Multiple-Differential Mechanism for Collision-Optimized Divide-and-Conquer Attacks

Changhai Ou, Siew-Kei Lam and Guiyuan Jiang

Hardware & Embedded Systems Lab, School of Computer Science and Engineering, Nanyang Technological University, Singapore.
CHOu@ntu.edu.sg,ASSKLam@ntu.edu.sg,gyjiang@ntu.edu.sg

Abstract. Several combined attacks have shown promising results in recovering cryptographic keys by introducing collision information into divide-and-conquer attacks to transform a part of the best key candidates within given thresholds into a much smaller collision space. However, these Collision-Optimized Divide-and-Conquer Attacks (CODCAs) uniformly demarcate the thresholds for all sub-keys, which is unreasonable. Moreover, the inadequate exploitation of collision information and backward fault tolerance mechanisms of CODCAs also lead to low attack efficiency. Finally, existing CODCAs mainly focus on improving collision detection algorithms but lack theoretical basis. We exploit Correlation-Enhanced Collision Attack (CECA) to optimize Template Attack (TA). To overcome the above-mentioned problems, we first introduce guessing theory into TA to enable the quick estimation of success probability and the corresponding complexity of key recovery. Next, a novel Multiple-Differential mechanism for CODCAs (MD-CODCA) is proposed. The first two differential mechanisms construct collision chains satisfying the given number of collisions from several sub-keys with the fewest candidates under a fixed probability provided by guessing theory, then exploit them to vote for the remaining sub-keys. This guarantees that the number of remaining chains is minimal, and makes MD-CODCA suitable for very high thresholds. Our third differential mechanism simply divides the key into several large non-overlapping "blocks" to further exploit intra-block collisions from the remaining candidates and properly ignore the inter-block collisions, thus facilitating the latter key enumeration. The experimental results show that MD-CODCA significantly reduces the candidate space and lowers the complexity of collision detection, without considerably reducing the success probability of attacks.

Keywords: MD-CODCA, CODCA, guessing theory, candidate space, key enumeration, collision attack, side-channel attack

1 Introduction

Implementations of cryptographic algorithms on devices produce unintentional leakages such as power consumption [15], electromagnetic radiation [1] and cache patterns [24], which pose security vulnerabilities to Side-Channel Attacks (SCAs). SCAs have been demonstrated successfully on various chips and devices, such as PDAs [10], desktop computers [11] and even cloud servers [14]. Existing SCAs can be divided into two general approaches: divide-and-conquer and analytical. These approaches can exploit two types of information: direct leakages and collision leakages. Analytical attacks such as collision attack [17], recover the key through solving a system of equations and exploit more leakage information than divide-and-conquer attacks, but are harder to launch.

Existing divide-and-conquer attacks, such as Correlation Power Analysis (CPA) [5] and Template Attack (TA) [7], divide the huge key candidate space into several small blocks (e.g. sub-keys) and conquer them one by one. They are often combined with key enumeration [16, 21] to avoid unnecessary guesses, but this is only feasible if the keys fall within the enumerable space. In practice, keys are often located in spaces that cannot be enumerated directly. To enhance the key recovery, several attacks introduced collision information into the divide-and-conquer attacks to transform a part of best candidates into a significantly smaller collision space. We call these attacks, Collision-Optimized Divide-and-Conquer Attacks (CODCAs). However, existing CODCAs still encounter difficulty in candidate space transformation due to inefficient collision utilization, backward fault tolerance mechanisms and time-consuming collision detection algorithms. Our work overcomes this problem by proposing a highly-efficient Multiple-Differential mechanism for CODCAs (MD-CODCA) and introducing guessing theory to enhance its key recovery and evaluations.

1.1 Related Works

Existing CODCAs introduced information from a collision attack into a divide-and-conquer attack to exploit more leakage information and transform a part of best candidates into a significantly smaller collision space. We consider TA and Correlation-Enhanced Collision Attack (CECA) [17] in this paper. CODCAs rank the candidates of each sub-key and collision value (e.g. XOR value exploited in [17]) between two sub-keys in two attacks from the most possible one to the least possible one and set thresholds for them. This means, CODCAs only consider the best candidates within thresholds, and try to recover the key from them. A collision happens if a pair of candidates of two sub-keys and their corresponding collision value are within their corresponding thresholds simultaneously. A collision chain includes one or several pairs of collisions. CODCAs only record desirable chains that satisfy the given collision conditions. They eliminate the independence between sub-keys, and leave us with a collision space composed of collision chains. This new space is much smaller than the one within threshold before CODCAs, thus significantly lowering the complexity of future key recovery.

The first CODCA called Test of Chain (TC) was proposed in [3]. TC attempts to find a long chain from the first sub-key to the last sub-key, which exploits 15 pairs of collisions when attacking AES-128. Another CODCA named Fault-Tolerant Chain (FTC) was proposed in [23], which only considers the most possible candidate of each collision value. It aims to find 15 pairs of collisions between the first sub-key and the other 15 sub-keys, and exhausts the first sub-key. Due to the insufficient usage of collision information, TC and FTC need to enumerate a large number of chains. Since the remaining sub-keys are only related to the first sub-key and are independent of each other, FTC performs fast collision detection and results in very large candidate space to enumerate. The chains of TC include all sub-keys, which facilitates the key verification. However, its space transformation is very time-consuming. Therefore, TC is feasible only in situations where thresholds of both two attacks are small.

Multiple-Differential Collision Attack (MDCA) in [2] exploits multiple differential mechanisms to launch voting when measuring the similarity of two power traces (e.g., binary voting and ternary voting based on Euclidean distance between them). MDCA is a single attack, not a combined collision attack like CODCAs. Information from CECA was also introduced into CPA in [19], and an attack named Group Verification based Multiple-Differential Collision Attack (GV-MDCA) was given. It votes a sub-key using the collisions between its candidates and the candidates of the remaining 15 sub-keys (i.e., the groups). The candidates are then ranked in descending order according to their number of votes (i.e. collisions). The differential mechanism here discards candidates that do not satisfy the required number of collisions, rather than those where the similarity is greater than a given threshold as in [2]. GV-MDCA enables more sub-keys to be ranked first, thus improving the attack efficiency. However, GV-MDCA does not alter the rank of the key due to the unchanged probabilities or scores of the candidates. Thus, MDCA and GV-MDCA have a different goal from our work.

