Group Collision Attack

Changhai Ou, Zhu Wang, Degang Sun, and Xinping Zhou

***Abstract*— Key enumeration schemes are used to post-process the scores given by side channel distinguishers and enumerate the key candidates from the most possible one to the least possible one, which can be regarded as optimal tools of key search. However, the application of them is limited by very large key candidate space and computing power consumption. For example, the attacker may spend several weeks or months enumerating the whole 2^{45} key candidates. Unlike the former literature that try to propose a more efficient algorithm to process the distinguishers, scores of key candidates directly, we focus on pre-processing and reducing the key candidate space. To achieve this goal, a new divide and conquer strategy named group collision attack (GCA) is proposed in this paper. The GCA works as follows in brief. The key candidates are first divided into groups on which intra-group collision attack is used to remove the impossible key combinations in each group. Then, the inter-group collision attack is performed to further remove the impossible key combinations between groups. Thus, the complexity of key enumeration is reduced significantly. A series of practical experiments are carried out by using our GCA and the experimental results verify its efficiency.**

***Index Terms*— Group collision attack, GCA, divide and conquer, DPA *contest v*4.1, side channel attack.**

I. Introduction

STANDARD differential side channel attacks [13] such as Correlation Power Analysis (CPA) [5], Differential Power Analysis (DPA) [12], Template Attack (TA) [6] and Mutual Information Analysis (MIA) [11] take advantage of the divide and conquer strategies that divide the full key into several chunks (aka sub-keys) and conquer them one by one to make the full key recovery much easier. For example, the computation complexity is reduced from 2^{128} to $2^8 \cdot 16$ with the divide and conquer strategy if the full 128-bit key of AES-128 is divided into 16 sub-keys. However, if the given side channel information is not sufficient the correct subkey guess may not be ranked the first in which situation the real key can't be extracted directly and the key enumeration process needs to be carried on.

For key exhaustion, the key candidates are exhausted in a random order. Side channel leakages provide information of the key for the attacker. Compared with key exhaustion, key enumeration solutions such as [3], [20] and [25], enumerate the key candidates with the hints given by side channel distinguishers from the most possible one to the least possible one which can be regarded as optimal key exhaustion tools. However, key enumeration is also limited by the computing power of the evaluator. As detailed in [26], the only leaky devices for which we can evaluate the security are the ones that are 'practically insecure' (i.e. for which the leakage allows key enumeration). To enumerate 2^{45} key candidates, several weeks or months may be needed. Moreover, large memory requirement also prevents the application of these solutions.

A. Related Works

In this paper, we aim at answering a very important question: *how can we recover the key from a very large candidate space beyond computing power of key exhaustion and enumeration?* Specifically, we aim at reducing the key candidate space to an enumerable one (e.g. from 2^{64} to 2^{30}). Since a wrong candidate ranking close to the top of a distinguisher may not rank close to the top of another distinguisher. So combining different distinguishers, a number of wrong candidates of this kind can be deleted. Thus, the attacker can enumerate the key in a smaller candidate space. In order to achieve this goal, CPA and Collision Attack (see [4], [15], [17], [23]) are used. Specifically, we use Correlation enhanced Collision Attack (CCA) [17] to post-process the outputs of CPA. CCA attempts to establish the relationship between different key bytes by collisions.

The first practical scheme named Test of Chain was proposed by Bogdanov and Kizhvatov [2] (as introduced in Section II-C). Each chain includes one or several pairs of collisions. Two thresholds Thr_k and Thr_Δ were used in [2], Thr_k was for the key and Thr_Δ was for $\Delta_{(k_a,k_b)} = k_a \oplus k_b$ between two key bytes k_a and k_b. For example, $Thr_k = 10$ denotes that for each key byte, the 10 candidates with maximum correlation coefficients output by CPA are used, and $Thr_\Delta = 10$ denotes that the 10 Δ-s with maximum correlation coefficients between any two key bytes k_a and k_b output by CCA are used. They tried to find a long chain from k_1 to k_{16} including 15 chains (16 key bytes). To ensure that all key bytes and Δ-s are within the thresholds, attacks require many power traces. Moreover, TC does not give solutions to the problem that how to recover the key if multiple collisions occur simultaneously which often happens in fact.

Another practical key recovery scheme in a large candidate space (e.g. 2^{64}) proposed by Wang *et al.* in [27] was named as Fault Tolerant Chain (FTC). In this paper, we only consider

Manuscript received December 25, 2017; revised July 3, 2018; accepted August 12, 2018. Date of publication August 31, 2018; date of current version October 30, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61372062. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Jean-Luc Danger. *(Corresponding author: Zhu Wang.)*

The authors are with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100089, China (e-mail: ouchanghai@iie.ac.cn; wangzhu@iie.ac.cn; sundegang@iie.ac.cn; zhouxinping@bctest.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2018.2868237

the first round key of AES-256. So, the length of each key chunk is 8 bits. Any key byte falling outside the threshold Thr_k will result in very complex or even failed key recovery in FTC. Moreover, if both k_b and $\Delta_{(k_a,k_b)}$ are wrong, and $k_a = \Delta_{(k_a,k_b)} \oplus k_b$ is still satisfied, FTC treats k_2 as a correct key byte (as detailed in Section II-D).

Ou *et al.* proposed Group Verification Chain (GVC) in [19], which enhanced FTC significantly. GVC is based on the theory of Multiple-Differential Collision Attack (MDCA) using binary voting and ternary voting (see [2]). They used several key bytes to verify one key byte. If k_a, k_b and $\Delta_{(k_a,k_b)}$ are all within thresholds, then k_a and k_b passes the verification (as detailed in Section II-E). The frequencies or weights of the correct key byte values, on which MDCA is performed, are higher than these of wrong ones. However, GVC is somewhat a kind of key re-ranking strategy and not a good choice for key recovery.

Hence, how to quickly recover the correct key from a very large space far beyond the computing power of key exhaustion and enumeration is still worthy of further research.

B. Our Contributions

In order to quickly recover the key, we propose a new strategy named Group Collision Attack (GCA) combining the advantages of full key recovery and current divide and conquer attacks in this paper. Compared with key exhaustion and enumeration performed on all possible key candidates (e.g. 128 bits key in the first round of AES-128), GCA only considers a subset of them (e.g. the first 2^{70} candidates). Impossible key combination deletions are performed to further reduce the candidate space. Taking AES-256 for example, GCA firstly divides the full key in the first round into several big groups (e.g. 4) and uses Intra-Group Collision Attack to delete the impossible key combinations in each group. GCA then uses Inter-Group Collision Attack to delete impossible key combinations among groups. The remaining key candidates are significantly reduced in that way.

The key can not be exhausted or enumerated if it locates in a very huge space. However, the magic is that, the key may be easily recovered under GCA. The attacker only needs to save the remaining key candidates after two impossible key combination deletion steps. The total amount of computation and memory required by GCA are very small. If thresholds are set properly, all correct key bytes and Δ-s fall within them, the attacker can construct a more complex GCA scheme to further reduce the remaining key candidates.

C. Organization

This paper is organized as follows: measurement setups, Collision Attack (CA), Test of Chain (TC), Fault Tolerant Chain (FTC) and Group Verification Chain (GVC) are introduced in Section II. Two steps of our Group Collision Attacks (GCA), Intra-Group Collision Attack and Inter-Group Collision Attack, are detailed in Sections III and IV. Experiments on measurements of RSM protected AES-128 implemented on the SASEBO downloaded from the website DPA *contest* $v4.1$ [1] are performed in Section V. Finally, we conclude this paper in Section VI.

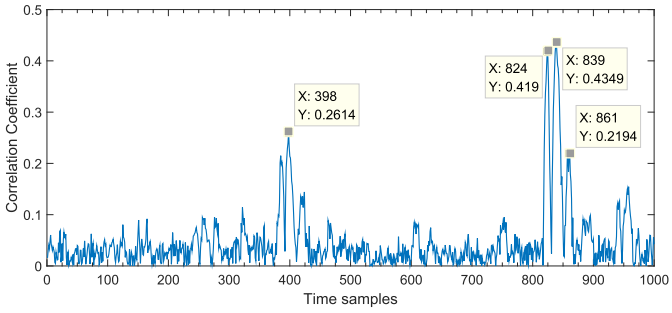

Fig. 1. POIs selection in our experiments.

II. Preliminaries

A. Measurement Setups

Our experiments are performed on an RSM [18] protected AES-256 algorithm implemented on the Side-Channel Attack Standard Evaluation Board (SASEBO). Let $K = \{k_j\}_{j=1}^{16}$, $k_j \in \mathsf{GF}(2^8)$ denote the 16 subkey bytes in the first round of AES-256, $P^i = \left\{p_j^i\right\}_{j=1}^{16}$, $p_j^i \in \mathsf{GF}(2^8)$ denote the encrypted plaintexts, where i=1,2,... is the number of AES-256 executions. 4000 power traces are downloaded from the website of DPA *contest* $v4.1$ [1]. We then implement our experiments on MATLAB R2014a on a Lenovo desktop computer with 4 Intel Core i7-3770 CPUs, 4 GB RAM and 500 GB disk space.

Moradi *et al.* proposed an optimal power consumption model for the S-box outputs of the flawed RSM protected AES-256 on DPA *contest* $v4.1$ [16]. First-order CPA can be directly performed using this model. We find that the POIs (Points-Of-Interest) [10] of the output of the first S-box in the first round are from 100001 to 101000, of which the correlation coefficients output by first-order CPA are shown in Fig. 1. The time samples of the other 15 S-boxes are aligned to these of the first S-box. Since CPA is more efficient than CCA, in order to narrow the attack efficiency gap between them, we extract 4 POIs with highest correlation coefficients (see Fig. 1) from time interval of about a clock cycle as suggested in [22] and use them to perform CCA.

Since CPA and CCA are used in our Group Collision Attack. Here let $\{\xi_j | j = 1, 2, \cdots, 16\}$ denote the sorted candidates output by CPA. We use blackboard bold symbols to represent sets, and math boldface symbols to represent schemes. Moreover, the name of set has subscripts, while the name of a scheme doesn't. For example, $\mathbb{C}_{a,b}^2$ and $\mathbf{C}^2$ introduced in Section II-B indicate a set including all key chains (k_a, k_b) and the collision attack scheme respectively.

B. Collision Attack

Bogdanov and Kizhvatov introduced linear collision attack in [2]. AES-256 performs the SubBytes operation (16 parallel S-box applications) in the first round. A generalized internal AES-256 linear collision occurs if there are two S-boxes in the

TABLE I

CANDIDATES OF THE $3^{rd} \sim 14^{th}$ KEY BYTES WITHIN Thr_k

ξ_3	ξ_4	ξ_5	ξ_6	ξ_7	ξ_8	ξ_9	ξ_{10}	ξ_{11}	ξ_{12}	ξ_{13}	ξ_{14}
198	127	40	125	8	61	235	135	102	240	115	139
122	172	230	80	46	118	7	23	41	64	47	38
69	105	251	174	187	123	243	250	194	4	241	141
98	204	97	25	196	146	94	145	147	239	53	96
124	111	103	173	30	57	244	53	47	33	227	207
24	146	154	209	76	153	6	89	183	76	122	17
150	168	5	52	103	170	37	22	190	84	153	41
153	219	12	59	106	27	43	67	2	112	217	115
21	57	16	101	7	121	44	70	65	165	43	1
22	71	55	106	12	126	188	87	141	40	54	55

TABLE II

PART OF GUESSING Δ-S BETWEEN THE $3^{rd} \sim 9^{th}$ KEY BYTES WITHIN Thr_Δ

$\Delta_{(k_3,k_4)}$	$\Delta_{(k_3,k_5)}$	$\Delta_{(k_4,k_5)}$	$\Delta_{(k_4,k_6)}$	$\Delta_{(k_5,k_6)}$	$\Delta_{(k_5,k_7)}$	$\Delta_{(k_6,k_7)}$	$\Delta_{(k_6,k_8)}$	$\Delta_{(k_7,k_8)}$	$\Delta_{(k_7,k_9)}$
185	238	87	2	49	32	117	64	53	162
148	244	134	102	135	236	169	11	115	83
153	110	77	208	189	252	130	6	126	97
115	223	129	136	138	70	83	9	124	180
174	168	17	174	85	91	23	116	191	249
20	163	231	123	193	117	32	239	140	10
180	197	242	234	209	215	185	26	164	14
128	42	147	150	141	96	102	140	113	251
110	32	203	80	112	66	0	212	154	227
159	100	161	3	75	125	53	209	194	185

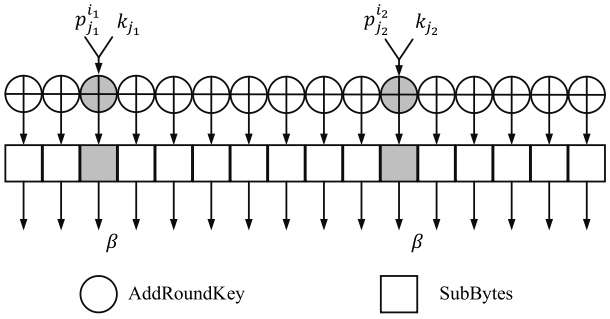

Fig. 2. A linear collision for two AES-256 executions.

same AES encryption or several AES-256 encryptions accepting the same byte value as their input (as shown in Fig. 2).

