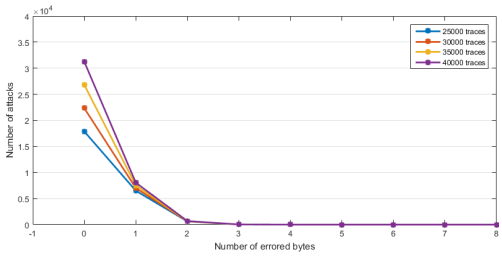
(a) The ranks of errored subkeys.



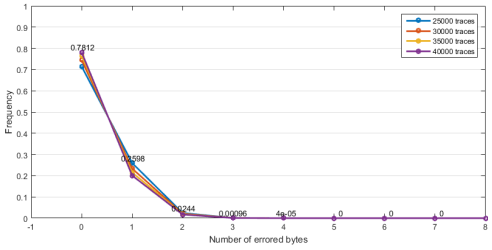
(b) Cumulative errors of each subkey.

Figure 5: The distribution of errored subkey candidates, "errored" means failed to recover a subkey by first-order rank.

candidates along with the increase of N_{err}, the most possible subkey could be covered. As a consequence, 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 in a very efficient way.



(a) Number of attacks corresponding to N_{err}



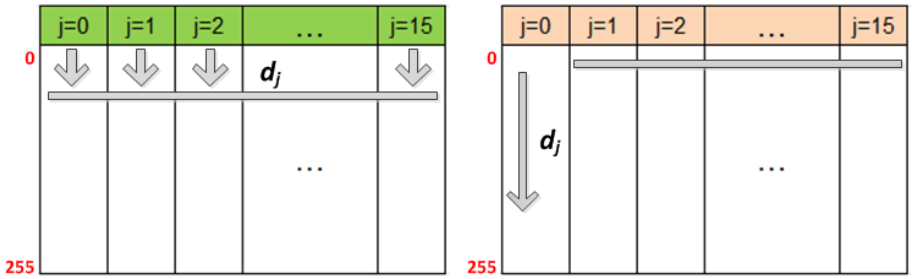
(b) Frequency

Figure 6: The distribution of number of errored subkeys (N_{err}) with number of traces from 25,000 to 40,000 (first eight datasets from Website of DPA Contest v4.2).

4.1 Proposal of DFKEA

Main idea behind the DFKEA is to enumerate all possible subkey candidates along with increase of N_{err} in a depth-first way. Specifically, we firstly enumerate and verify the subkey candidates with $N_{err} = 0$, which means to enumerate subkey candidates with the highest rank. Next, subkey candidates with $N_{err} = 1$ are enumerated, that is to test all subkey candidates in which only one subkey is not ranked first (while other fifteen correct subkey candidates ranked first). Similarly, subkey candidates with N_{err} from 2 to 15 can

be enumerated and then verified. Particularly, considering aforementioned observations, our key enumerations could be efficiently done by restricting $N_{err} \leq 4$.



(a) state-of-the-art KEA (b) Our DFKEA with $N_{err} = 1$

Figure 7: Main enumeration strategy of the state-of-the-art KEA vs our DFKEA.

Let d_j denotes the enumeration depth of j-th subkey byte enumerated by our DFKEA for $j \in [0, \ldots, 15]$, N_{err} is as aforementioned and then $d_{j,n_{err}}$ for d_j with $N_{err} = n_{err}$. Note that we have different $d_{j,n_{err}}$ according to varied n_{err} (namely bounded depths). On the basis of observations from Fig.6, the schematic of our DFKEA with $N_{err} = 1$ is depicted as Fig.7(b). In order to have a comparison point, principles behind of the existing state-of-the-art KEA methods are breadth-first strategy as illustrated in Fig.7(a), in which all sixteen subkeys candidates are enumerated with an approximately equal enumeration 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.

Our algorithm of DFKEA is described as Alg.1. Note that convertKey(c_i) in line 10 converts each possible combination c_i to the errored subkey indexes by which these errored subkeys will be enumerated later. However, tradeoffs are made between the efficiency and success rate, resulting that all possible subkey combinations are enumerated with bounded enumeration depth $d_{j,n_{err}}$.