TC and FTC only exploit 15 pairs of collisions, and the majority of 120 pairs of collisions between 16 sub-keys of AES-128 algorithm are wasted. The insufficient utilization of collision information makes it time-consuming for them to conquer the case wherein sub-keys or collision values are deeply ranked. To optimize this, Group Collision Attack (GCA) [18] was further proposed. The 16 sub-keys were divided into several big groups. A part of sub-keys of each group overlap with its former and latter groups. Intra-group collisions are used to perform the first round of chain construction within groups, and inter-group collisions are used to perform another round of chain construction among groups (see Section 2.4 for details). GCA exploits twice as many collisions as TC and FTC. Thus, GCA extends their conquerable space. However, GCA still encounters great difficulty when dealing with larger space (see Section 7 for details). Moreover, GCA has no fault tolerance strategy, and is prone to mistakenly discarding sub-keys under such strict collision conditions. This may lead to the failure in key recovery. Finally, FTC and GCA in [18, 23] set a unified threshold for all sub-keys. To ensure that all sub-keys are within the threshold, it shouldn't be smaller than the one with the deepest position. Obviously, a large number of additional undesirable candidates need to be considered, which increases the complexity of key recovery.

1.2 Our Contributions

This paper aims to exploit information from CECA to optimize the key recovery of TA. The main contributions are as follows:

- Firstly, we introduce guessing theory into TA to optimize its threshold setting according to the available computing power. This also aids in rapid estimation of the success probability and the corresponding considered guessing space.

- Secondly, a novel Multiple-Differential mechanism for CODCAs (MD-CODCA) is proposed. The first differential mechanism considers the number of candidates of each sub-key in TA under a fixed probability provided by guessing theory, and extracts several sub-keys with the fewest candidates to construct chains, thus minimizing the number of candidates. The second differential mechanism exploits these chains to vote for the candidates of the remaining sub-keys. These guarantee that the complexity of voting is almost the optimal and the remaining space is almost the minimum.

- Finally, we discover unique advantages of performing key enumeration after CODCAs. Therefore, we construct another simple yet efficient differential mechanism on the remaining candidates. Specifically, we divide the 16 sub-keys into several big non-overlapping blocks, and further exploit collisions from the remaining candidates within each of them. The collision information between blocks, which may have already been used, is properly ignored to facilitate the enumeration. Compared to the existing CODCAs, the flexible differential mechanisms of our MD-CODCA make it suitable for very large thresholds, and allow for a notable performance gap between two combined attacks.

A good CODCA should reduce candidate space as much as possible without significantly reducing the probability of successful key recovery. The proposed MD-CODCA well achieves this goal. Its success rate is close to the theoretical success probability of TA provided by guessing theory, and is significantly higher than those of the existing CODCAs.

1.3 Organization

The rest of the paper is organized as follows. The experimental setups, principles of TA and collision attack, and the existing CODCAs are introduced in Section 2. The attack flow of our MD-CODCA is given in Section 3. Guessing theory in TA including guessing model, partial guessing metrics, success probability and complexity estimation, are given in Section 4. The three differential mechanisms of our MD-CODCA are introduced in detail in Sections 5 and 6. Experiments are performed on an *AT89S52* micro-controller in Section 7 to demonstrate its practicability and efficiency. Finally, Section 8 concludes this paper.

2 Preliminaries

2.1 Experimental Setups

Our experiments are performed on the power traces leaked from an AT89S52 micro-controller. The operating frequency of the AT89S52 system is 12 MHz. We use assembly language to implement the AES-128 algorithm, and sample the leakage of its first round execution. The sampling rate of our Picoscope 3000 is set to 125 MS/s. We finally acquire 51200 power traces and perform CPA to select a Point-of-interest (POI) [9] with the highest correlation coefficient for each S-box to perform the subsequent experiments.

2.2 Template Attack

Template Attack (TA) [7] is one of the most powerful side-channel attacks. It can be divided into two stages: template construction and classification. For template construction, let $x_{j,i}^{\kappa}$ $(1 \le i \le 256; 1 \le j \le 16; 1 \le \kappa \le N)$ denote the κ-th encryption of the i-th value of the j-th plaintext byte, and $\mathbf{t}_{j,i}^{\kappa}$ denote the corresponding POIs of the intermediate value (e.g. the S-box output in the first round of AES-128). TA profiles template $(\mathbf{m}_{j,i}, \mathbf{C}_{j,i})$ satisfying:

$$\mathbf{m}_{j,i} = \frac{1}{N}\sum_{\kappa=1}^{N} \mathbf{t}_{j,i}^{\kappa} \tag{1}$$

and

$$\mathbf{C}_{j,i} = \frac{1}{N}\sum_{\kappa=1}^{N} \left(\mathbf{t}_{j,i}^{\kappa} - \mathbf{m}_{j,i}\right)\left(\mathbf{t}_{j,i}^{\kappa} - \mathbf{m}_{j,i}\right)^{\mathsf{T}}. \tag{2}$$

Here $\mathbf{m}_{j,i}$ denotes the mean power consumption vector, $\mathbf{C}_{j,i}$ denotes the noise covariance matrix and symbol “T" denotes matrix transposition.

For classification, let $\mathbf{X} = \left\{x_j^{\kappa} \middle| 1 \le j \le 16; 1 \le \kappa \le n\right\}$ denote the encrypted n plaintexts, $\mathbf{K} = \left\{k_j \middle| j = 1, \ldots, 16\right\}$ denote the key, and $\mathbf{T} = \left\{\mathbf{t}_j^{\kappa} \middle| 1 \le j \le 16; 1 \le \kappa \le n\right\}$ denote the corresponding POIs. The probability of $\mathbf{t}_j^{\kappa}$ corresponding to template $(\mathbf{m}_{j,i}, \mathbf{C}_{j,i})$ is:

$$p\left(\mathbf{t}_j^{\kappa} \middle| \mathbf{m}_{j,i}, \mathbf{C}_{j,i}\right) = \frac{e^{-\frac{\left(\mathbf{m}_{j,i}-\mathbf{t}_j^{\kappa}\right)\cdot\left(\mathbf{C}_{j,i}\right)^{-1}\cdot\left(\mathbf{m}_{j,i}-\mathbf{t}_j^{\kappa}\right)^{\mathsf{T}}}{2}}}{\sqrt{(2\cdot\pi)^{|m_{j,i}|}\,\mathbf{det}\left(\mathbf{C}_{j,i}\right)}}, \tag{3}$$

where $|\mathbf{m}_{j,i}|$ represents the size of $\mathbf{m}_{j,i}$, i.e., the number of POIs used to profile each template. The probability of a candidate to be the correct one in the improved TA given in [25], is the product of the probabilities that all n power traces used for attacks are classified into their corresponding templates according to it. If they are too many power traces in attacks and the probability products are too small to be expressed, we can exploit logarithmic function to transform the multiplication into addition.