As detailed in [2] and [19], if a collision

$$\mathsf{Sbox}(p_{j_1}^{i_1} \oplus k_{j_1}) = \mathsf{Sbox}(p_{j_2}^{i_2} \oplus k_{j_2}) \tag{1}$$

happens in the first round of AES-256 (as shown in Fig. 2), the attacker obtains a linear equation

$$p_{j_1}^{i_1} \oplus p_{j_2}^{i_2} = k_{j_1} \oplus k_{j_2} = \Delta_{(k_{j_1},k_{j_2})}. \tag{2}$$

We define (k_{j_1}, k_{j_2}) as an impossible key combination as long as there is at least one of k_{j_1}, k_{j_2} and $\Delta_{(k_{j_1},k_{j_2})}$ out of their corresponding thresholds. If m collisions are detected, then a system of m linear equations can be obtained:

$$\begin{cases} k_{j_1} \oplus k_{j_2} = \Delta_{(k_{j_1},k_{j_2})}, \\ k_{j_3} \oplus k_{j_4} = \Delta_{(k_{j_3},k_{j_4})}, \\ \vdots \\ k_{j_{2m-1}} \oplus k_{j_{2m}} = \Delta_{(k_{j_{2m-1}},k_{j_{2m}})}. \end{cases} \tag{3}$$

It is worth noting that some of these equations are independent. Thus they can be divided into h_0 independent subsystems with respect to the parts of key [2], of which each may have one free variable and one or more equations. Let h_1 denote the number of all missing variables which are not in these subsystems. Each of the subsystems or missing variables is called a chain and each equation is defined as a step of a chain. Hence the number of chains is $h = h_0 + h_1$.

Here we take key bytes k_5, k_6, k_7 and k_8 in our experiments for example to illustrate the attack efficiency of collision attack. Thr_k and Thr_Δ here are set to 10. The key bytes candidates and guessing Δ-s of these 4 key bytes within the thresholds Thr_k and Thr_Δ are shown in Table I and Table II. Other key bytes candidates and Δ-s listed here will be used in Section IV.

In order to extend the following contents and prepare for our introduction of group collision attacks, we change the way to express collision attacks. Let (k_a, k_b) $(a, b \in [1, 16], a < b)$ denote a pair of collision including two key bytes k_a and k_b within thresholds Thr_k and Thr_Δ. Then, a set $\mathbb{C}_{a,b}^2$ including all chains (k_a, k_b) can be defined as

$$\mathbb{C}_{a,b}^2 = \{(k_a, k_b) | k_a \in \xi_a, k_b \in \xi_b, k_a \oplus k_b \in \Delta_{(k_a,k_b)}\}. \tag{4}$$

As introduced in Section II-A, we name this simplest collision scheme as $\mathbf{C}^2$. We search key pairs (k_a, k_b) $(a, b \in [5, 8], a < b)$ of some two key bytes in Table I satisfying that their corresponding $\Delta_{(k_a,k_b)}$ are in Table II. Each pair (k_a, k_b) constitutes a chain in $\mathbb{C}_{a,b}^2$. Let $\left|\mathbb{C}_{a,b}^2\right|$ represent the number of elements of the set $\mathbb{C}_{a,b}^2$. As shown in Table I and Table II, if k_5, k_6 and $\Delta_{(k_5,k_6)}$ are all within the thresholds, we then add (k_5, k_6) to set $\mathbb{C}_{5,6}^2$ (as shown in Table III).

Example 1: For example, the value 40 of k_5, the value 125 of k_6, and the value 85 of $\Delta_{(k_5,k_6)}$ satisfy $85 = 40 \oplus 125$. *So, we add (40,125) to $\mathbb{C}_{5,6}^2$ (as shown in Table III). Although the value 40 of k_5 and the value 80 of k_6 are within the*

TABLE III

PART OF $\mathbf{C}^2$ CHAINS OF THE $5^{th} \sim 8^{th}$ KEY BYTES

$\mathbb{C}^2_{5,6}$		$\mathbb{C}^2_{5,7}$		$\mathbb{C}^2_{6,7}$		$\mathbb{C}^2_{6,8}$		$\mathbb{C}^2_{7,8}$	
40	25	40	8	125	8	125	61	8	61
40	125	40	196	125	46	125	118	8	123
230	173	40	106	125	106	125	123	8	118
251	174	251	7	125	196	125	146	8	121
97	80	103	7	174	7	173	121	8	146
97	52			52	103	52	61	46	27
154	209			106	12	59	123	187	121
5	52			106	106	59	61	196	123
5	80							30	146
16	173							76	121
16	209							76	61
								103	27
								106	27
								7	121
								7	123
								7	118
								12	57

threshold Thr_k, $40 \oplus 80 = 120$ *is out of threshold* $\Delta_{(k_5,k_6)}$. *So, key pair* $(40, 80)$ *is discarded.*

Finally, all key chains (k_5, k_6) are saved in the first two columns of Table III. Each row saves a key chain so that the number of possible key chains in $\mathbb{C}^2_{5,6}$ is 11. The attacker continues to construct key chain sets $\mathbb{C}^2_{5,7}$, $\mathbb{C}^2_{6,7}$, $\mathbb{C}^2_{6,8}$ and $\mathbb{C}^2_{7,8}$. All $\mathbf{C}^2$ chains of them calculated from Table I and Table II are shown in Table III.

As shown in Table III, there are only 11, 5, 8 and 17 possible combinations (chains) in sets $\mathbb{C}^2_{5,6}$, $\mathbb{C}^2_{5,7}$, $\mathbb{C}^2_{6,7}$, $\mathbb{C}^2_{6,8}$ and $\mathbb{C}^2_{7,8}$ respectively. If brute-force attack is used, 100 combinations between any two key bytes should be exhausted. So, the key candidate space becomes much smaller after impossible key combination deletion in each group. Although the complexity of $\mathbf{C}^2$ scheme is $(Thr_k)^2 \cdot (Thr_\Delta)$, if the 16 bytes key is divided into 8 independent groups, the complexity of calculating all $\mathbf{C}^2$ chains is only $8 \cdot (Thr_k)^2 \cdot (Thr_\Delta)$ $(8 \cdot 10^2 \cdot 10)$ here.

C. Test of Chain

Bogdanov and Kizhvatov defined Test of Chain (TC) in [2]. Suppose that the attacker uses CPA to calculate the correlation coefficient of each key candidate. He sorts all 256 candidates of a key byte in descending order and obtains the 16 guessing key byte sequences $\{\xi_j | j = 1, 2, \cdots, 16\}$ of AES-256 algorithm. He also uses CCA to calculate the correlation coefficients in $\Delta_{(k_a,k_b)}$.

In each list ξ_j $(1 \le j \le 16)$, Bogdanov *et al.* only consider values among the top Thr_k $(1 \le Thr_k \le 256)$ positions, which are the most possible candidates of the key byte k_j. Let a circle with (without) a numeric value denote a correct (wrong) guessing key, the attacker tries to find a chain from ξ_1 to ξ_{16} including 16 key bytes (see Fig. 3). The guess chain is accepted if all key bytes of the chain are among the top Thr_k candidates in their corresponding list ξ_j. The guess chain is rejected if at least one key byte of the chain falls outside the Thr_k top candidates in its corresponding list ξ_j.

D. Fault Tolerant Chain

In order to recover the key efficiently, the attacker usually hopes that a key chain includes 15 steps as introduced in [2].

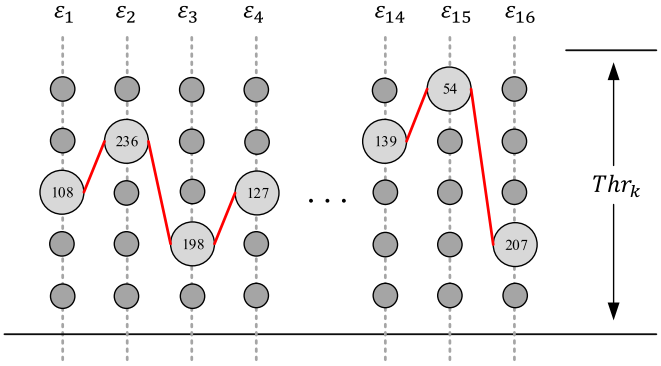

Fig. 3. Test of Chain (TC).

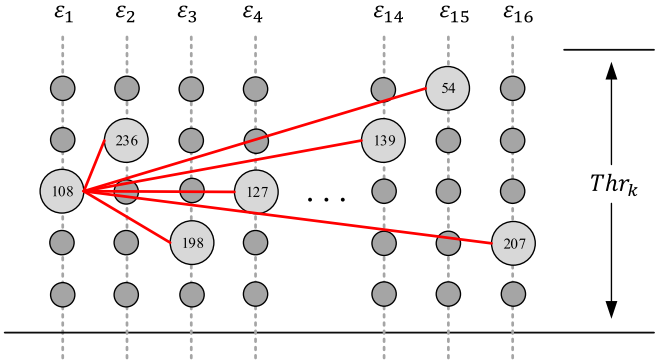

Fig. 4. Fault Tolerant Chain (FTC).

For a chain, one of the common cases is that there are several steps on the path from the free variable to the end. If an error takes place in one of these steps, the key bytes computed in the following steps will be wrong, which will result in the failure of the whole attack. Unfortunately, this kind of errors happen with non-negligible probability, which lead to low efficiency of Bogdanov's attack.

Wang *et al.* constructed a new scheme named Fault Tolerant Chain (FTC) [27]. In their scheme, $k_j (j \ge 2)$ only depends on k_1 instead of any other 14 key bytes (as shown in Fig. 4). There is a path from k_1 to k_j $(j = 2, \cdots, 16)$. If k_j is wrong (out of the threshold Thr_k), they can still attempt to recover other key bytes. In their paper, the threshold Thr_Δ of collision attack is set to 1. So, only Thr_k is taken into consideration. Enlarging the threshold will lead to very complex key recovery. If k_j is out of the threshold, they deduce that the chain is wrong. Subsequently, a brute-force attack is performed to find this correct key byte.

The search complexity of FTC is determined by the number of wrong key bytes as given by Wang *et al.* in [27]. They indicated that the maximum number of candidates was $2^8 \cdot (2^8)^n \cdot \binom{15}{n}$ for n wrong key bytes. If there are total τ times that more than one candidate are within the threshold, then $15 - \tau$ errors can be detected. Then the number of key candidates the attacker needs to exhaust is

$$\mathsf{N_e} = 2^8 \cdot \left(2^8\right)^n \cdot \binom{\tau}{\tau + n - 15}. \tag{5}$$

Cautionary Note. Suppose that both k_2 and $\Delta_{(k_1,k_2)}$ are wrong. If $k_1 = \Delta_{(k_1,k_2)} \oplus k_2$ is still satisfied, then the attacker gets a wrong key byte value of k_2. However, he is completely

unaware of the mistake. Actually, the threshold Thr_Δ is always set to 1. The larger Thr_Δ and Thr_k, the higher probability of this type of error. In other words, with the increase of Thr_Δ and Thr_k, experimental failures caused by this type of error increase significantly.