In practice, the efficiency is straightforwardly determined by the number of enumerations (or key verifications). For our DFKEA, we assume that the enumeration depths $d_{j,n_{err}}$ keep the same for each subkey indexed by $j \in [0, \ldots, 15]$, 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 subkey 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 average number of total enumerations $\overline{EC}_{total}$ can be computed as follows.

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

where $\leqslant$ is used because our DFKEA will stop once the correct key is found, and $d_{j,n_{err}}$ keep the same for $j \in [0, \ldots, 15]$. In particular, $n_{err} = 0$ means all correct subkeys are ranked first thus no errored byte occurred. 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

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

Input : Sixteen lists of possible subkey candidates, $skCandi[16][256]$, N_{err}, $d_{j,n_{err}}[]$, plaintexts $P[][16]$ and ciphertexts $C[][16]$.
Output: Secret key or the most possible subkey candidates $sk[16]$.

$Flag = false$
if $true ==$ KeyVerification($P[][16]$, $C[][16]$, $skCandi[16][1]$) **then**
  Flag = $true$
  return $sk[16] = skCandi[16][1]$
else
  for $n_{err} \in [0, N_{err}]$ **do**
    $comb[n_{err}] = C_{16}^{n_{err}}$ /* C_m^k *is the combination formula* */
    for $c_i \in comb[n_{err}]$ **do**
      /*$convertKey(c_i)$ *converts* c_i *to subkey indexes for enumeration* */
      $keyInd$ = **convertKey**(c_i)
      for $key \in skCandi[keyInd][d_{j,n_{err}}[]]$ **do**
        $Flag$ = KeyVerification($P[][16]$, $C[][16]$, key)
        if $true == Flag$ **then**
          return $sk[16] = key$
        end if
      end for
    end for
  end for
end if
if $false == Flag$ **then**
  /*Deal with failures on finding the correct key */
  return $sk[16] = skCandi[16][1]$ /* return the subkeys ranked first */
end if

increased $n_{err} = 3$, we set $d_{j,n_{err}} = d_{j,3} = 10$, thus $EC_{n_{err}} = EC_3 = C_{16}^3 * (10-1)^3 \approx 2^{18.64}$. 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 analysis of an algorithm is a very important issue when it's applied in practice. Here, we primarily take the number of enumerations EC into account, which is also the number of key-verification. The computational complexity (number of enumerations) comparison of our DFKEA with a simple KEA is tabled as Tab.3. Importantly note that the enumeration principles of this simple KEA are similar to the state-of-the-art KEA, since they all take a breadth-first strategy.

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 half part is on the contrary. Obviously, compared to this breadth-first KEA, our DFKEA are much powerful on enumerating subkey candidates with high enumeration depths. Although we always theoretically assume that attacks against different subkeys would obtain similar distribution for different subkey candidates, in practice, attacking results of different subkeys always varies from each other, especially when attacking electromagnetic traces (leakages). Hence, our DFKEA is very suitable for these attacking scenarios.

With the same 40,000 traces (the same as used in Fig.6), we practical evaluate the effectiveness of our DFKEA. Considering the high efficiency requirement of the attacks, 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

Table 3: Comparison of our DFKEA and a simple KEA with EC as evaluation criteria

Depth $d_{j,n_{err}}$	DFKEA using (Equ.10)				existing 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	$2^{11.99}$	$2^{22.90}$	$2^{33.11}$	$2^{33.11}$	$256^{16} = 2^{128.00}$

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.8. 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 simple KEA for $d_1 = 2$ and $d_2 = 5$, respectively. These results are also in very accordance with our observations from Fig.6 that the number of errored subkeys dropped off sharply with the increase of n_{err}, thus it's very advantageous to adopt 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 part of other subkey candidates would be verified after a typical attack to obtain a high coverage of possible subkey hypothesises.