2.3 Collision Attacks

AES-128 performs 16 parallel S-box operations in its first round. A linear collision happens if two S-boxes generate the same intermediate value:

$$\mathsf{Sbox}\left(x_{j_1} \oplus k_{j_1}\right) = \mathsf{Sbox}\left(x_{j_2} \oplus k_{j_2}\right) \tag{4}$$

as shown in Fig. 1. Here 'Sbox' denotes the S-box operation. Eq. 4 means that these two S-boxes accept the same input: $x_{j_1} \oplus k_{j_1} = x_{j_2} \oplus k_{j_2}$. In this case, this pair of collision can determine the XOR value of two sub-keys:

$$\delta_{j_1,j_2} = k_{j_1} \oplus k_{j_2} = x_{j_1} \oplus x_{j_2}. \tag{5}$$

This collision can also occur between two different plaintexts. For simplicity, we use $k_{j_1} \leftrightarrow k_{j_2}$ to represent this pair of collision.

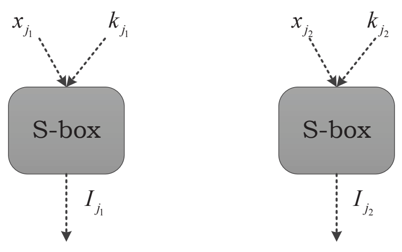

Figure 1: A pair of collision between two S-boxes in AES-128.

A specific implementation of Correlation-Enhanced Collision Attack (CECA) was given in Algorithm 2 in [23]. Taking the collision between the first and second S-boxes of AES-128 as an example, CECA divides their power traces into 256 classes according to their plaintext byte values, and calculates the mean power consumption vector of each class. To distinguish them from the templates of TA in Eq. 1, we use $\bar{\mathbf{t}}_{j,i}$ $(1 \leq j \leq 16; 0 \leq i \leq 255)$ to represent the mean power consumption of plaintext byte value i of the j-th sub-key. The classic CECA then computes the correlation coefficient:

$$\rho\left\{\left(\bar{\mathbf{t}}_{1,x_1}, \bar{\mathbf{t}}_{2,x_1\oplus\delta_{1,2}}\right) \middle| x_1 = 0, 1, 2, \cdots, 255\right\} \tag{6}$$

under a guessing $\delta_{1,2}$ (see Eq. 5). It is noteworthy that CECA's performance in [17] is usually far inferior to the optimized divide-and-conquer attack TA. Therefore, we need to consider more candidates for each δ than sub-keys. Here we exploit the templates profiled in TA and 4^{th}-order Minkowski distance to enhance CECA as:

$$\sqrt[4]{\sum_{\kappa=1}^{n}\left(\mathbf{t}_2^{\kappa} - \mathbf{m}_{1,x_1^{\kappa}\oplus\delta_{1,2}}\right)^4}, \tag{7}$$

since higher-order distance means higher performance, but the performance improved by Minkowski distance above the 4^{th} order is very small. If we get ν XOR values:

$$\begin{cases} \delta_{j_1,j_2} & = & k_{j_1} \oplus k_{j_2}, \\ \delta_{j_3,j_4} & = & k_{j_3} \oplus k_{j_4}, \\ & \vdots & \\ \delta_{j_{2\nu-1},j_{2\nu}} & = & k_{j_{2\nu-1}} \oplus k_{j_{2\nu}}, \end{cases} \tag{8}$$

in CECA, then we obtain a collision system as defined in [23]. Due to insufficient exploitation of collision information, a collision system often includes several chains, where each contains several non-overlapping sub-keys. Therefore, to recover the key, we should exhaust a sub-key for each chain.

2.4 Collision-Optimized Divide-and-Conquer Attacks

Traditional divide-and-conquer attacks conquer the sub-keys one at a time. Collision-Optimized Divide-and-Conquer Attacks (CODCAs) introduce collision information into them and eliminate the independency of sub-keys. Specifically, they rank the candidates of each sub-key and each δ from the best one to the worst one separately, and exploit the candidates of δ-s within threshold to transform a part of the best candidates in the divide-and-conquer attack into a much smaller collision space, and try to recover the key from it. Let $\tau_k = 10$ and $\tau_d = 10$ denote two thresholds for the improved TA and CECA in Sections 2.2 and 2.3 respectively, which means only the best 10 candidates of each sub-key and collision value δ_{j_1,j_2} are considered. Experimental results performed on randomly extracted 240 power traces are shown in Tables 1 and 2.

Test of Chain (TC), the first CODCA given in [3], attempts to find a long chain from the first sub-key to the 16-th sub-key including 15 pairs of collisions: $k_1 \leftrightarrow k_2$, $k_2 \leftrightarrow k_3$, ..., $k_{14} \leftrightarrow k_{15}$ and $k_{15} \leftrightarrow k_{16}$. The collision-pairs among the first three sub-keys built from Tables 1 and 2 are given in Table 3. TC exploits collisions between k_1 and k_2, and k_2 and k_3, and its results are shown in Table 4. To recover the sub-keys 212, 153 and 7, we only need to consider up to 12 chains, which demonstrates its benefits. The chain space will be further reduced compared to the original space of TA within τ_k when considering more sub-keys.

Table 2: The ranked candidates of δ-s within threshold τ_d in the improved CECA.

k_1	k_2	k_3
75	204	17
72	182	52
212	14	105
169	150	44
62	82	100
35	58	222
167	41	142
191	236	169
130	153	77
128	8	128

$\delta_{1,2}$	$\delta_{1,3}$	$\delta_{2,3}$
77	84	136
43	141	181
71	197	173
116	178	192
72	176	251
136	10	253
193	92	175
49	182	243
235	248	134
56	189	20

Table 4: Chains of TC include $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$.