E. Group Verification Chain

Bogdanov proposed MDCA using binary voting and ternary voting for collision detection between any two S-boxes [2]. Specifically, it is used to compare the power traces of two S-boxes to judge if collision has happened.

We introduced Group Verification Chain (GVC) in [19], which could be used to re-rank the key sequences under the condition that Thr_k and Thr_Δ were set largely. Group verification chain here is defined as the mutual verification among key bytes. Let $\xi_j^{\kappa_1}$ and $\xi_{\gamma+1}^{\kappa_2}$ denote the κ_1-th and κ_2-th key values in ranks ξ_j and $\xi_{\gamma+1}$, $\Delta^{\kappa_3}_{(k_j,k_{\gamma+1})}$ denote the κ_3-th value in the rank $\Delta_{(k_j,k_{\gamma+1})}$. Then,

$$\xi_j^{\kappa_1} \oplus \xi_{\gamma+1}^{\kappa_2} = \Delta^{\kappa_3}_{(k_j,k_{\gamma+1})} \tag{6}$$

is satisfied if $\xi_j^{\kappa_1}$, $\xi_{\gamma+1}^{\kappa_2}$ and $\Delta^{\kappa_3}_{(k_j,k_{\gamma+1})}$ are the correct ones. Then the frequencies of $\xi_j^{\kappa_1}$ and $\xi_{\gamma+1}^{\kappa_2}$ are increased by 1. We say, $\xi_{\gamma+1}^{\kappa_2}$ is verified by $\xi_j^{\kappa_1}$. Actually, we don't care if $\xi_j^{\kappa_1}$, $\xi_{\gamma+1}^{\kappa_2}$ and $\Delta^{\kappa_3}_{(k_j,k_{\gamma+1})}$ are correct. Each key candidate can be verified by other candidates using collisions. When the frequency of $\xi_j^{\kappa_1}$ is larger than the differential threshold, $\xi_j^{\kappa_1}$ is of high probability to be the correct key.

Let Thr_d denote a decision threshold of possible key byte values. Suppose that we use the $1, \cdots, \gamma$ key bytes to verify the $(\gamma+1)$-th key byte. Then, a Frequency based GV-MDCA (FGV-MDCA) [19] can be defined as:

$$\Psi_{\xi_{\gamma+1}^{\kappa_2}} = \begin{cases} 1(\textbf{collision}), & \text{if} \quad \Phi_{\xi_{\gamma+1}^{\kappa_2}} > Thr_d \\ 0(\textbf{no collision}), & \text{else} \quad \Phi_{\xi_{\gamma+1}^{\kappa_2}} < Thr_d \end{cases} \tag{7}$$

where $\Psi_{\xi_{\gamma+1}^{\kappa_2}}$ denotes that $\xi_{\gamma+1}^{\kappa_2}$ is a candidate byte value in the sequence $\xi_{\gamma+1}$, and

$$\Phi_{\xi_{\gamma+1}^{\kappa_2}} = \sum_{j=1}^{\gamma} \Theta(\xi_j^{\kappa_1}, \xi_{\gamma+1}^{\kappa_2}). \tag{8}$$

$\Theta(\xi_j^{\kappa_1}, \xi_{\gamma+1}^{\kappa_2})$ here is defined as

$$\Theta(\xi_j^{\kappa_1}, \xi_{\gamma+1}^{\kappa_2}) = \begin{cases} 1, & \text{if} \quad \xi_j^{\kappa_1} \oplus \xi_{\gamma+1}^{\kappa_2} = \Delta^{\kappa_3}_{(j,\gamma+1)} \\ 0, & \text{else.} \end{cases} \tag{9}$$

There are 16 guessing key byte sequences $\{\xi_j | j = 1, 2, \cdots, 16\} \in \mathsf{GF}(2^8)$ corresponding to S-boxes $1, \cdots, 16$ of AES algorithm. Suppose that we use key byte values in $\xi_1, \cdots, \xi_4$ to verify key byte candidates in $\xi_5, \cdots, \xi_{16}$ of AES algorithm (as shown in Fig. 5). 120 sequences of $\Delta_{(k_a,k_b)}$ between any two key bytes k_a and k_b $(1 \le a < b \le 16)$ are calculated. For correct key byte values and Δ-s, the Equation 6 is usually satisfied. Finally, the attacker gets the correct key.

The frequencies of the correct byte values will be higher than these of wrong candidates in our group verification chain.

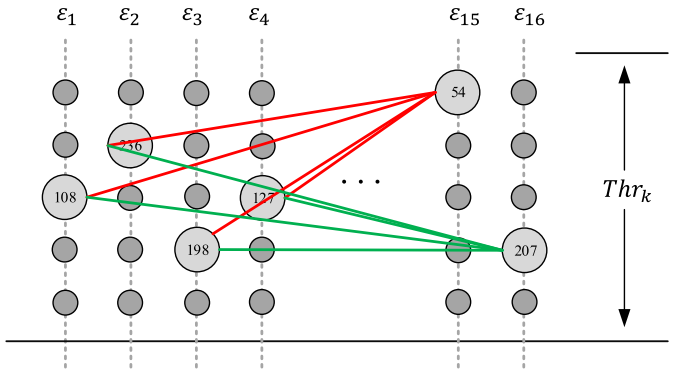

Fig. 5. Group Verification Chain (GVC).

This also indicates that the correct key byte values obtain more support in the process of mutual verification, which makes them obvious. The attacker can effectively recover the key by observing the frequencies of key byte values. He does not need to exhaust all possible key values within the threshold.

III. Intra-Group Collision Attack

The collision attack only including a pair of collision introduced in Section II-B is the simplest. In this paper, our purpose is to reduce the key candidate space by making full use of collision attacks. We divide the 16 bytes key in the first round of AES-256 into several big groups (pieces). Collisions within each group establish relationship among key bytes. We name these collisions as group collisions. Intra-Group Collision Attack is defined as collisions within each group. Each group establishes the connection among key bytes by multi-pairs of collisions. GCA can even delete more impossible key bytes combinations before enumeration or exhaustion. Then, the attacker can search the key more efficiently in a much smaller key candidate space.

A. Key Chain Based Intra-Group Collision Attack

Actually, $\mathbf{C}^2$ scheme introduced in Section II-B are the simplest side channel collision between any two key bytes. The attacker or evaluator can find several collisions among 3 or 4 key bytes simultaneously. For example, $\mathbf{C}^3$ scheme includes two collisions, one is between k_a and k_b, and the other is between k_b and k_c. Then, a set including all $\mathbf{C}^3$ chains of k_a, k_b and k_c is defined as

$$\mathbb{C}^3_{a,b,c} = \{(k_a, k_b, k_c) | (k_a, k_b) \in \mathbb{C}^2_{a,b}, (k_b, k_c) \in \mathbb{C}^2_{b,c}\}, \tag{10}$$

which means both (k_a, k_b) and (k_b, k_c) are $\mathbf{C}^2$ chains. Specifically, k_a, k_b and k_c are within the Thr_k; $\Delta_{(k_a,k_b)}$ and $\Delta_{(k_b,k_c)}$ are within the Thr_Δ; k_b in (k_a, k_b) and (k_b, k_c) is the same. Similar to $\mathbf{C}^2$, we name this new scheme as $\mathbf{C}^3$. Obviously, $\mathbf{C}^3$ is more complex than $\mathbf{C}^2$, it includes more collisions.

The implementation of GCA is very simple. The construction of $\mathbb{C}^3_{a,b,c}$ chains from $\mathbb{C}^2_{a,b}$ and $\mathbb{C}^2_{b,c}$ is shown in Algorithm 1. $\left(\mathbb{C}^2_{a,b}\right)_{m_1}$ here represents the m_1-th element of $\mathbb{C}^2_{a,b}$. The algorithm traverses sets $\mathbb{C}^2_{a,b}$ and $\mathbb{C}^2_{b,c}$. If $k_{b_1} = k_{b_2}$,

then (k_a, k_{b_1}, k_c) (i.e. (k_a, k_{b_2}, k_c)) is added to $\mathbb{C}^3_{a,b,c}$. This algorithm achieves Equation 10. The implementation of other GCA schemes in this paper is similar to $\mathbb{C}^3_{a,b,c}$, of which the algorithms are not given.

Algorithm 1 Construction of $\mathbb{C}^3_{a,b,c}$ Chains.

Input: Two $\mathbf{C}^2$ sets $\mathbb{C}^2_{a,b}$ and $\mathbb{C}^2_{b,c}$
Output: The set $\mathbb{C}^3_{a,b,c}$

```
for m1 = 1, ..., |C^2_{a,b}| do
    (k_a, k_{b1}) ← (C^2_{a,b})_{m1}
    for m2 = 1, ..., |C^2_{b,c}| do
        (k_{b2}, k_c) ← (C^2_{b,c})_{m2}
        if k_{b1} = k_{b2} then
            add (k_a, k_{b1}, k_c) to C^3_{a,b,c}
        end if
    end for
end for
```

Compared to $\mathbf{C}^2$ chains, $\mathbf{C}^3$ chains delete impossible combinations where k_b in (k_a, k_b) and (k_b, k_c) is not the same. So, the number of remaining candidates is reduced significantly. Similarly, the attacker or evaluator can construct more complex key chain based intra-group collision attack schemes such as $\mathbf{C}^4$, of which a set including k_a, k_b, k_c and k_d can be defined as

$$\mathbb{C}^4_{a,b,c,d} = \Big\{(k_a, k_b, k_c, k_d) | (k_a, k_b, k_c) \in \mathbb{C}^3_{a,b,c}, (k_b, k_c, k_d) \in \mathbb{C}^3_{b,c,d}\Big\}. \quad (11)$$

Compared to $\mathbf{C}^2$ and $\mathbf{C}^3$, $\mathbf{C}^4$ puts forward higher requirements to candidates. For many $\mathbf{C}^3$ chains, (k_a, k_b, k_c) and (k_b, k_c, k_d) are not satisfied simultaneously. Since k_b is verified by k_a and k_c, and k_c is verified by k_b and k_d, $\left|\mathbb{C}^4_{a,b,c,d}\right|$ is much smaller than $\left|\mathbb{C}^3_{a,b,c}\right| \cdot \left|\mathbb{C}^3_{b,c,d}\right|$. If the thresholds Thr_Δ and Thr_k are reasonable, the correct key bytes can be successfully used to construct $\mathbf{C}^4$ chains and a lot of error $\mathbf{C}^3$ chains are deleted.

Example 2: Taking the $\mathbf{C}^3$ *chain set* $\mathbb{C}^3_{5,6,7}$ *for example,* (k_5, k_6) *and* (k_6, k_7) *are satisfied simultaneously. For example,* $(k_5, k_6) = (40, 125)$ *and* $(k_6, k_7) = (125, 8)$ *constitute* $(k_5, k_6, k_7) = (40, 125, 8)$. *However,* $(k_5, k_6) = (16, 209)$ *and* $(k_6, k_7) = (125, 8)$ *can neither constitute* $(k_5, k_6, k_7) = (16, 209, 8)$ *nor constitute* $(k_5, k_6, k_7) = (16, 125, 8)$. $\left|\mathbb{C}^2_{5,6}\right|$ *and* $\left|\mathbb{C}^2_{6,7}\right|$ *are 11 and 8 respectively. To recover key bytes* (k_5, k_6, k_7)*,* 88 *key candidates should be considered. If* $\mathbb{C}^3_{5,6,7}$ *chains are constructed, only 7 candidates should be taken into consideration.*

The construction of $\mathbf{C}^3$ chains using guessing key bytes in Table I and Δ-s in Table II are shown in Table IV. We get a conclusion that $\left|\mathbb{C}^3_{5,6,7}\right| = 7$ and $\left|\mathbb{C}^3_{6,7,8}\right| = 14$. Compared to $\mathbf{C}^3$ chains, $\mathbf{C}^4$ chains are more efficient. There are only 13 $\mathbf{C}^4$ chains in thresholds (as shown in Table VII). The constraints of $\mathbf{C}^4$ are more strict than $\mathbf{C}^3$. Compared with 10^4 candidates in the thresholds, only 13 possible $\mathbf{C}^4$ chains of k_5, k_6, k_7 and k_8 are considered by the attacker.