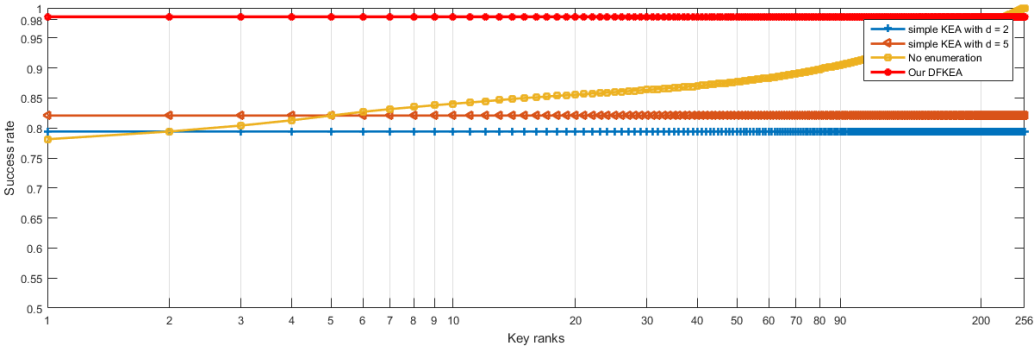


Figure 8: Comparison between our DFKEA and the simple KEA with respect to different enumeration depth(with logarithmic X-axis).

To summarize up, these experimental results strongly validated the effectiveness of our DFKEA in terms of both key-recovery efficiency and attacking success rates (also equivalent to guessing entropy). More importantly, compared to the state-of-the-art KEA, our DFKEA doesn't require to combine all sixteen lists of subkey candidates to get a global key-ranking list[PSG16], which in return reduces computational burden and makes our DFKEA much more efficient and competitive than existing KEAs. Furthermore, our DFKEA is also generally applicable to both profiling and non-profiling attacks in SCA.

5 Ideas for Further Improving RSM-AES-128

RSM scheme is a first-order masking scheme which featured with its high efficiency and low overhead, even integrated with shuffling scheme as implemented 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 are not immune to profiling attacks like TAs and leakage "fingerprints" matching methods [LGT+16]. 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 protected implementations) is to reduce or even eliminate these distinguishable leakage features.

Based on our observations and experimental results of MTA and DFKEA, we propose two ideas to improve the practical security of RSM-AES-128, which are concentrating on the shuffling scheme and RSM scheme adopted in RSM-AES-128, respectively.

5.1 Fully Shuffled Round Transformation

From an protecting point of view, shuffling scheme plays a important role in thwarting SCAs. In essence, shuffling scheme only randomizes the order of operations during the execution of cryptographic implementations, while no alteration on intermediate values. Considering the combined countermeasure applied in RSM-AES-128, the first major vulnerability is that shuffling scheme was only adopted in Sbox layer ($SubBytes$) of AES, which means the other three transformations ($KeyAdd$, $ShiftRows$ and $MixColumns$) are not protected by it. As a consequence, the call order of all key-dependent variables only shuffled in Sbox layer but kept the same during operating other three transformations. Typical leakages of inputs of Sbox and inputs of $MixColumns$ are depicted as Fig.3 (with the first byte). This property was thoroughly exploited by our MTA, which not only targeted on outputs of Sboxes, but also aimed at inputs of Sboxes (also the output of $KeyAdd$) and inputs of $MixColumns$.

The main idea to settle this vulnerability is to apply shuffling scheme into all four transformations of one round, namely Fully Shuffled Round Transformation (FSRT). Importantly note that shuffling scheme only changes the order of operations, but keeps the value of operands unchanged. Specifically, in FSRT, the shuffled Sboxes are the same as in RSM-AES-128, and it's similar for $KeyAdd$ and $ShiftRows$ since these transformations are byte-oriented and operations are independent between different bytes. Nevertheless, the $MixColumns$ is complex because of its design requirements for diffusion effect. One naive option is to update all sixteen bytes of state matrix one by one. The shuffle vectors used in each of four transformation could be the same one like in RSM-AES-128 but updated for different rounds. For the sake of simplicity, here we use the same shuffle vector in these four transformations. The algorithm is showed as Alg.2.

Note that the secret key are firstly Xored with masks $M_{\overrightarrow{offset}+sh[0]}$ before feeded into the first round of AES. The $MixColumns$ is also removed in the last round and the unmasking of cipher is similar to RSM-AES-128 (referred to Alg.3). Finally, the entire improved version of RSM-AES-128 is obtained by concatenating N_r rounds ($N_r = 10$ for AES-128) together.