$k_1 \leftrightarrow k_2$		$k_1 \leftrightarrow k_3$		$k_2 \leftrightarrow k_3$	
212	236	75	142	182	77
212	153	212	17	82	169
62	182	212	105	236	17
35	8	212	44	236	44
167	150	212	100	236	100
128	8	212	222	153	17
		212	128	153	52
		62	52	153	44
		62	142	153	100
		167	17	8	142
		130	52	8	128
		130	222		

k_1	k_2	k_3
212	236	17
212	236	44
212	236	100
212	**153**	**17**
212	153	52
212	153	44
212	153	100
62	182	169
35	8	142
35	8	128
128	8	142
128	8	128

Fault Tolerant Chain (FTC), the second practical CODCA proposed in [23], tries to find the collisions between the first sub-key and the other 15 sub-keys. The remaining

15 chains of FTC are shown in Table 5. Unlike TC, FTC has only the collision relationship between the first sub-key and the remaining 15 sub-keys. In other words, the remaining 15 sub-keys are still independent of each other. In this case, we still need to enumerate them using traditional key enumeration techniques under a certain candidate of k_1.

Group Collision Attack (GCA), another CODCA given in [18], continuously verifies long chains from short chains and greatly alleviates the growth of guessing space. It divides the 16 sub-keys of AES-128 into 8 big "groups", of which each contains 4 sub-keys. For intra-group collisions, GCA detects $k_1 \leftrightarrow k_3$ after detecting $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$. It then detects $k_2 \sim k_4$ and constructs the first group $k_1 \leftrightarrow k_4$, other groups are constructed in the same way. For inter-group collisions, the first two candidates of each group are in the former group and the last two are in the latter group. Take the 3 groups $k_1 \sim k_4$, $k_3 \sim k_6$ and $k_4 \sim k_8$ output by an experiment in Fig. 2 as an example, 17 $\leftrightarrow$ 242 and 192 $\leftrightarrow$ 24 in 17 $\leftrightarrow$ 242 $\leftrightarrow$ 192 $\leftrightarrow$ 24 are also in 212 $\leftrightarrow$ 153 $\leftrightarrow$ 17 $\leftrightarrow$ 242 and 192 $\leftrightarrow$ 24 $\leftrightarrow$ 229 $\leftrightarrow$ 126. In this way, we obtain a longer chain 212 $\leftrightarrow$ 153 $\leftrightarrow$ 17 $\leftrightarrow$ 242 $\leftrightarrow$ 192 $\leftrightarrow$ 24 $\leftrightarrow$ 229 $\leftrightarrow$ 126 including candidates of 8 sub-keys, although the sub-key k_4 is 9 and this chain is incorrect. Finally, a total of 32 pairs of collisions are exploited. The remaining candidates of GCA performed on the candidates of the improved TA and CECA given in Tables 1 and 2 are shown in Table 6, which are significantly fewer than TC and FTC.

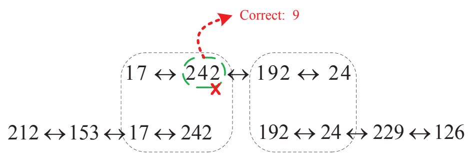

Figure 2: Key verification in GCA.

Table 6: Chains of GCA include $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$ and $k_2 \leftrightarrow k_3$.

k_1	k_2	k_3
212	236	17
212	236	105
212	236	44
212	236	100
212	236	222
212	236	128
212	**153**	**17**
212	153	105
212	153	44
212	153	100
212	153	222
212	153	128
62	182	52
62	182	142
167	150	17

k_1	k_2	k_3
212	236	17
212	236	44
212	236	100
212	**153**	**17**
212	153	44
212	153	100

3 Multiple-Differential Mechanism for CODCAs

Our novel Multiple-Differential mechanism for CODCAs (MD-CODCA) and its attack flow is provided in Algorithm 1. We encrypt plaintexts set $\mathbf{X}$, and acquire the corresponding power trace set $\mathbf{T}$. Then, we perform the improved CECA in Section 2.3 on the extracted POIs and acquire the candidates ranks $\phi = \{\phi_j | j = 1, \ldots, 120\}$ within threshold τ_d (Step 1). Here ϕ_j^i denotes the i-th best candidate of δ_j. We then perform the improved TA in Section 2.2 in Step 2 exploiting the well profiled templates $\left\{\left(\mathbf{m}_j^i, \mathbf{C}_j^i\right) \middle| 1 \leq i \leq 256; 1 \leq j \leq 16\right\}$, and acquire the ranked candidates $\xi = \left\{\xi_j \middle| j = 1, \ldots, 16\right\}$ for sub-keys under a success probability α (see Section 4). Here ξ_j^i denotes the i-th best candidate of the j-th sub-key k_j.

Algorithm 1: Multiple Differential Mechanism for CODCAs (MD-CODCA).

```
1 $\phi \leftarrow$ **ImprovedCECA** $\left(\mathbf{X}, \mathbf{T}, \left(\mathbf{m}_j^i, \mathbf{C}_j^i\right), \tau_d\right)$;
2 $\xi \leftarrow$ **ImprovedTA** $\left(\mathbf{X}, \mathbf{T}, \left(\mathbf{m}_j^i, \mathbf{C}_j^i\right), \alpha\right)$;
3 $\left(\xi'_{1,\ldots,n_s}, \xi'_{n_s+1,\ldots,16}\right) \leftarrow$ **SubkeySelection** $(\xi, n_s)$;
4 $\Omega \leftarrow$ **ChainConstruction** $\left(\xi'_{1,\ldots,n_s}, \phi, \tau_\sigma\right)$;
5 $\xi'_{n_s+1,\ldots,16} \leftarrow$ **RemainingSubkeyVoting** $\left(\Omega, \xi', \tau'_\sigma\right)$;
6 $\mathbf{B}_{1,\ldots,8} \leftarrow$ **BlockDivision** $\left(\xi', \phi\right)$;
7 $r \leftarrow$ **KeyEnumeration** $(\mathbf{B}_{1,\ldots,8}, k_{1,\ldots,16})$;
```

We then perform the first two differential mechanisms of our MD-CODCA. Specifically, we rank the number of candidates of 16 sub-keys in ξ in ascending order, and extract the first n_s sub-keys with the fewest candidates within threshold τ_k under a fixed probability α (see Step 3). Next, we perform collision detection on these n_s extracted sub-keys, and save the chains with the number of collisions more than the threshold τ_σ to Ω, thus achieving the first differential mechanism (Step 4). We further use these chains to vote for the candidates of the remaining $16 - n_s$ sub-keys exploiting a new threshold τ'_σ and complete the second differential mechanism (Step 5). More details will be introduced in Sections 5 and 6.