TABLE IV
$\mathbf{C}^3$ CHAINS OF THE $5^{th} \sim 8^{th}$ KEY BYTES

$\mathbb{C}^3_{5,6,7}$			$\mathbb{C}^3_{6,7,8}$		
40	125	8	125	8	61
40	125	46	125	8	123
40	125	106	125	8	118
40	125	196	125	8	121
251	174	7	125	8	146
97	52	103	125	46	27
5	52	103	125	106	27
			125	196	123
			174	7	121
			174	7	123
			174	7	118
			52	103	27
			106	12	57
			106	106	27

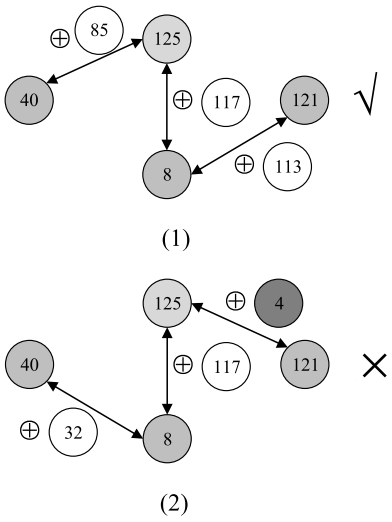

Fig. 6. A correct $\mathbf{C}^4$ chain and a wrong $\mathbf{C}^4$ chain.

The advantage of key chain based GCA schemes is that they are simple to construct and suitable for small Thr_k and Thr_Δ. Since more strict constraint may delete the correct key bytes. However, the number of error $\mathbf{C}^3$ chains deleted by these schemes is still very limited. After all, the key candidate space ${Thr_k}^{16}$ here is too huge. For example, if Thr_k is set to 32, the key candidate space reaches 2^{80}.

Actually, for key chain based GCA schemes, key bytes in the middle of chains are verified two times, and the two key bytes at two ends are verified only once. So, key bytes in the middle are more likely to be the correct ones. In other words, k_b and k_c are more credible than k_a and k_d in $\mathbb{C}^4_{a,b,c,d}$, since they are verified by two collisions (k_a, k_b), (k_b, k_c) and (k_b, k_c), (k_c, k_d) separately. However, k_a and k_d are verified only once. Thus, k_b and k_c are more likely to be the correct candidates. Each verification means that more impossible key combinations are removed, the attacker or evaluator can get smaller key candidate space after recombination.

Key chain based GCA schemes have a problem of insufficient verification (see Example 3).

Example 3: Taking $(k_5, k_6, k_7, k_8) = (40, 125, 8, 121)$ *in Fig. 6 for example, the light-grey circles denote key*

candidates in ξ_j $(5 \le j \le 8)$. *The colorless circles denote that* Δ *values are within threshold* Thr_Δ, *the dark-grey circle denotes that the* Δ *values are out of threshold* Thr_Δ. $(k_5, k_6) = (40, 125)$, $(k_6, k_7) = (125, 8)$ *and* $(k_7, k_8) = (8, 121)$ *in the first chain of Fig. 6 are all* $\mathbf{C}^2$ *chains and satisfy that* $40 \oplus 125 = 85$, $125 \oplus 8 = 117$ *and* $8 \oplus 121 = 113$. *So, in this case,* $(k_5, k_6, k_7, k_8) = (40, 125, 8, 121)$ *is a* $\mathbf{C}^4$ *chain. The second chain also includes the same key candidate values, and satisfies that* $40 \oplus 8 = 32$ *and* $125 \oplus 8 = 117$. *However, the dark-grey circle* $125 \oplus 121 = 4$ *is out of* $\Delta_{(k_6,k_8)}$. *So, in this case,* $(k_5, k_6, k_7, k_8) = (40, 125, 8, 121)$ *is not a* $\mathbf{C}^4$ *chain.*

In our $\mathbf{C}^4$ scheme, the contradiction given in Example 4 happens due to the insufficient verification of chains. In order to further improve the reliability of key bytes located in two ends of key chain and reduce the key candidate space, we propose ring based GCA schemes in the next subsection.

B. Key Ring Based Inter-Group Collision Attack

In Section III-A, we introduce our key chain based GCA schemes, which work very well under small thresholds Thr_k and Thr_Δ. If the thresholds are small, the correct key bytes may fall outside of them. If the attacker enlarges thresholds, the success rate [24] will be improved, since more correct key bytes and Δ-s fall within Thr_k and Thr_Δ. However, it is very time-consuming, since the key candidate space becomes larger. In order to improve the reliability of key bytes at two ends of chain and further reduce the key candidate space in large thresholds, we propose the conception of the **Key Ring** in this section. A key ring $\mathbf{R}^n$ consists of n $\mathbf{C}^2$ chains (k_1, k_2), $(k_2, k_3), \ldots, (k_{n-1}, k_n)$ and (k_1, k_n). A set of ring $\mathbf{R}^3_{a,b,c}$ constituting of k_a, k_b and k_c is defined as

$$\mathbb{R}^3_{a,b,c} = \Big\{(k_a, k_b, k_c) | (k_a, k_b) \in \mathbb{C}^2_{a,b}, (k_b, k_c) \in \mathbb{C}^2_{b,c}, (k_a, k_c) \in \mathbb{C}^2_{a,c}\Big\}, \quad (12)$$

which means (k_a, k_b), (k_b, k_c) and (k_a, k_c) are $\mathbf{C}^2$ chains simultaneously. In order to reduce the loops in algorithm implementation, all $\mathbf{C}^2$ chains are saved in the same table (e.g. Table III). The attacker firstly finds if (k_a, k_b) and (k_a, k_c) are in the table (this only needs to traverse the table once). If the hypothesis is true, the attacker then continues to find if (k_b, k_c) is also in the table (see Equation 12). So, to construct the ring $\mathbf{R}^3$, the attacker only needs to traverse the $\mathbf{C}^2$ table twice, its complexity is similar to the construction of the $\mathbf{C}^3$ chains.

The $\mathbb{R}^3_{5,6,7}$ and $\mathbb{R}^3_{6,7,8}$ rings constructed by the corresponding guessing key bytes and guessing Δ-s in Table I and Table II are shown in Table V. We get a conclusion that $\left|\mathbb{R}^3_{5,6,7}\right| = 4$ and $\left|\mathbb{R}^3_{6,7,8}\right| = 5$. However, we get another conclusion from Table IV that $\left|\mathbb{C}^3_{5,6,7}\right| = 7$ and $\left|\mathbb{C}^3_{6,7,8}\right| = 14$. This indicates that, $\mathbf{R}^3$ scheme, which adds a constraint (a pair of collision) on $\mathbf{C}^3$ chains, can effectively reduce the key candidate space. This also indicates that ring based GCA schemes are more efficient than chain based ones when deleting impossible key combinations within group because of more strict constraints (collisions).

TABLE V
$\mathbf{R}^3$ RINGS OF THE $5^{th} \sim 8^{th}$ KEY BYTES

$\mathbb{R}^3_{5,6,7}$			$\mathbb{R}^3_{6,7,8}$		
40	125	8	125	8	61
40	125	196	125	8	118
40	125	106	125	8	123
251	174	7	125	8	146
			125	196	123

The $\mathbf{R}^3_{a,b,c}$ only has a pair of collision (k_a, k_c) more than $\mathbb{C}^3_{a,b,c}$. However, an ring is constructed since the existence of this pair of collision. Each of the three key bytes k_a, k_b, k_c on the ring $R^3_{a,b,c}$ is verified by the other two key bytes. Thus, the probability of these three key bytes being the correct ones increases. Key ring based GCA can delete more impossible combinations than key chain based GCA. The attacker can also construct intersecting rings of two $\mathbf{R}^3$ rings (as shown in Fig. 7(1)). An $\mathbb{R}^{2-3}_{a,b,c,d}$ includes 2 rings $\mathbb{R}^3_{a,b,c}$ and $\mathbb{R}^3_{b,c,d}$, and has more stringent constraint than $\mathbf{R}^3$ rings. A set $\mathbb{R}^{2-3}_{a,b,c,d}$ is defined as

$$\mathbb{R}^{2-3}_{a,b,c,d} = \Big\{(k_a, k_b, k_c, k_d) | (k_a, k_b, k_c) \in \mathbb{R}^3_{a,b,c}, (k_b, k_c, k_d) \in \mathbb{R}^3_{b,c,d}\Big\}, \quad (13)$$

where $a < b < c < d$. Actually, $\mathbf{R}^{2-3}$ rings are easy to construct, the attacker can traverse the sets $\mathbb{R}^3_{a,b,c}$ and $\mathbb{R}^3_{b,c,d}$ (see Equation 13) and select all double-rings from each of them if k_b, k_c are equal.

The number of $\mathbf{R}^{2-3}$ rings $\left|\mathbb{R}^{2-3}_{5,6,7,8}\right|$ constructed by the guessing key bytes and guessing Δ-s in Table I and Table II is 5 (as shown in Table VIII). However, if the attacker exhausts all the 4 key bytes in the threshold $Thr_k = 10$, the complexity is 10^4. Obviously, the attacker gets higher efficiency than $\mathbf{C}^3$, $\mathbf{C}^4$ and $\mathbf{R}^3$ schemes. The construction complexity of $\mathbf{R}^{2-3}_{a,b,c,d}$ rings is average $\left|\mathbb{R}^3_{5,6,7}\right| \cdot \left|\mathbb{R}^3_{6,7,8}\right|$ more complex than that of $\mathbf{R}^3$ schemes.

Compared to $\mathbf{C}^4$ chains shown in Fig. 6, $\mathbf{R}^{2-3}$ performs better. It can detect the contradiction shown in Fig. 6 (as shown in Fig.7). Since $\mathbb{R}^{2-3}_{5,6,7,8}$ only contains $\mathbf{C}^2$ chain $\mathbb{C}^2_{7,8} = (9, 121)$, and $(k_6, k_8) = (125, 121)$ is not a $\mathbf{C}^2$ chain. This also indicates that more constraints (chains) can delete more impossible combinations.

In the case of large thresholds Thr_k and Thr_Δ, the correct key bytes and Δ-s are within thresholds no matter what kind of constraints we use. The more harsh conditions, the smaller number of remaining $\mathbf{R}^{2-3}$ rings. Moreover, we divide the 16-byte key in the first round of AES-256 into 4 big groups, which are independent of each other. The number of remaining key rings in each group is small, the attacker can efficiently exhaust the 16 bytes key in a much smaller candidate space. The attacker can also construct more complex rings containing more collisions. With the increase of Thr_k and Thr_Δ, the number of $\mathbf{R}^3$ rings or $\mathbf{C}^4$ chains increases very rapidly. Moreover, more complex rings or chains mean more loops

TABLE VI

PART OF GUESSING Δ-S BETWEEN THE $8^{th} \sim 14^{th}$ KEY CANDIDATES

$\Delta_{(k_8,k_9)}$	$\Delta_{(k_8,k_{10})}$	$\Delta_{(k_9,k_{10})}$	$\Delta_{(k_9,k_{11})}$	$\Delta_{(k_{10},k_{11})}$	$\Delta_{(k_{10},k_{12})}$	$\Delta_{(k_{11},k_{12})}$	$\Delta_{(k_{11},k_{13})}$	$\Delta_{(k_{12},k_{13})}$	$\Delta_{(k_{12},k_{14})}$	$\Delta_{(k_{13},k_{14})}$
214	186	108	196	225	119	150	21	131	123	49
59	199	109	133	168	171	78	191	197	50	197
190	172	252	141	174	199	98	151	1	47	248
204	150	235	78	233	247	142	255	128	223	254
94	239	70	120	195	55	38	83	41	214	98
148	16	122	194	238	27	170	2	188	219	136
224	144	150	170	20	18	219	28	214	125	120
112	42	199	249	223	31	139	6	223	121	62
241	43	17	198	149	202	133	152	95	153	172
129	123	226	224	57	111	246	136	105	89	85

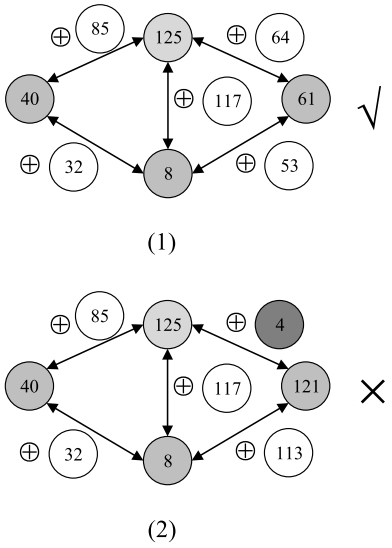

Fig. 7. A correct $\mathbf{R}^{2-3}$ ring and a wrong $\mathbf{R}^{2-3}$ ring.

in the program. So, the efficiency of program should be also taken into consideration.