In practice, $MixColumns$ always implemented using **gf256mul** function [DR02] to reduce the redundant computations. Therefore, the shuffled $MixColumns$ in Alg.2 (line 11 to 14) could be optimized by shuffling between four columns.

5.2 New Further Improved RSM Scheme

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 with the same indexes. From a 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 each of sixteen

Algorithm 2 Fully shuffled round transformation

Input : Input state matrix $State[16]$, round index r, and Round key $rk[16]$, Shuffle vector $Shuffle[16]$, Offset $offset[16]$ for masking.
Output: Updated state matrix $State[16]$.

for $i \in Shuffle([0,15])$ **do**
  $State[i] = State[i] \oplus rk[i]$ /* *Shuffled KeyAdd* */
end for
for $i \in Shuffle([0,15])$ **do**
  $State[i] = MaskedSB_{offset[i]+sh[r-1]+(r-1)}(State[i])$ /*New Shuffled SubBytes*/
end for
tmp $= State[16]$ /* *Shuffled ShiftRows* */
for $i \in Shuffle([0,15])$ **do**
  $State[SR(i)] =$ tmp$[i]$
end for
tmp $= State[16]$ /* *Shuffled MixColumns* */
for $i \in Shuffle([0,15])$ **do**
  $State[i] = MC($tmp$)$ /* *Column-oriented operations* */
end for
/* *New MaskCompensation refreshed by* $sh[r]$*, where* $sh[r] = Shuffle[r]$ */
for $i \in [0,15]$ **do**
  $MaskComp[i] = MC(SR(Mask[offset[i] + sh[r-1] + r]))$
  $\oplus Mask[(offset[i] + sh[r] + r)]$
end for
$State[] = State[] \oplus MaskComp[]$

offsets [LGT+16]. The leakages of masks in the first two bytes and first four bytes are showed in Fig.9.

In Fig.9(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 by attackers to reveal the first two offsets. Similarly, all sixteen offsets could be revealed. In Fig.9(b), one main observation is that leakages of masks in the first four bytes clearly occur in sequential order, which indicates the necessity of shuffling scheme in all operations, in which our proposal in Sec.5.1 could be applied.

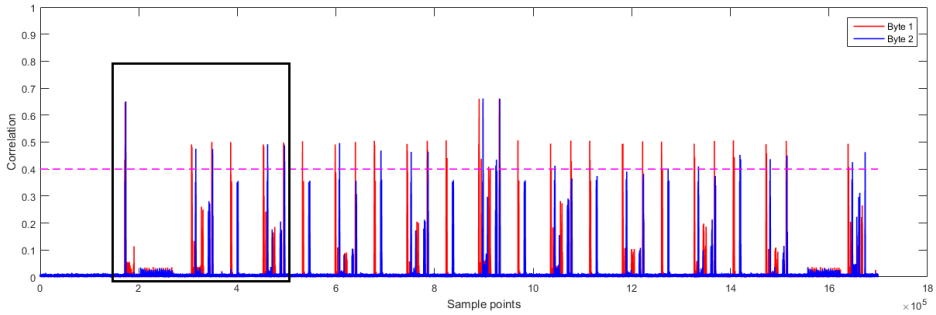
In order to eliminate the leakages of offsets during whole encryption (decryption) process, we propose a new mask compensation method for RSM scheme. The rationale of new $MaskComp(\cdot)$ is to update all offsets for every rounds of encryption. Considering the implementing overhead and performance loss caused by changes, we make minimal but effective modifications to achieve our goal. 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.