It is worth noting that the collision detection would be very time-consuming in the existing CODCAs if both τ_d and τ_k (provided by α) are very large. However, the first two differential mechanisms of our MD-CODCA are very flexible and very suitable for the situations where both τ_d and τ_k are very large. Therefore, we can build the third differential mechanism of our MD-CODCA by simply dividing 16 sub-keys into 8 non-overlapping blocks $k_1 \leftrightarrow k_2$, $k_3 \leftrightarrow k_4$, …, $k_{15} \leftrightarrow k_{16}$ like GCA, and computing the probability product of each pair of collision within each block (Step 6). Finally, key enumeration is performed on them to recover the key (Step 7).

4 Guessing Theory in TA

4.1 Guessing Model

We profile template for each intermediate value using 100 power traces, and randomly extract $160 \sim 360$ out of the remaining 25600 power traces in each of 200 repetitions to perform the improved TA introduced in Section 2.2. We then normalize and rank the probabilities of each sub-key in descending order to satisfy the guessing model in guessing theory, and obtain $p_j = \left\{p_j^1, p_j^2, \ldots, p_j^{|\mathcal{K}_j|}\right\}$ satisfying

$$\sum_{i=1}^{|\mathcal{K}_j|} p_j^i = 1 \tag{9}$$

and

$$p_j^1 > p_j^2 > \ldots > p_j^{|\mathcal{K}_j|} \tag{10}$$

as introduced in [8]. Here $|\mathcal{K}_j|$ denotes the number of candidates of k_j (e.g. 256 candidates for each sub-key of AES-128), and p_j^i denotes the i-th largest probability corresponding to its i-th best candidate ξ_j^i.

Considering the above sub-key k_j, guessing theory aims to evaluate the efficiency of an attacker trying to recover it given access to an oracle for queries "is $\xi_j^i = k_j$?" In this case, it needs to evaluate the probability that the sub-key $k_j = \xi_j^i$. The optimal brute-force attack proved by guessing theory in [8] is that, the attacker will achieve the minimum number of candidates that he needs to guess if he guesses the candidates of a sub-key according to their probabilities from the largest to the smallest. The average probability of successfully recovering the first sub-key under different number of guesses is shown in Fig. 3. With the increasing number of power traces n, the success probability of key recovery by guessing its first several best candidates also increases.

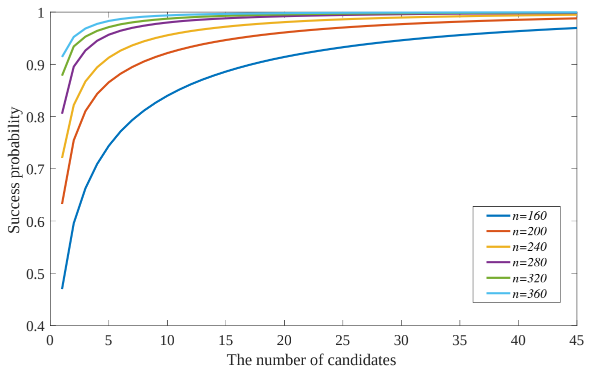

Figure 3: The average probability of successfully recovering the first sub-key under different number of guesses and different number of power traces.

4.2 Partial Guessing Metrics

Guessing entropy given in [22] provides the expected position of a sub-key. However, as stated in [4], entropy metrics failed to model the tendency of real-world attackers to avoid guessing the most difficult cases. For example, they may choose a small part of candidates with high probabilities to guess and discard most candidates with low probabilities in TA, such as only enumerating the first 2^{40} full-key candidates with the greatest probabilities from the total space 2^{128} of AES-128 in key enumeration. This often happens when the computing power is limited.

Suppose that an attacker tries to recover the sub-key k_j only from its first a few best candidates that satisfy the given probability value $\alpha \in (0, 1)$:

$$\mu_\alpha(\mathcal{K}_j) = \min\left\{ i' \middle| \sum_{i=1}^{i'} p_j^i \geq \alpha \right\}, \tag{11}$$

which is named as $\alpha\text{-}work\text{-}factor$ in [20]. We rank all the 16 lists of candidates of AES-128 in ascending order according to their $\alpha\text{-}work\text{-}factor$, and evaluate their corresponding average number of guesses under different number of power traces. Each evaluation is repeated 200 times. To be consistent with the experiments in Section 7, we set α to 0.996, and obtain the results shown in Fig. 4. The average number of candidates and probability distribution of the re-ranked sub-keys vary greatly in TA. Moreover, with the increasing number of power traces n, the average number of candidates of the 8 sub-keys with the largest μ_α-s decreases significantly. However, the average number of candidates of the 5 sub-keys with the smallest μ_α-s is less than 25 when $n \geq 200$.

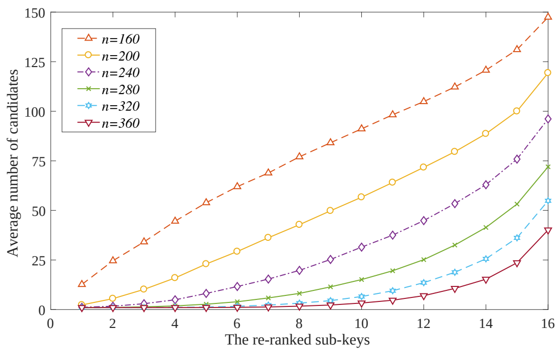

Figure 4: The average number of candidates of 16 re-ranked sub-keys.

4.3 Success Probability and Complexity

Previous works such as FTC and GCA, set a unified threshold for all sub-keys in TA, the smallest threshold should just be the position of the deepest sub-key. Obviously, most of the sub-keys are not ranked at such a depth, and thus a large number of additional undesirable candidates need to be considered. This significantly increases the cost of chain construction. For MD-CODCA, the number of candidates of different sub-keys under the same α-*work-factor* defined in Section 4.2 can be very different, each sub-key can also have an independent threshold. Success probability and the corresponding candidate space are two core factors to determine the thresholds. The larger probability also leads to huger candidate space and makes the key recovery more difficult. The existing CODCAs and our MD-CODCA can transform the original huge candidate space within thresholds into a much smaller chain space, thus reducing the complexity and guarantee the key recovery at a high probability.