IV. INTER-GROUP COLLISION ATTACK

In Section III, we introduce intra-group collision attack, which is a distinguisher to delete impossible key combinations in each group. This is the first step of our Group Collision Attack (GCA). The second step of GCA is inter-group collision attack, which aims at deleting impossible key combinations among groups. Then, the attacker can use the state-of-art key enumeration solutions to enumerate the key, or directly use exhaustion. For simplicity, here we only use key exhaustion.

The correct key bytes and Δ-s are within thresholds if Thr_k and Thr_Δ are set largely enough. In this case, each correct chain or ring is within the thresholds. What the attacker needs to do is using GCA proposed in Section III to delete impossible combinations in each big group. Suppose that he divides the entire key into 4 groups, and uses $\mathbf{C}^4$ chains to delete impossible key combinations in each group. Here we recombine the remaining key candidates of the first step.

Actually, the possible key combinations in each group are greatly reduced after the first round of deletion. Referring to Section II-D, let $\mathsf{N_e}$ denote the number of possible key candidates needs to be enumerated. Suppose that there are $\mathsf{N_{e_1}}$, $\mathsf{N_{e_2}}$, $\mathsf{N_{e_3}}$ and $\mathsf{N_{e_4}}$ chains in $\mathbb{C}^4_{1,2,3,4}$, $\mathbb{C}^4_{5,6,7,8}$, $\mathbb{C}^4_{9,10,11,12}$ and $\mathbb{C}^4_{13,13,15,16}$ respectively, a very simple solution to re-combine the full key is to exhaust every possible combinations. By doing this, the attacker will get $\mathsf{N_{e_1}} \cdot \mathsf{N_{e_2}} \cdot \mathsf{N_{e_3}} \cdot \mathsf{N_{e_4}}$ possible combinations. This value increases very rapidly with Thr_k and Thr_Δ. Obviously, it is not a good combination strategy.

So, we propose a new solution to re-combine the full key in GCA. We use verification chain or ring to re-combine the remaining candidates in each group and carry out a second round impossible key combinations deletion. For example, we use $\mathbb{C}^4_{3,4,5,6}$ and $\mathbb{C}^4_{7,8,9,10}$ to verify $\mathbb{C}^4_{5,6,7,8}$, and use $\mathbb{C}^4_{7,8,9,10}$ and $\mathbb{C}^4_{11,12,13,14}$ to verify $\mathbb{C}^4_{9,10,11,12}$ (as shown in Table VII). Δ values are shown in Table II and Table VI. We delete chain (k_5, k_6, k_7, k_8) in set $\mathbb{C}^4_{5,6,7,8}$ if (k_3, k_4, k_5, k_6) is not in $\mathbb{C}^4_{3,4,5,6}$ or (k_7, k_8, k_9, k_{10}) is not in $\mathbb{C}^4_{7,8,9,10}$. $\mathbb{C}^4_{9,10,11,12}$ is processed in the same way. There are 13, 16 possible combinations in the set $\mathbb{C}^4_{5,6,7,8}$ and $\mathbb{C}^4_{9,10,11,12}$ respectively before verification. However, 4 and 3 chains are left after verification (as shown in Table IX). So, the number of possible combinations drops from $13 \cdot 16 = 208$ to $3 \cdot 4 = 12$, which indicates the high efficiency of our GCA. We also use $\mathbb{C}^4_{15,16,1,2}$ and $\mathbb{C}^4_{3,4,5,6}$ to verify $\mathbb{C}^4_{1,2,3,4}$, and $\mathbb{C}^4_{11,12,13,14}$ and $\mathbb{C}^4_{15,16,1,2}$ to verify $\mathbb{C}^4_{13,14,15,16}$.

In order to enhance the verification, we also verify $\mathbf{R}^{2-3}$ in this way. $\mathbb{R}^{2-3}_{3,4,5,6}$, $\mathbb{R}^{2-3}_{5,6,7,8}$, $\mathbb{R}^{2-3}_{7,8,9,10}$, $\mathbb{R}^{2-3}_{9,10,11,12}$ and $\mathbb{R}^{2-3}_{11,12,13,14}$ calculated from Table I, II and VI are shown in Table VIII. An verification procedure is given in Example 4.

Example 4: A verification example is given in Fig. 8, Two $\mathbf{R}^{2-3}$ *rings* $R^{2-3}_{3,4,5,6} = (198, 127, 40, 125)$ *and* $R^{2-3}_{7,8,9,10} = (8, 61, 188, 250)$ *are used to verify ring* $R^{2-3}_{5,6,7,8} = (8, 61, 235, 135)$. *Since the verification is satisfied, the attacker or evaluator can get a correct combination* $(4, 125, 8, 61)$ *including 4 key bytes* $k_5 \sim k_8$. *If* 118 *in* ξ_8 *is used,* $(4, 125, 8, 118)$ *is also an* $\mathbf{R}^{2-3}$ *ring. However, there is no* $\mathbb{R}^{2-3}_{7,8,9,10}$ *rings beginning with* $k_7 = 8$ *and* $k_8 = 118$. *So, the combination* $(4, 125, 8, 118)$ *doesn't pass the verification.*

Finally, there are only a possible $\mathbf{R}^{2-3}$ ring $(40, 125, 8, 61)$, $(235, 135, 102, 240)$ for these two verified $\mathbf{R}^{2-3}$ rings $\mathbb{R}^{2-3}_{5,6,7,8}$ and $\mathbb{R}^{2-3}_{9,10,11,12}$ respectively (as shown in Table X). The attacker only has to enumerate $(40, 125, 8, 61, 235, 135, 102, 240)$ for key bytes $(k_5, k_6, k_7, k_8, k_9, k_{10}, k_{11}, k_{12})$. Actually, this only remaining combination corresponds to the correct key bytes $k_5 \sim k_{12}$. So, compared to $\mathbf{C}^4$ chains, $\mathbf{R}^{2-3}$ rings are more powerful. It can quickly delete impossible key combinations when Thr_k and Thr_Δ are reasonable.

TABLE VII

$\mathbf{C}^4$ CHAINS OF THE $3^{rd} \sim 14^{th}$ KEY BYTES

$\mathbb{C}^4_{3,4,5,6}$				$\mathbb{C}^4_{5,6,7,8}$				$\mathbb{C}^4_{7,8,9,10}$				$\mathbb{C}^4_{9,10,11,12}$				$\mathbb{C}^4_{11,12,13,14}$			
198	127	40	25	40	125	8	61	8	61	235	135	235	135	102	240	102	240	115	139
198	127	40	125	40	125	8	123	8	61	235	23	235	135	102	40	102	240	115	141
98	219	16	173	40	125	8	118	8	61	235	145	235	135	102	4	102	240	115	17
98	219	16	209	40	125	8	121	8	61	235	250	235	135	102	64	102	240	115	38
24	172	251	174	40	125	8	146	8	61	6	250	235	135	190	40	102	240	53	96
21	172	251	174	40	125	46	27	8	61	6	23	235	135	190	240	102	240	241	207
				40	125	106	27	8	61	188	87	7	22	190	40	102	240	47	17
				40	125	196	123	8	61	188	250	7	22	190	240	102	240	153	17
				251	174	7	121	8	118	6	250	7	22	2	76	190	240	115	139
				251	174	7	123	8	118	6	23	244	22	190	40	190	240	115	141
				251	174	7	118	8	146	6	250	244	22	190	240	190	240	115	17
				97	52	103	27	8	146	6	23	244	22	2	76	190	240	115	38
				5	52	103	27	30	146	6	250	188	87	190	40	190	240	53	96
								30	146	6	23	188	87	190	240	190	240	241	207
								76	61	235	135	188	87	194	84	190	240	47	17
								76	61	235	23	188	87	194	76	190	240	153	17
								76	61	235	145								
								76	61	235	250								
								76	61	6	250								
								76	61	6	23								
								76	61	188	87								
								76	61	188	250								
								7	118	6	250								
								7	118	6	23								

TABLE VIII

PART OF $\mathbf{R}^{2-3}$ RINGS OF THE $3^{rd} \sim 14^{th}$ KEY BYTES

$\mathbb{R}^{2-3}_{3,4,5,6}$				$\mathbb{R}^{2-3}_{5,6,7,8}$				$\mathbb{R}^{2-3}_{7,8,9,10}$				$\mathbb{R}^{2-3}_{9,10,11,12}$				$\mathbb{R}^{2-3}_{11,12,13,14}$			
198	127	40	125	40	125	8	61	8	61	188	250	235	135	102	240	102	240	115	139
198	127	40	25	40	125	8	118	8	61	6	250	235	135	102	64	102	240	115	38
21	172	251	174	40	125	8	123	8	61	6	23					102	240	115	141
				40	125	8	146	8	61	235	135								
				40	125	196	123	8	61	235	250								
								8	61	235	145								
								8	61	235	23								

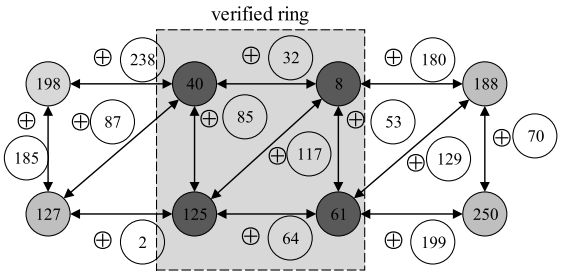

Fig. 8. Two $\mathbf{R}^{2-3}$ rings $R^{2-3}_{3,4,5,6} = (198, 127, 40, 125)$ and $R^{2-3}_{7,8,9,10} = (8, 61, 188, 250)$ are used to verify ring $R^{2-3}_{5,6,7,8} = (8, 61, 235, 135)$.

TABLE IX

VERIFIED $\mathbf{C}^4$ CHAINS OF THE $5^{th} \sim 8^{th}$ AND $9^{th} \sim 12^{th}$ KEY BYTES

$\mathbb{C}^4_{5,6,7,8}$				$\mathbb{C}^4_{9,10,11,12}$			
40	125	8	61	188	87	190	240
40	125	8	118	235	135	102	240
40	125	8	146	235	135	190	240
251	174	7	118				

TABLE X

THE VERIFIED $\mathbb{R}^{2-3}_{5,6,7,8}$ AND $\mathbb{R}^{2-3}_{9,10,11,12}$ RINGS

$\mathbb{R}^{2-3}_{5,6,7,8}$				$\mathbb{R}^{2-3}_{9,10,11,12}$			
40	125	8	61	235	135	102	240

The attacker or evaluator can also put forward different levels of requirements for verification according to the size of thresholds Thr_k and Thr_Δ. Larger thresholds may need higher level of constraints. What the attacker should take into consideration is that the combination of smaller groups may bring in greater computation. Therefore, the full key had better not be divided into very small groups (pieces). For example, dividing the 16 bytes key in the first round of AES-256 into $3 \sim 5$ groups may be a good decision.

V. EXPERIMENTAL RESULTS

In this paper, we also use our group verification chain proposed in [19] to re-rank the outputs of CPA. In this case, the average positions of correct key bytes are closer to the top of sequences $\{\xi_j | j = 1, 2, \cdots, 16\}$. Another advantage of group verification chain is, if the threshold Thr_k is reasonable, the possibility that the correct key bytes fall outside the threshold is reduced. In fact, our algorithm does not take up too much memory space. In order to quickly get the experimental results, we run 4 MATLAB main programs simultaneously on our desktop computer. Each of them takes up less than 500MB memory. We compare the attack efficiency of Test of Chain (TC), FTC, $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ under different Thr_Δ, Thr_k and different numbers of power traces.