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

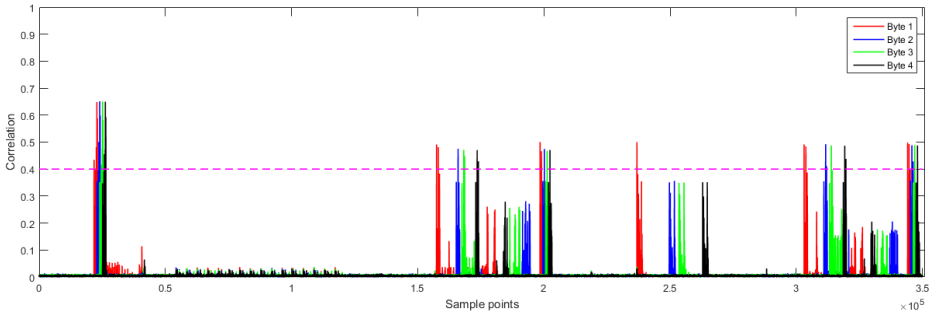
where $MSB'(\cdot)$ denotes the new masked sboxes. The mask compensation then updated as follows.

$$\begin{aligned} NewMaskComp_{i,r} &= \begin{cases} M_{\overrightarrow{offset}_{out}+r} \oplus MC(SR(M_{\overrightarrow{offset}_{in}+r})), & r = 1,...,9 \\ SR(M_{\overrightarrow{offset}_{in}+r}), & r = 10 \end{cases} \\ \overrightarrow{offset}_{out} &= P(\overrightarrow{offset}_{in}) \end{aligned} \tag{12}$$

where r is the round index of AES, and $P(\cdot) : \{0,\ldots,15\} \to \{0,\ldots,15\}$ denotes a simple (random) permutation, which could be implemented by shuffling to update the masks used



(a) Correlation on whole traces with 1,700,000 points



(b) Zoomed in of samples from 150,000 to 450,000

Figure 9: Leakages of (a) masks used in the first two bytes and (b) masks used in the first four bytes respectively (HW, Correlation, with 40,000 traces).

in different rounds. A straightforward way to implement $P(\cdot)$ is to using element-oriented cyclic shift like $ShiftRows$ in AES, which means $\overrightarrow{offset}_{out}$ is generated by cyclicly shifting $\overrightarrow{offset}_{in}$. In order to improve efficiency, we assign $\overrightarrow{offset}_{in} = \overrightarrow{offset} + sh[r-1]$ and $\overrightarrow{offset}_{out} = \overrightarrow{offset} + sh[r]$, where $sh[r]$ is the r-th element of shuffle vector $Shuffle[16]$. Then the mask compensation could be done easily as follows.

$$MSB'(X) = SB(X \oplus M_{\overrightarrow{offset}+sh[r-1]+(r-1)}) \oplus M_{\overrightarrow{offset}+sh[r-1]+r}$$
$$MaskComp_{i,r} = \begin{cases} M_{\overrightarrow{offset}+sh[r]+r} \oplus MC(SR(M_{\overrightarrow{offset}+sh[r-1]+r})), & r = 1, ..., 9 \\ SR(M_{\overrightarrow{offset}+sh[r-1]+r}), & r = 10 \end{cases} \quad (13)$$

Note that this proposal is also implemented in Alg.2.

More importantly, our new improved RSM scheme is more secure than RSM scheme applied in RSM-AES-128, since it would be degraded to the original RSM scheme if the shuffle vector $\overrightarrow{Shuffle}$ is compromised or deactivated by adversaries (in the worst cases). By adopting independent $\overrightarrow{offset}$ in adjacent rounds, attacks exploiting multi-round leakages [LGT+16] to against RSM scheme could be effectively hindered.

Apparently, both of our proposals for improving the RSM-AES-128 are compatibility with other proposals like new ordered mask set [VG17] to provide a high level of practical security for cryptographic implementations. In addition, these proposals could be implemented on the other platforms like FPGA integrated with typical hardware-oriented countermeasures (e.g. random delays [MOP07]).