The probabilities of 16 sub-keys of AES-128 in TA are independent. Therefore, the probability of the 16-byte key falling within the threshold $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{16})$ is:

$$\mathbf{p}_\alpha \geq \prod_{j=1}^{16} \alpha_j. \tag{12}$$

Standaert et al. defined partial success rate and global success rate in [22], the former is like the success probability of a sub-key of AES-128, the latter is like the success probability of the first-round key. However, the success probability here is different from success rate, since the former represents the theoretical expectation of success and the latter represents the success in practice. The size of candidate space under given $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{16})$ for AES-128 algorithm with 16 sub-keys is:

$$\Psi = \prod_{j=1}^{16} \mu_{\alpha_j}(\mathcal{K}_j). \tag{13}$$

It is noteworthy that TC, FTC, GCA and MD-CODCA vote and discard some candidates for each sub-key when they transform the original candidate space to a collision space. Therefore, they will reduce the probability of success. The probability of successful key recovery should be equal to the global success probability of the remaining candidates (i.e. satisfying Eq. 12).

5 Differential Mechanisms on Chains Construction for Voting

Our MD-CODCA extracts several sub-keys with the smallest μ_α-s provided by guessing theory introduced in Section 4. Then, fault tolerance on sub-keys in TA and additional fault tolerance on collision values δ-s in CECA are performed in the first differential mechanism (see Steps 3 $\sim$ 4 in Algorithm 1). The former (latter) is to allow a part of sub-keys (δ-s) beyond threshold τ_k (τ_d), thus guaranteeing the remaining sub-keys (δ-s) can be within threshold with a large probability. Here let n_s denote the number of sub-keys with the fewest candidates extracted as given in Step 3 in Algorithm 1. The collision chains among them can be established for voting the remaining $16 - n_s$ sub-keys.

5.1 Sub-keys Selection

CODCAs introduce collision information from CECA to transform a part of the best candidates in divide-and-conquer attacks to a collision space. They need to theoretically provide relaxed filtering conditions to reduce the possibility of mistakenly deleting sub-keys, and optimization is required in this case. Therefore, we rank the sub-keys according to their number of candidates μ_α in guessing theory optimization in our MD-CODCA, and extract the n_s sub-keys with the fewest candidates to vote for the remaining $16 - n_s$ sub-keys. Voting here means that only candidates of a sub-key satisfying the number of required collisions are considered. The collision chains among them are almost the fewest, and the number of candidates of the remaining $16 - n_s$ sub-keys they can exclude is also almost the largest.

5.2 Fault Tolerance on Sub-keys in TA

Since the probability distribution of each sub-key of AES-128 in TA is independent, the probability P_r that X of n_s sub-keys fall within the threshold μ_α follows the Bernoulli distribution $X \sim \mathcal{B}(n_s, \alpha)$:

$$P_r(X = \kappa) = \binom{n_s}{\kappa} \alpha^\kappa (1-\alpha)^{n_s - \kappa}. \tag{14}$$

Here $\kappa = 0, 1, 2, \ldots, n_s$ and

$$\binom{n_s}{\kappa} = \frac{n_s!}{\kappa!\,(n_s - \kappa)!}. \tag{15}$$

Actually, our fault tolerance on sub-keys in TA is to improve the success rate in theory. In this case, if we set n_s and α to 6 and 0.996, and a sub-key out of its threshold $\tau_k = \mu_\alpha$ is allowed, the success probability will reach $P_r(X = 6\ or\ 5) = 0.996^6 + 6 \cdot (0.996^5) \cdot 0.004^1 = 0.9998$, compared to $P_r(X = 6) = 0.996^6 = 0.9762$. If $\alpha = 0.90$, $P_r(X = 6\ or\ 5) = 0.8857$ and $P_r(X = 6) = 0.5314$, which shows significant improvements in success probability and illustrates the effectiveness of our fault tolerance mechanism.

5.3 Fault Tolerance on δ-s in CECA

Suppose that the 4 sub-keys k_1, k_2, k_3 and k_4 have the fewest candidates in an experiment. A simplified example of voting on the 5-th sub-key in TA is shown in Fig. 5, wherein the sub-key k_1 beyond τ_k is tolerated. Benefiting from this, MD-CODCA skips the consideration of collisions between it and other sub-keys including the one to be voted. In other words, δ-s between it and the other 4 sub-keys $k_2 \sim k_5$ in Fig. 5 are allowed to be beyond τ_d. It is noteworthy that not all δ-s beyond τ_d are related to the fault-tolerant sub-keys, and the fault tolerance on δ-s correlating to other sub-keys also needs to be considered.

For example, the collision value in $153 \leftrightarrow 26$ (i.e. $153 \oplus 26$) beyond τ_d is additionally tolerated in Fig. 5.

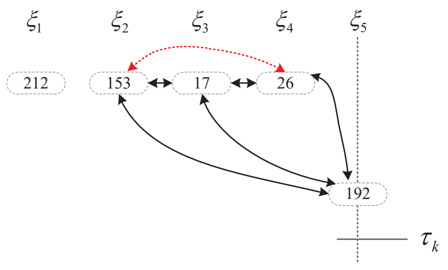

Figure 5: Sub-key k_1 and its related δ-s beyond τ_k and τ_d are tolerated, and the δ between k_2 and k_4 is additionally allowed to be beyond τ_d.

It's noteworthy that it is not flexible to perform fault tolerance on fixed δ-s. A good fault tolerance mechanism should allow δ-s between any two sub-keys to be beyond τ_d. Therefore, we perform cyclic fault tolerance on δ-s when detecting collisions. Specifically, let $\psi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right)$ denote the collision detection function of $\xi_{j_1}^{i_1}$ and $\xi_{j_2}^{i_2}$, if the collision establishes, i.e., $\xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \left\{\phi_{j_1,j_2}^{1}, \cdots, \phi_{j_1,j_2}^{\tau_d}\right\}$, this function returns 1. Otherwise, it returns 0, which means $\xi_{j_1}^{i_1}$ and $\xi_{j_2}^{i_2}$ is not a pair of collision. It satisfies:

$$\psi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right) = \begin{cases} 1, & \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \left\{\phi_{j_1,j_2}^{1}, \cdots, \phi_{j_1,j_2}^{\tau_d}\right\} \\ 0, & otherwise. \end{cases} \tag{16}$$