In this paper, we use the remaining key candidate space $\mathsf{N_e}$ after GCA to represent the ability of the cryptographic devices to resist exhaustion. $\mathsf{N_e}$ here denotes the number of possible key candidates needs to be enumerated (as introduced in Section II-D). It provides security measure for the evaluators.

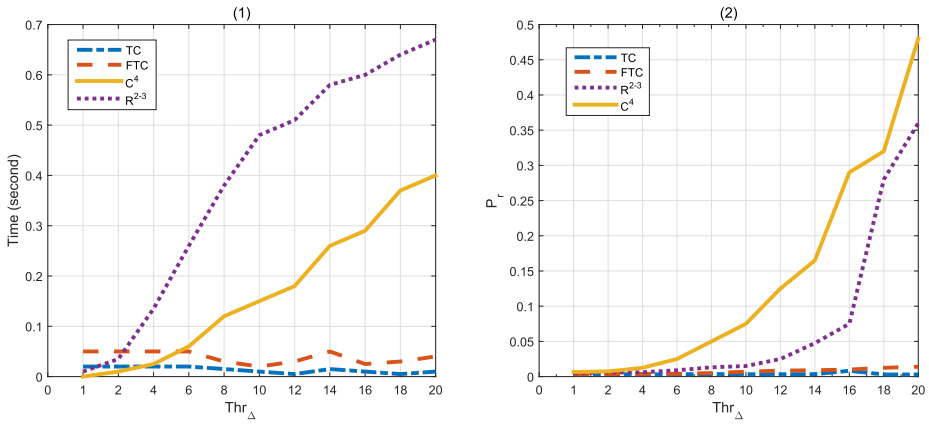

Fig. 9. Time consumption and $\mathsf{P_r}$ of 4 schemes under different Thr_Δ.

The premise of performing collision attack is that the attacker needs to choose Thr_k. This means that he can not recover the key with a probability of 1. The larger the Thr_k, the greater the probability of correct key falling within the threshold after impossible key combinations deletion. We use the symbol $\mathsf{P_r}$ to present this probability. We also use another symbol $\mathsf{P_{thr}}$ to present the probability of the correct key falling within the threshold Thr_k. For example, if we repeat each experiment 200 times and there are 165 times that the correct key falls within Thr_k after GCA, then $\mathsf{P_r} = 82.5\%$. Since collision attack requires correct key bytes and their corresponding Δ-s to be within the thresholds Thr_k and Thr_Δ simultaneously, the attacker's probability of recovering the key should be smaller than $\mathsf{P_r}$. Since all collisions may be not within thresholds simultaneously, partial key can not be recovered. So, $\mathsf{P_r} < \mathsf{P_{thr}}$.

A. Experimental Results Under Different Thresholds Thr_Δ

In this subsection, we compare our GCA schemes with TC proposed in [2] and FTC proposed by Wang *et al.* in [27]. Each experiment below is repeated 200 times. Let $\mathsf{N_p}$ denote the number of power traces used in each repetition. We divide the 16 bytes full key into 4 groups, $\mathbf{C}^4$ chain and $\mathbf{R}^{2-3}$ ring are used. Here $\mathsf{N_p}$ is set to 100 and Thr_k is set to 16 (the candidate space here is 2^{64}). If Thr_Δ is from 1 to 20, the time consumption and $\mathsf{P_r}$ of the 4 schemes TC, FTC, $\mathbb{C}^4$, and $\mathbf{R}^{2-3}$ are shown in Fig. 9. Since Thr_Δ of TC and FTC is set to 1 in [2] and [27], and no practical schemes for larger Thr_Δ are given in these two papers. Here we set Thr_Δ of TC and FTC to 1. The time consumption of TC and FTC increases very slowly when Thr_Δ is from 2 to 20. It is still less than 0.025 second when $Thr_\Delta = 20$. Compared to TC and FTC, our $\mathbb{C}^4$ and $\mathbf{R}^{2-3}$ consume more time, nearly 0.4 second is needed when $Thr_\Delta = 20$. It increases rapidly if both Thr_k and Thr_Δ increase. For example, if $Thr_k = 32$ and $Thr_\Delta = 26$, it may need several minutes for these two schemes.

$\mathsf{P_r}$ of the 4 schemes TC, FTC, $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ is very low under small threshold Thr_Δ. For example, if $Thr_\Delta = 2$, the $\mathsf{P_r}$ of these 4 schemes is about 0.02, 0.05, 0.035 and 0.05 respectively. With the increase of Thr_Δ, more correct Δ-s fall within the threshold, $\mathsf{P_r}$ increases. When Thr_Δ reaches 20, $\mathsf{P_r}$ of these 4 schemes is about 0.01, 0.04, 0.675 and 0.405 respectively. Since some correct Δ-s fall out of Thr_Δ, $\mathsf{P_r}$ of $\mathbf{C}^4$ is higher than that of $\mathbf{R}^{2-3}$.

When Thr_Δ is from 1 to 20, $\mathsf{N_e}$ is shown in Fig. 10. $\log_2(\mathsf{N_e})$ is the logarithmic operation. For example, if $\mathsf{N_e} = 2^{16}$, $\log_2(\mathsf{N_e}) = 16$. With the increase of Thr_Δ, $\mathsf{N_e}$ of $\mathbf{C}^4$ scheme grows very rapidly. When Thr_Δ equals 2, the attacker only has to consider $2^0 \sim 2^6$ key candidates. However, when Thr_Δ reaches 20, he has to consider $2^{17} \sim 2^{30}$ key candidates. Larger thresholds mean larger key candidate space. However, $\mathbf{R}^{2-3}$ with more strict constraints has fewer key candidates to be considered. When Thr_Δ reaches 20, $\mathbf{R}^{2-3}$ scheme only needs to consider almost 390 key candidates. Compared to $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ solutions, FTC appears to be random since Thr_Δ is set to 1, $2^{20} \sim 2^{90}$ key candidates may need to be considered, which may far beyond exhaustion (as shown in Fig. 10). $\mathsf{N_e}$ of TC is usually very small, which we don't give in Fig. 10, Fig. 12 and Fig. 14.

Larger $\mathsf{N_e}$ is ultimately reflected in computing and storage power. The upper bound of $\mathsf{N_e}$ is 2^{30} in Fig. 10. According to this bound, each $\mathbf{C}^4$ set has an average of $2^{7.5}$ possible combinations. Referring to Section IV, we know that the upper bound of possible combinations of 8 $\mathbf{C}^4$ sets is $8 * 2^{7.5} = 2^{11.5}$. Obviously, this only lead to very small storage overhead. $\mathbf{R}^{2-3}$ based on $\mathbf{R}^3$ has less possible combinations, of which the storage overhead is smaller.

$\mathsf{P_r}$ of $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ continues to increase in Fig. 9(2). The attacker can increase Thr_Δ to get larger $\mathsf{P_r}$. He can obtain an optimal value of Thr_Δ when $\mathsf{P_r}$ is stable. In this case, $\mathsf{P_r}$ approaches $\mathsf{P_{thr}}$. If $\mathsf{P_r}$ is still far smaller than 1, the attacker needs to enlarge Thr_k.

B. Experimental Results Under Different Thresholds Thr_k

When average 100 power traces are used and Thr_Δ of $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ is set to 14 (Thr_Δ of TC and FTC is always set to 1, since no practical schemes for larger threshold were given in [2] and [27]), if the Thr_k is set to from 2 to 22 (the key candidate space is $2^{16} \sim 2^{72}$), the time consumption and $\mathsf{P_r}$ of these 4 schemes are shown in Fig. 11. Like time consumption in Fig. 9, TC and FTC are performed very quickly, changing Thr_k does not bring too much computation. When $Thr_\Delta = 14$

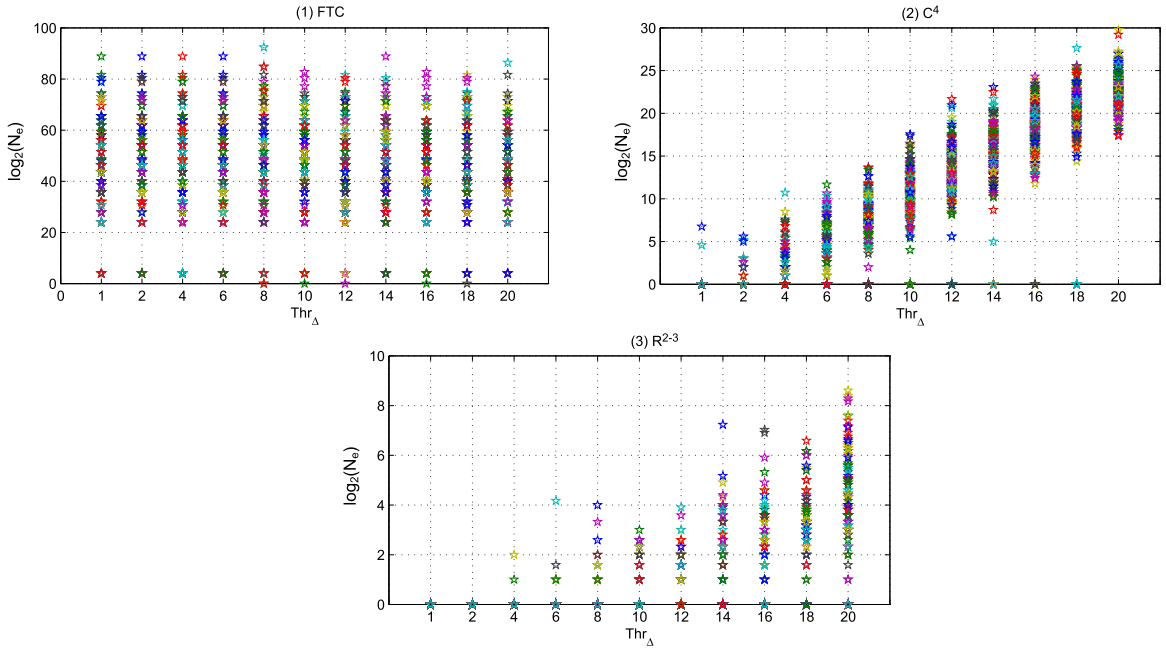

Fig. 10. Different $\mathsf{N_e}$ of 3 schemes when Thr_Δ is from 2 to 20.

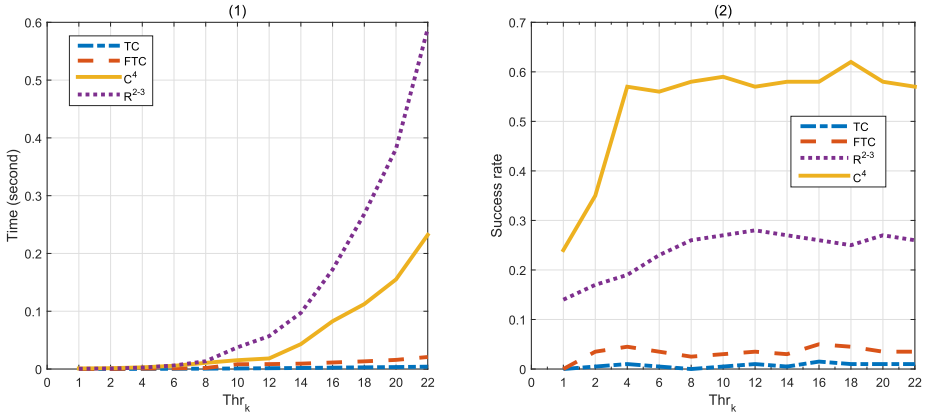

Fig. 11. Time consumption and $\mathsf{P_r}$ of 4 schemes under different Thr_k.

and $Thr_k = 16$, about 0.0483 and 0.0925 second are used. When $Thr_k = 14$ and $Thr_\Delta = 16$, about 0.0734 second and 0.1616 second are used. Enlarging Thr_Δ consumes more time than enlarging Thr_k, more key candidates need to be exhausted (as shown in Fig. 10 and Fig. 12).