6 Conclusions and Perspectives

As a highly efficient and lightweight Low Entropy Masking Scheme, Rotating Sbox Masking (RSM) scheme is proposed to protect cryptographic implementations like AES from side channel attacks. The public target of DPA Contest v4.2 (the latest version) is a RSM-masked implementation called RSM-AES-128. It is protected by a combined countermeasure composed of RSM scheme and Shuffling scheme. Although RSM-AES-128 is the improved version of the previous RSM-masked implementation, its practical security are not intensively studied, especially from a perspective of profiling attack with minimal traces. In this paper, by means of profiling attacks, we have thoroughly analyzed and evaluated the practical security of RSM-AES-128 implementation in an extreme condition, in which only one trace is used for attack. Specifically, under the framework of DPA Contest v4.2, we firstly propose a Multivariate Template Attack (MTA) against RSM-AES-128. Considering global success rate (GSR) as evaluation criteria, our MTA recovers the secret key of RSM-AES-128 with first-order global success rate increased from 55% to 83%, which significantly increased by 50.91% compared to the best univariate TA. Secondly, based on observations from the distribution of subkey candidates after our MTA, 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% (released DPA Contest Official), with a increase of 14.46% compared to original MTA. Note that for our experiments, the global success rate is optimized to 100% (with a increase of 20.48% compared to original MTA). We also have a comparison between our DFKEA and a simple breadth-first KEA (with the same principle applied in the state-of-the-art KEA). Both theoretical analysis and experimental shows that our DFKEA is significantly more effective than the breadth-first KEA. After all, the purpose of the attack is to effectively protect cryptographic implementation. Finally, we propose two proposals to further improve the practical security of RSM-AES-128 (and other protected cryptographic implementations). Specifically, Fully Shuffled Round Transformation (FSRT) makes all four layer's operations shuffled, and the new improved RSM scheme eliminates the directly related offsets in consecutive rounds. Therefore, these two proposals could be applied to further improve the practical security of RSM-AES-128.

However, although our DFKEA is very efficient and effective than existing breadth-first KEAs, there are still some theoretical and practical issues need to be studied and clarified. The first one is the relation between the global success rate and enumeration depths (also with number of errored subkeys). Moreover, our two proposals for improving the practical security could be further optimized, considering when they are implemented on different platforms like FPGA. As a consequence, we will investigate these questions (e.g. the quantitative relation between success rate and enumeration depths) in the future.

Acknowledgments.

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

References

[BBD+14] Shivam Bhasin, Nicolas Bruneau, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Analysis and Improvements of the DPA Contest v4 Implementation. In *Security, Privacy, and Applied Cryptography Engineering - 4th International Conference, SPACE 2014, Pune, India, October 18-22, 2014. Proceedings*, pages 201–218, 2014.

[BGNT15] Nicolas Bruneau, Sylvain Guilley, Zakaria Najm, and Yannick Teglia. Multivariate High-Order Attacks of Shuffled Tables Recomputation. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 475–494, 2015.

[BKM+15] Andrey Bogdanov, Ilya Kizhvatov, Kamran Manzoor, Elmar Tischhauser, and Marc Witteman. Fast and Memory-Efficient Key Recovery in Side-Channel Attacks. In *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, pages 310–327, 2015.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, pages 252–263, 2000.

[CDGM12] Claude Carlet, Jean-Luc Danger, Sylvain Guilley, and Houssem Maghrebi. Leakage Squeezing of Order Two. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 120–139, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards Sound Approaches to Counteract Power-Analysis Attacks. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 398–412, 1999.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, pages 28–44, 2007.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 13–28, 2002.

[DPA14] TELECOM paristech SEN research group: DPA Contest (4th edn.) (2013-2014, DPACv4.1), http://www.dpacontest.org/v4/index.php. 2014.

[DPA17] TELECOM paristech SEN research group: DPA Contest (4th edn.) (2014-2017, DPACv4.2), http://www.dpacontest.org/v4/42_doc.php. 2017.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 251–261, 2001.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES Smart Card Implementation Resistant to Power Analysis Attacks. In *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, pages 239–252, 2006.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 388–397, 1999.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996.

[LGT+16] Zeyi Liu, Neng Gao, Chenyang Tu, Jian Zhou, Yuan Ma, and Yuan Zhao. Leakage Fingerprints: A Non-negligible Vulnerability in Side-Channel Analysis. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 807–818, 2016.

[MGD11] Houssem Maghrebi, Sylvain Guilley, and Jean-Luc Danger. Leakage Squeezing Countermeasure against High-Order Attacks. In *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication - 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings*, pages 208–223, 2011.

[MGH14] Amir Moradi, Sylvain Guilley, and Annelie Heuser. Detecting Hidden Leakages. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 324–342, 2014.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks - Revealing the Secrets of Smart Cards*. Springer, 2007.