Here ϕ_{j_1,j_2}^{i} denotes the i-th best candidate of δ between sub-keys k_{j_1} and k_{j_2} as introduced in Step 1 of Algorithm 1. Not only all possible chains are used for voting but also their corresponding total number of collisions should be recorded. Suppose that $\xi_1^{i_1} \leftrightarrow \xi_2^{i_2} \leftrightarrow \ldots \leftrightarrow \xi_{n_s}^{i_{n_s}}$ is a chain, its total number of collisions is:

$$\eta\left(\xi_1^{i_1} \leftrightarrow \xi_2^{i_2} \leftrightarrow \ldots \leftrightarrow \xi_{n_s}^{i_{n_s}}\right) = \sum_{j_1=1}^{n_s-1} \sum_{j_2=j_1+1}^{n_s} \psi\left(\xi_{j_1}^{i_{j_1}}, \xi_{j_2}^{i_{j_2}}\right). \tag{17}$$

5.4 Thresholds Selection

To reduce the repetitive detection of collisions, we firstly detect chains from the n_s sub-keys with the fewest candidates within threshold $\tau_k = \mu_\alpha$, then directly use them for voting. There will be a total of $\binom{n_s}{2}$ δ-s among these selected sub-keys. Fault tolerance on sub-keys used for voting in TA is to guarantee that the remaining ones are still within τ_k with a high probability, collision values δ-s between them and the remaining sub-keys (including the one to be voted) should be also allowed, just as we explained in Section 5.3. If σ_1 sub-keys of them are beyond threshold τ_k, a total of $\binom{n_s}{2} - \binom{n_s - \sigma_1}{2}$ δ-s associated with them should be ignored, and the differential threshold should be set to:

$$\tau_{\sigma_1} = \binom{n_s - \sigma_1}{2}. \tag{18}$$

Taking $\sigma_1 = 1$ and $\sigma_1 = 2$ under $n_s = 6$ as an example, the threshold τ_{σ_1} is only 10 and 6 respectively. In other words, 10 (or 6) δ-s between other 5 (or 4) sub-keys are required when constructing a chain.

Compared to fault tolerance on sub-keys in TA, if we only consider fault tolerance on δ-s in CECA and σ_2 of them are allowed to be out of threshold τ_d, the differential threshold can be set to:

$$\tau_{\sigma_2} = \binom{n_s}{2} - \sigma_2. \tag{19}$$

It is worth noting that δ-s between the selected sub-keys except the σ_1 fault-tolerant ones, may still be out of τ_d as we have explained before. Therefore, an efficient differential mechanism should consider both these two cases. If σ_1 sub-keys in TA and additional σ_2 δ-s in CECA are allowed to be beyond thresholds when constructing chains for voting, the differential threshold is:

$$\tau_\sigma = \tau_{\sigma_1,\sigma_2} = \binom{n_s - \sigma_1}{2} - \sigma_2. \tag{20}$$

5.5 Chain Construction Algorithm

In principle, the performance of the collision detection algorithms wherein all collisions are not repetitively detected would be better. However, it is difficult to achieve this goal in fault tolerance mechanism. Since any sub-key or δ can exceed their thresholds and cyclic fault tolerance is needed to allow these situations to occur, thus guaranteeing a high probability of successful key recovery. We vote the other 10 sub-keys by using $n_s = 6$ sub-keys with the fewest candidates within the fixed μ_α, where one sub-key and an additional δ ($\sigma_1 = 1$, $\sigma_2 = 1$) are allowed to be beyond thresholds τ_k and τ_d in turn. We first find collisions between these sub-keys used for voting and obtain all chains with the number of collisions no less than the threshold τ_σ (as shown in Algorithm 2).

Algorithm 2: Chain construction performed on the sub-keys selected for voting.

```
1  for $j$ from 1 to $n_s$ do
2      $\xi'' \leftarrow$ FaultTolerance$\left(\xi', \xi'_j\right)$;
3      $\Omega'_j \leftarrow \xi''_1$; $\eta' \leftarrow 0$;
4      for $r$ from 1 to $n_s - 2$ do
5          for each chain $\xi''_1 \leftrightarrow \cdots \leftrightarrow \xi''_r$ in $\Omega'_j$ do
6              for each candidate within $\mu_\alpha$ of $\xi''_{r+1}$ do
7                  record $\eta'\left(\xi''_1 \leftrightarrow \cdots \leftrightarrow \xi''_r\right) + \sum_{1\le i\le r} \psi\left(\xi''_i, \xi''_{r+1}\right)$ to $\eta'_{j+1}$;
8                  record $\xi''_1 \leftrightarrow \cdots \leftrightarrow \xi''_r \leftrightarrow \xi''_{r+1}$ to $\Omega'_{j+1}$;
9              end
10         end
11         $\left(\Omega'_{j+1}, \eta'_{j+1}\right) \leftarrow$ ChainFilter$\left(\Omega'_{j+1}, \eta'_{j+1}, \tau_\sigma\right)$;
12         update $\Omega'_j$, $\eta'_j$ with $\Omega'_{j+1}$, $\eta'_{j+1}$;
13     end
14     record chains in $\Omega'_j$ to $\Omega$;
15     record the number of collisions in $\eta'_j$ to $\eta$;
16 end
```

Let k'_r (k''_r) denote the sub-key with the r-th fewest number of candidates within the fixed μ_α before (after) fault tolerance on sub-keys k'_j, and ξ'_r (ξ''_r) denote the corresponding candidates ($1 \le r \le n_s - 1$). We first remove the candidates of ξ'_j from ξ', and construct chains on the remaining $n_s - 1$ selected sub-keys (see Step 2 of Algorithm 1). We further use Ω'_j to record the chains without sub-key k'_j, and exploit η'_j to record their number of collisions (Step 3). Here $\eta_j^{'}$ satisfies the definition given in Eq. 17, and $\Omega_j^{'}$ is initialized by $\xi_1^{''}$. For each sub-key added subsequently, we detect the total number of collisions between its candidates and each chain in $\Omega_j^{'}$, and record the possible new chain to $\Omega_{j+1}^{'}$ (Steps 7 and 8). We then perform chain filtering after traversing all candidates of $\xi_{r+1}^{''}$ within threshold (Steps 11 and 12). Since $\sigma_2 = 1$ and only a δ beyond τ_d is allowed except for the δ-s associated with the fault-tolerant sub-key $k_j^{'}$ in Step 2, we initialize the differential threshold $\tau_\sigma = \binom{n_s - 1}{2} - 1 = 9$ according to Eq. 20. The filtering results are saved to $\Omega^{'}$ and $\eta^{'}$. They are finally saved to the chains set Ω and η used for voting after finishing fault tolerance on the sub-key $k_j^{'}$ (Steps 14 $\sim$ 15).