$\mathsf{P_r}$ of $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ reaches the highest when Thr_k is 8 and 4 respectively. However, $\mathsf{P_r}$ of TC and FTC is still very low. Increasing Thr_k does not significantly improve the $\mathsf{P_r}$ (as shown in Fig. 11). We think that $Thr_k = 8$ and 4 are good threshold for $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ under the condition that $\mathsf{N_p} = 100$ and $Thr_\Delta = 14$. In fact, CPA is more efficient than CCA in our experiments, this makes the correct key bytes k_a and k_b rank closer to the top of ξ_a and ξ_b than $\Delta_{(k_a,k_b)}$ in Δ sequence. $\mathsf{P_r}$ is slightly increased with Thr_k. This also indicates that the probability can not be significantly improved by only increasing Thr_k.

In fact, when Thr_k increase to a certain degree, almost all the correct key bytes fall into threshold. When $Thr_k = 8$, $\mathsf{P_{thr}}$ approaches 1.00. However, $Thr_\Delta = 14$ is not ideal, there are still a lot of correct Δ-s fall outside of it. This makes the corresponding collisions impossible. In this case, too many correct key bytes within Thr_k can not be recovered using $\mathbf{R}^{2-3}$ scheme. The $\mathsf{P_r}$ of $\mathbf{R}^{2-3}$ is smaller than that of $\mathbf{C}^4$. Similar conclusions can be drawn from Fig. 9. The attacker needs to increase Thr_Δ to make more correct Δ-s fall into threshold.

The complexity of the 3 schemes FTC, $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ is very different. Like the experimental results shown in Fig. 10, $\mathsf{N_e}$ of FTC seems to be random, enlarging Thr_k does not reduce the number of key candidates. A large number of key candidates are left in many experiments. This makes the key exhaustion unreachable (e.g. larger than 2^{60}). Compared with experimental results shown in Fig. 10, enlarging Thr_k does not result in higher $\mathsf{P_r}$ but brings in more computation, while the attacker needs to enumerate more candidates. So, it will be better for the attacker to choose a reasonable Thr_k. When $Thr_k = 2$, $\mathsf{N_e}$ of $\mathbf{C}^4$ scheme is $2^0 \sim 2^6$. When $Thr_k = 22$,

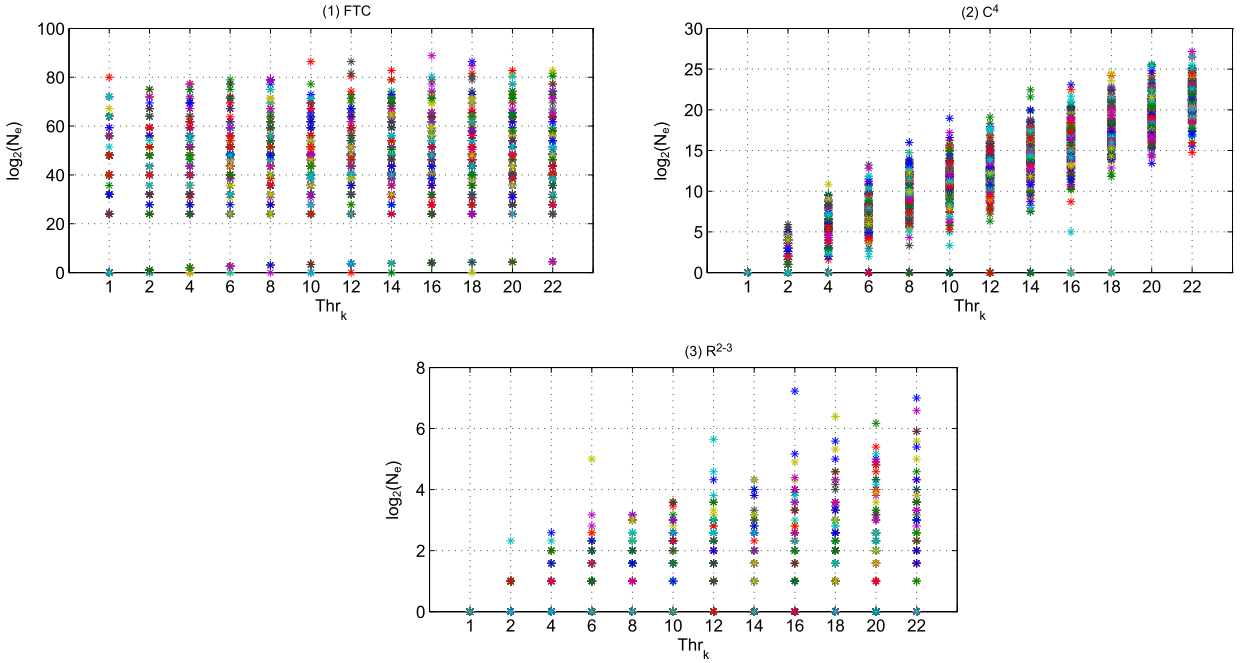

Fig. 12. Different $\mathsf{N_e}$ of 4 schemes when Thr_k is from 2 to 22.

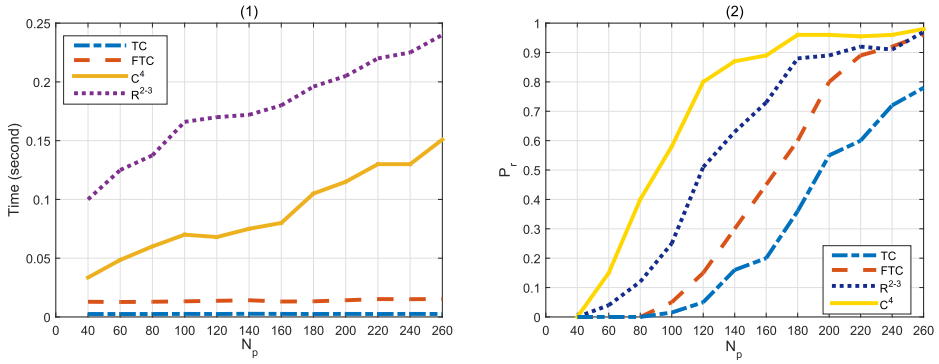

Fig. 13. Time consumption and $\mathsf{P_r}$ of 4 schemes under different numbers of power traces.

it becomes $2^{14} \sim 2^{26}$. The attacker needs less storage space than $8 * 2^7 = 2^{10}$. However, our $\mathbf{R}^{2-3}$ scheme only needs to exhaust less than 2^7 possible candidates, of which the storage requirement can be ignored. So, the attacker can easily exhaust the correct key.

It is worth noting that when both Thr_k and Thr_Δ are small, the attacker can not simply pursue complex GCA schemes to remove more impossible key combinations. Since Δ-s within Thr_Δ are nearly fixed, more stringent constraints will make more correct key bytes within Thr_k be deleted. If the attacker enlarges Thr_k and Thr_Δ, and the number of remaining key candidates of $\mathbf{R}^{2-3}$ is still very large, then the state-of-art key enumeration solutions can be used to search the entire key. In order to reduce $\mathsf{N_e}$, the attacker can choose a more strictly constrained scheme under large thresholds.

C. Experimental Results Under Different Numbers of Power Traces

We also compare the time consumption and $\mathsf{P_r}$ under different numbers of power traces. Thr_k and Thr_Δ are set to 16 and 14 respectively. The experimental results are shown in Fig. 13. The time consumption of TC is almost the same when $\mathsf{N_p}$ is from 40 to 260 compared to $0.0066 \sim 0.0114$ second used in FTC. Compared to TC and FTC, our $\mathbf{C}^4$ and R^{2-3} spend $0.0345 \sim 0.1534$ and $0.1014 \sim 0.2405$ second respectively. This indicates that, with the increase of $\mathsf{N_p}$, more correct key bytes and Δ-s are in Thr_k and Thr_Δ, more time is needed to construct chains and rings.

Compared with increasing Thr_k and Thr_Δ, it's more efficient to increase the number of power traces. The $\mathsf{P_r}$ of 4 schemes increases rapidly with $\mathsf{N_p}$ in each repetition (as shown in Fig. 13(2)). When $\mathsf{N_p} = 100$, the $\mathsf{P_r}$ of these 4 schemes is 0.0150, 0.0500,0.5800 and 0.2600 respectively. They become 0.0550, 0.1500, 0.8000 and 0.5200 when $\mathsf{N_p} = 120$. The $\mathsf{P_r}$ of these 4 schemes reaches 0.7850, 0.9650, 0.9850 and 0.9750 respectively when $\mathsf{N_p} = 260$. Actually, the $\mathsf{P_r}$ of our $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ schemes are much higher than that of TC and FTC, which indicates that our $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ schemes can significantly improve the efficiency of TC and FTC.

The $\mathsf{N_e}$ of $\mathbf{C}^4$ and $\mathbf{R}^{2-3}$ under different $\mathsf{N_p}$ is shown in Fig. 14, which changes much smaller compared to different thresholds Thr_k and Thr_Δ(as shown in Fig. 10 and Fig. 12).

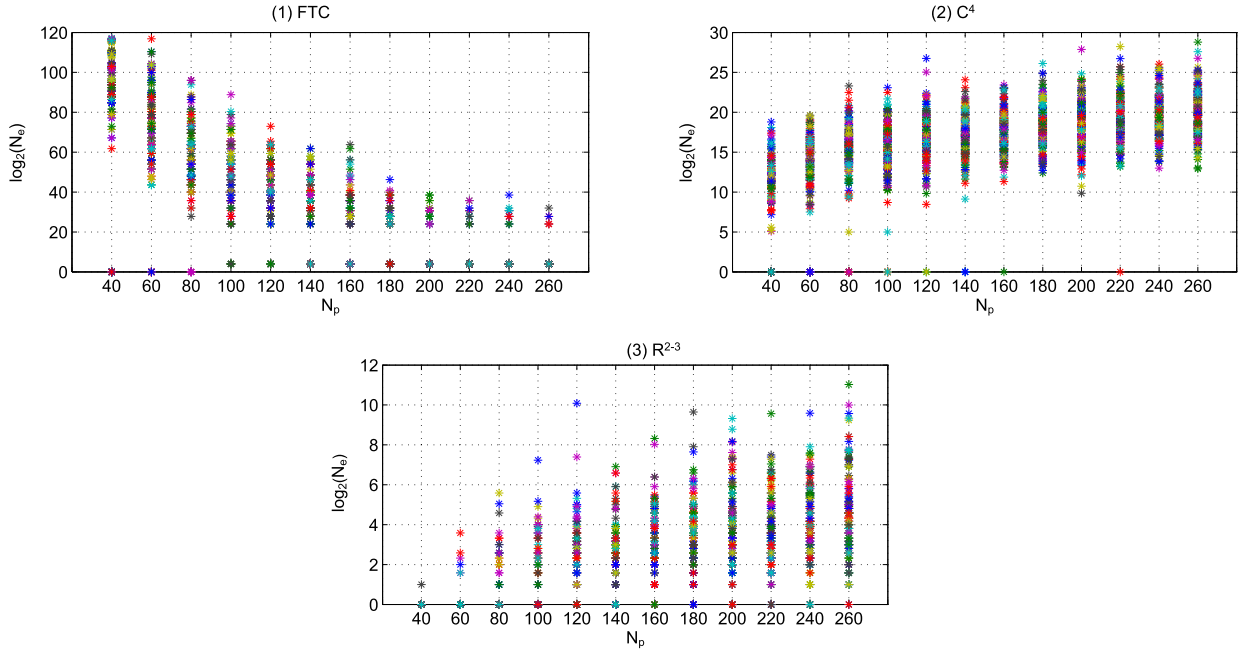

Fig. 14. Different $\mathsf{N_e}$ of 3 schemes under different numbers of power traces.

$\mathsf{N_e}$ of FTC decreases with increase of $\mathsf{N_p}$. When average 40, 80, 120, 160 and 200 power traces are used, the range of $\mathsf{N_e}$ is $2^{60} \sim 2^{120}$, $2^{30} \sim 2^{100}$, $2^{24} \sim 2^{80}$, $2^{24} \sim 2^{64}$ and $2^{4} \sim 2^{40}$ respectively. This indicates that, with more power traces being used in each repetition, more correct key bytes and Δ-s locate in the front of sequences output by side channel distinguishers. The advantages of TC and FTC begin to appear.