[MOW14] Luke Mather, Elisabeth Oswald, and Carolyn Whitnall. Multi-target DPA Attacks: Pushing DPA Beyond the Limits of a Desktop Computer. In *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, pages 243–261, 2014.

[NGD11] Maxime Nassar, Sylvain Guilley, and Jean-Luc Danger. Formal Analysis of the Entropy / Security Trade-off in First-Order Masking Countermeasures against Side-Channel Attacks. In *Progress in Cryptology - INDOCRYPT 2011 - 12th International Conference on Cryptology in India, Chennai, India, December 11-14, 2011. Proceedings*, pages 22–39, 2011.

[NSGD12] Maxime Nassar, Youssef Souissi, Sylvain Guilley, and Jean-Luc Danger. RSM: A Small and Fast Countermeasure for AES, Secure against 1st and 2nd-order Zero-offset SCAs. In *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1173–1178, 2012.

[Pro05] Emmanuel Prouff. DPA Attacks and S-Boxes. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, pages 424–441, 2005.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 -*

18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings, pages 61–81, 2016.

[RGN13] Pablo Rauzy, Sylvain Guilley, and Zakaria Najm. Formally Proved Security of Assembly Code Against Leakage. *IACR Cryptology ePrint Archive*, 2013:554, 2013.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 413–427, 2010.

[RPD09] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-Order Masking and Shuffling for Software Implementations of Block Ciphers. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 171–188, 2009.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 411–425, 2008.

[SKS09] François-Xavier Standaert, François Koeune, and Werner Schindler. How to Compare Profiled Side-Channel Attacks? In *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 485–498, 2009.

[SNG+10] Youssef Souissi, Maxime Nassar, Sylvain Guilley, Jean-Luc Danger, and Florent Flament. First Principal Components Analysis: A New Side Channel Distinguisher. In *Information Security and Cryptology - ICISC 2010 - 13th International Conference, Seoul, Korea, December 1-3, 2010, Revised Selected Papers*, pages 407–419, 2010.

[VG17] Nikita Veshchikov and Sylvain Guilley. Implementation Flaws in the Masking Scheme of DPA Contest v4. *IET Information Security*, 2017.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 390–406, 2012.

[YE13] Xin Ye and Thomas Eisenbarth. On the Vulnerability of Low Entropy Masking Schemes. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 44–60, 2013.

A Appendix

A.1 RSM-AES-128 Implementation Adopted in DPA Contest v4.2

The Alg.3 is a pseudo-code form of RSM-AES-128 implementation [BBD+14] which is adopted as open target of DPA Contest v4.2.

Algorithm 3 RSM-AES-128 for the DPA Contest v4.2

Input : Public: $X \leftarrow Plain[16]$.
Private: $Shuffle0[16]$, $Shuffle10[16]$, $offset[16]$, $Key[16]$.
Output: $X \rightarrow Cipher[16]$.

$RoundKey[0] \leftarrow RoundKey[0] \oplus Mask[offset[]]$
for $r \in [0, 8]$ **do**
  $X = X \oplus RoundKey[r]$
  if r=0 **then**
    for $i \in Shuffle0([0, 15])$ **do**
      $X_i = MaskedSubBytes_{offset[i]+r}(X_i)$
    end for
  else
    for $i \in [0, 15]$ **do**
      $X_i = MaskedSubBytes_{offset[i]+r}(X_i)$
    end for
  end if
  $X = ShiftRows(X)$
  $X = MixColumns(X)$
  for $i \in [0, 15]$ **do**
    $MaskCompensation[i] = ShiftRows(MixColumns(Mask[offset[i]+(r+1)]))\oplus Mask[(offset[i] + (r + 1))]$
  end for
  $X = X \oplus MaskCompensation[]$
end for
$X = X \oplus RoundKey[9]$
for $i \in Shuffle10([0, 15])$ **do**
  $X_i = MaskedSubBytes_{offset[i]+9}(X_i)$
end for
$X = ShiftRows(X)$
$X = X \oplus RoundKey[10]$
for $i \in [0, 15]$ **do**
  $MaskCompensationLastRound[i] = ShiftRows(Mask[offset[i] + 10])$
end for
$X = X \oplus MaskCompensationLastRound[]$

A.2 Evaluation Results of Our MTA Attack from DPA Contest v4.2

Note that 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 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) released by DPA Contest Official are depicted as Fig.10 and Fig.11. For the purpose of comparison, the evaluation results of Partial Guessing Entropy (PGE) are showed as Fig.12.