$\mathsf{N_e}$ of $\mathbf{C}^4$ also increases with $\mathsf{N_p}$, but not so rapidly as the ones in Fig. 10 and Fig. 12. When $\mathsf{N_p}$ equals 40 and 260, the attacker has to exhaust $2^5 \sim 2^{20}$ and $2^{12} \sim 2^{28}$ key candidates respectively. If $\mathbf{R}^{2-3}$ is used, $2^0 \sim 2^1$ and $2^0 \sim 2^{11}$ key candidates need to be considered. However, similar to $\mathsf{N_e}$ in Fig. 9 and Fig. 11, $\mathsf{N_e}$ of $\mathbf{R}^{2-3}$ is smaller than 2^8 in most of repetitions. This also indicates that when Thr_k and Thr_Δ are fixed, the number of keys considered by FTC is decreasing. However, $\mathsf{N_e}$ corresponding to $\mathbf{C}^4$ gradually increases, which is less obvious than Fig. 10. This is mainly due to the fact that more correct key bytes and Δ-s fall within thresholds Thr_k and Thr_Δ when $\mathsf{N_p}$ increases. Similar to the conclusion drawn from Sections V-A and V-B, GCA schemes only brings in very small storage cost. The more complex the GCA schemes, the smaller the storage cost. However, the time consumption of them increases significantly due to the loops in algorithm. Therefore, it is important for the attacker to quickly implement complex GCA schemes.

VI. Conclusions and Open Problems

Cryptographic algorithms always have very large key candidate space to resist exhaustive attacks. If the attacker only acquires a small number of power traces, only partial key bytes are ranked on the top of side channel distinguishers. In this case, key enumeration, an optimal tool of key exhaustion, enumerates the key candidates from the most possible one to the least possible one. However, both key exhaustion and enumeration are very time-consuming, the attacker has to spend several weeks or months to enumerate a huge key candidate space (e.g. 2^{45}). Huge storage and computation power requirements even make them infeasible.

In this paper, we aim at recover the key from a very large space which key exhaustion and enumeration are unreachable, a divide and conquer strategy named GCA is proposed. The huge key candidate space is divided into several groups, the intra-group and inter-group collision attack are performed to delete impossible key combinations. Compared with the classical key exhaustion and enumeration performed on all key candidates, the attacker only needs to consider a subset of key candidates much smaller than the original one.

It is worth noting that we only simply compare our key chain and key ring based GCA schemes in this paper. The attacker can propose more efficient GCA schemes. In principle, more collisions means that the scheme can delete more impossible key candidates. Fault tolerance needs to be introduced into GCA to reduce the probability of accidentally deleting the correct subkey bytes. The problem that how to delete the impossible combinations under large thresholds Thr_k and Thr_Δ is also very interesting. Since compared to 2^{128}, the subspace $2^{64} \sim 2^{71}$ in this paper is still very small. With the increase of both two thresholds, more and more key candidates satisfy the conditions, the construction and the implementation of complex GCA schemes also become more important.

There are still several open problems of GCA. We list them as follows:

1) **Formalization of Thr_k and Thr_Δ.** In the Section V, we discuss the experimental results under different thresholds Thr_k and Thr_Δ. Reasonable thresholds need continuous testing. We leave the problem that how to determine and formulate Thr_k and Thr_Δ in our

GCA. Actually, CCA determines the size of Thr_Δ, CPA determines the size of Thr_k. Wiemers and Klein proposed very interesting independent parallel work in [28], which might solve this problem very well. They obtained the distinguishable correlation coefficients distributions of correct key and wrong keys from CCA. The correlation coefficients distributions of correct key and wrong keys from CPA can also calculated in this way. The reasonable size of Thr_k and Thr_Δ may be found.

2) **GCA in key enumeration**. The problem that how to efficiently introduce GCA into the existing key enumeration solutions is also very interesting. GCA can be used as an optimal pre-processing tool for key enumeration. For example, the original key candidate space is 2^{70} and becomes 2^{35} after performing our GCA. The attacker still needs to enumerate the remaining 2^{35} key candidates. Enumeration efficiency relates to the specific key enumeration schemes, such as histograms-based enumeration [21] and Optimal Key Enumeration (OKE) [25]. Recently, there were several papers discussing key enumeration in parallel (see [14], [8]). Our GCA is also very easy to perform in parallel since the 16 bytes key of AES-256 used in this paper is divided into several independent groups (other algorithm like DES can be also attacked using GCA). Each group can be calculated independently. The efficient GCA parallel algorithms are also an interesting problem.
3) **GCA in evaluation**. Standaert *et al.* proposed a side channel evaluation framework in [24]. Guessing entropy and success rate became the common criteria of side channel evaluation. Recent years, researchers also measured the security of devices from the upper bound and lower bound of guessing entropy and success rate (see [9], [7]). The work of Wiemers and Klein proposed in [28] showed how to reduce the remaining entropy of the key in an attack. In this paper, we only use the remaining key candidate space to present the difficulty of key exhaustion or enumeration. It can be seen from Fig. 10, Fig. 12 and Fig. 14 that the number of remaining key candidates changes dramatically in our repetitions, it is difficult to give the theoretical bounds. So, how to introduce the above evaluate tools into GCA to evaluation the security bounds of GCA is also meaningful.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments. Most of them helped to improve the paper.

REFERENCES

[1] *DPA Contest*. Accessed: Sep. 2013. [Online]. Available: http://www.dpacontest.org/home/
[2] A. Bogdanov and I. Kizhvatov, "Beyond the limits of DPA: Combined side-channel collision attacks," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1153–1164, Aug. 2012.
[3] A. Bogdanov, I. Kizhvatov, K. Manzoor, E. Tischhauser, and M. Witteman, "Fast and memory-efficient key recovery in side-channel attacks," in *Proc. 22nd Int. Conf. Sel. Areas Cryptogr. (SAC)*, Sackville, NB, Canada, Aug. 2015, pp. 310–327.
[4] A. Bogdanov, I. Kizhvatov, and A. Pyshkin, "Algebraic methods in side-channel collision attacks and practical collision detection," in *Proc. 9th Int. Conf. Cryptol. Prog. Cryptol.*, Kharagpur, India, Dec. 2008, pp. 251–265.
[5] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. 6th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Cambridge, MA, USA, Aug. 2004, pp. 16–29.
[6] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. 4th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Redwood Shores, CA, USA, Aug. 2002, pp. 13–28.
[7] M. O. Choudary and P. G. Popescu, "Back to Massey: Impressively fast, scalable and tight security evaluation tools," in *Proc. 19th Int. Conf. Cryptograph. Hardw. Embedded Syst. (CHES)*, Taipei, Taiwan, Sep. 2017, pp. 367–386.
[8] L. David and A. Wool, "A bounded-space near-optimal key enumeration algorithm for multi-subkey side-channel attacks," in *Proc. Cryptograph. Track RSA Conf. Topics Cryptol. (CT-RSA)*, San Francisco, CA, USA, Feb. 2017, pp. 311–327.
[9] A. Duc, S. Faust, and F.-X. Standaert, "Making masking security proofs concrete—Or how to evaluate the security of any leaking device," in *Proc. 34th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Sofia, Bulgaria, Apr. 2015, pp. 401–429.
[10] F. Durvaux and F.-X. Standaert, "From improved leakage detection to the detection of points of interests in leakage traces," in *Proc. 35th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Vienna, Austria, May 2016, pp. 240–262.
[11] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis: A generic side-channel distinguisher," in *Proc. 10th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Washington, DC, USA, Aug. 2008, pp. 426–442.
[12] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. 19th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 1999, pp. 388–397.
[13] S. Mangard, E. Oswald, and F.-X. Standaert, "One for all–all for one: Unifying standard differential power analysis attacks," *IET Inf. Secur.*, vol. 5, no. 2, pp. 100–110, 2011.
[14] D. P. Martin, J. F. O'Connell, E. Oswald, and M. Stam, "Counting keys in parallel after a side channel attack," in *Proc. 21st Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Auckland, New Zealand, 2015, pp. 313–337.
[15] A. Moradi, "Statistical tools flavor side-channel collision attacks," in *Proc. 31st Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Cambridge, U.K., 2012, pp. 428–445.
[16] A. Moradi, S. Guilley, and A. Heuser, "Detecting hidden leakages," in *Proc. 12th Int. Conf. Appl. Cryptogr. Netw. Secur. (ACNS)*, Lausanne, Switzerland, 2014, pp. 324–342.
[17] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Proc. 12th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Santa Barbara, CA, USA, Aug. 2010, pp. 125–139.
[18] M. Nassar, Y. Souissi, S. Guilley, and J. L. Danger, "RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs," in *Proc. Design, Automat. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2012, pp. 1173–1178.
[19] C. Ou, Z. Wang, D. Sun, X. Zhou, and J. Ai, "Group verification based multiple-differential collision attack," in *Proc. 18th Int. Conf. Inf. Commun. Secur. (ICICS)*, Singapore, 2016, pp. 145–156.
[20] J. Pan, J. G. J. van Woudenberg, J. I. den Hartog, and M. F. Witteman, "Improving DPA by peak distribution analysis," in *Proc. 17th Int. Workshop Sel. Areas Cryptogr. (SAC)* Waterloo, ON, Canada, Aug. 2010, pp. 241–261.
[21] R. Poussier, F.-X. Standaert, and V. Grosso, "Simple key enumeration (and rank estimation) using histograms: An integrated approach," in *Proc. 18th Int. Conf. Cryptograph. Hardw. Embedded Syst. (CHES)*, Santa Barbara, CA, USA, Aug. 2016, pp. 61–81.
[22] C. Rechberger and E. Oswald, "Practical template attacks," in *Proc. 5th Int. Workshop Inf. Secur. Appl. (WISA)*, Jeju Island, South Korea, Aug. 2004, pp. 440–456.
[23] K. Schramm, G. Leander, P. Felke, and C. Paar, "A collision-attack on AES: Combining side channel- and differential-attack," in *Proc. 6th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, Cambridge, MA, USA, Aug. 2004, pp. 163–175.
[24] F.-X. Standaert, T. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Proc. 28th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Cologne, Germany, 2009, pp. 443–461.

[25] N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F.-X. Standaert, "An optimal key enumeration algorithm and its application to side-channel attacks," in *Proc. 19th Int. Conf. Sel. Areas Cryptogr. (SAC)*, Windsor, ON, Canada, Aug. 2012, pp. 390–406.
[26] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert, "Security evaluations beyond computing power," in *Proc. 32nd Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, Athens, Greece, 2013, pp. 126–141.
[27] D. Wang, A. Wang, and X. Zheng, "Fault-tolerant linear collision attack: A combination with correlation power analysis," in *Proc. 10th Int. Conf. Inf. Secur. Pract. Exper. (ISPEC)*, Fuzhou, China, 2014, pp. 232–246.
[28] A. Wiemers and D. Klein, "Entropy reduction for the correlation-enhanced power analysis collision attack," International Association for Cryptologic Research, Tech. Rep. 2017/1079, 2018.

Changhai Ou was born in Tongren, Guizhou, China, in 1989. He received the bachelor's degree in computer and information science from the School of Computer and Information Technology, Beijing Jiaotong University, in 2013, and the Ph.D. degree in cyber security from the School of Cyber Security, University of Chinese Academy of Sciences (Institute of Information Engineering, Chinese Academy of Sciences).

Since 2018, he has been a Research Fellow with the Hardware and Embedded Systems Laboratory, School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include side channel attacks, hardware security, signal processing, leakage evaluation, and machine learning.

Zhu Wang received the Ph.D. degree from the State Key Laboratory of Information Security, Chinese Academy of Science. She is currently a Senior Engineer with the Institute of Information Engineering, Chinese Academy of Science. She is also an Associate Professor with the School of Cyber Security, University of Chinese Academy of Sciences. Her research interests include cryptographic protocol, network security, information hiding, and side channel attack.

Degang Sun is currently a Research Professor and a Ph.D. Supervisor with the Institute of Information Engineering, Chinese Academy of Sciences. He is also a Professor with the School of Cyber Security, University of Chinese Academy of Sciences. His research interests include electromagnetic leakage protection, wireless communication technology, and high security level information system protection technology.

Xinping Zhou received the bachelor's degree in mathematics from the Huazhong University of Science and Technology and the Ph.D. degree in computer science from the Institute of Information Engineering, Chinese Academy of Sciences, in 2012 and 2017, respectively. He is currently with the National Financial Card Test Center of China. His research interests include side channel attacks, cryptanalysis, and cryptographic security implementation.