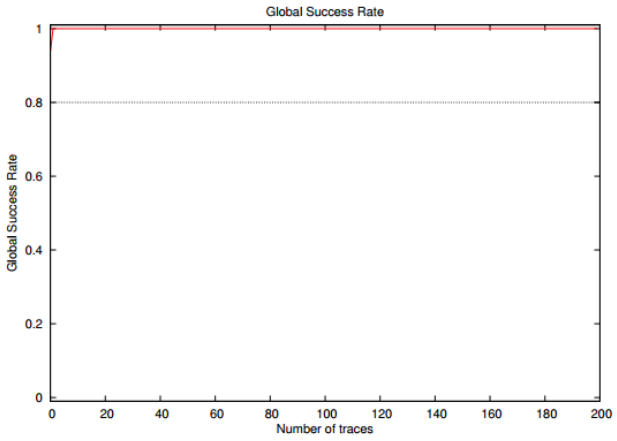


Figure 10: The Global Success Rate (GSR) of our proposed attack released by DPA Contest Official.

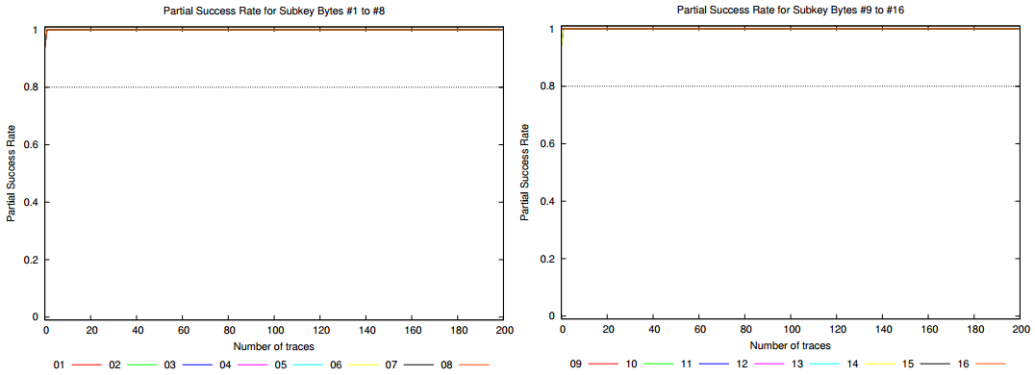


Figure 11: The Partial Success Rate (PSR) of our proposed attack released by DPA Contest Official.

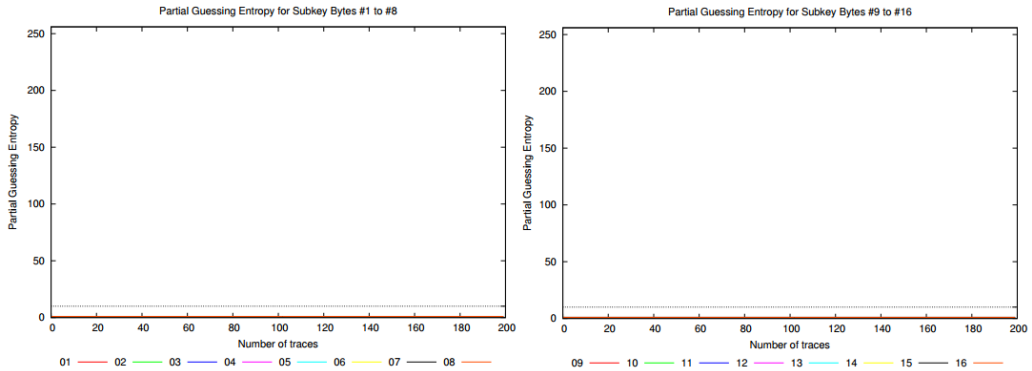


Figure 12: The Partial Guessing Entropy (PGE) of our proposed attack released by DPA Contest Official.