6 Efficient Voting and Key Recovery Mechanisms

6.1 Voting Mechanism on the Remaining Sub-keys

Based on the outputs Ω and η of Algorithm 2, we then exploit these chains to vote for the remaining sub-keys. The Voting mechanism exploited on the candidates of a remaining sub-key $k_j^{'}$ ($n_s + 1 \leq j \leq 16$) is shown in Algorithm 3. For each candidate to be voted, we simply traverse each chain of Ω, and detect the total number of collisions (Step 4). A candidate having collisions more than the differential threshold $\tau_\sigma^{'}$ will pass the candidate filter and be reserved. Otherwise, we discard it (Step 7). Here $\tau_\sigma^{'}$ is set to 14 compared to $\tau_\sigma = 9$ in Algorithm 2 of our experiments, since a sub-key and an additional δ have been tolerated and the remaining 14 δ-s will fall into the threshold τ_d with a very high probability. Actually, $\tau_\sigma^{'}$ is not a fixed threshold, and it can be set to a large value in the voting (e.g. 12 or 13) to achieve more flexible fault tolerance. It is also noteworthy that we can save a candidate once the number of collisions between it and the candidates on a chain is greater than $\tau_\sigma^{'}$, rather than traverse all chains. This will significantly improve the efficiency of voting.

Algorithm 3: Voting mechanism on the remaining sub-keys.

```
1 for $j = n_s + 1, \ldots, 16$ do
2     for each candidate within $\mu_\alpha$ of $\xi_j^{'}$ do
3         for each chain $\xi_1^{''} \leftrightarrow \cdots \leftrightarrow \xi_{n_s-1}^{''}$ in $\Omega$ do
4             $\eta_j^{'} \leftarrow \eta^{'}\left(\xi_1^{''} \leftrightarrow \cdots \leftrightarrow \xi_{n_s-1}^{''}\right) + \sum_{1 \leq i \leq n_s - 1} \psi\left(\xi_i^{''}, \xi_j^{'}\right)$;
5         end
6     end
7     $\xi_j^{'} \leftarrow$ CandidateFilter$\left(\xi_j^{'}, \eta_j^{'}, \tau_\sigma^{'}\right)$;
8 end
```

6.2 Key Enumeration in MD-CODCA

The key recovery will become much easier after CODCAs compared to traditional enumeration. However, CODCAs have their own limitations. For example, all chains are considered when splicing the long ones from the short ones, thus making the final key recovery very time-consuming. In this case, we can omit some collision information and use key enumeration to optimize them. If we keep the 4 big non-overlapping groups after GCA and ignore the collision information between them, 4 big independent lists of normalized probabilities can be obtained for key enumeration. The number of candidates to enumerate will be very small since the collision information has been used once, and these 4 groups are already the remaining chains.

The number of candidates of the first $n_s = 6$ sub-keys in Fig. 4 will not change significantly in a wide range of number of measurements. Therefore, The number of possible chains in Ω is usually very small. Moreover, the flexible voting mechanism introduced in Section 6.1 makes it easy for our MD-CODCA to vote for the other $16-n_s$ sub-keys, which may be deeply ranked. The above unique differential mechanisms make our MD-CODCA very suitable for the situations where both τ_k and τ_d are very large, thus guaranteeing that the sub-keys and δ-s are within their thresholds with a very large probability.

Benefiting from the above mentioned advantages, we can build another very simple differential mechanism in our MD-CODCA, thus facilitating our later key enumeration. Specifically, we simply divide the 16 lists of remaining candidates into 8 big "blocks": $k_1 \leftrightarrow k_2$, $k_3 \leftrightarrow k_4$, $\ldots$, $k_{15} \leftrightarrow k_{16}$, and detect the collisions within them, just like the 4 big non-overlapping "blocks" in GCA: $k_1 \leftrightarrow \cdots \leftrightarrow k_4$, $k_5 \leftrightarrow \cdots \leftrightarrow k_8$, $\ldots$, $k_{13} \leftrightarrow \cdots \leftrightarrow k_{16}$. To make them independent, other collision information is ignored in this stage. We then compute the probability product of each pair of collision, and normalize their probabilities. Key enumeration is then performed on the 8 lists of probability products of collisions output by this third differential voting mechanism to recover the key as introduced in Section 3.

7 Experiments Results

The main purpose of MD-CODCA is to introduce collision information from CECA into divide-and-conquer attacks, and transform a part of the best candidates to a much smaller collision space, thus facilitating the latter key recovery. The performance of the improved CECA and TA will be given in Section 7.1. Important parameters except for $\tau_k = \mu_\alpha$, such as threshold τ_d for CECA and the number of power traces n, are involved in our MD-CODCA. We will discuss them separately in Sections 7.2 and 7.3. Power traces are randomly extracted to launch attacks, and each experiment is repeated 200 times (100 times for TC). Histogram based key rank estimation in [21] is directly performed on the original space and the remaining space of FTC, the 4 big "blocks" in GCA and the 8 big "blocks" in MD-CODCA as introduced in Section 3. α is set to 0.996 and the corresponding global success probability is $0.996^{16} = 0.9372$.

7.1 Performance of Improved CECA and TA

To satisfy the guessing theory model, this paper uses the improved TA given in [25], which achieves performance significantly better than the traditional CECA. We need to set τ_d much larger than τ_k in this case. Otherwise, the collision information is difficult to be exploited in the existing CODCAs. To facilitate the performance comparison, we simply improved CECA in Section 2.3. Although their performance is still quite different as the guessing entropy [22] shown in Fig. 6, CECA's performance is significantly improved.

It's noteworthy that the two improved attacks given in Sections 2.2 and 2.3 exploit templates information, which indicates that both of them are profiled attacks. However, unlike the existing schemes TC, FTC and GCA, MD-CODCA doesn't need to strictly balance their performance. This is because our MD-CODCA is a relatively flexible differential mechanism, which allows for obvious performance differences between the combined divide-and-conquer attacks and collision attacks. Experiments in Sections 7.2 and 7.3 will show that it can still work well.

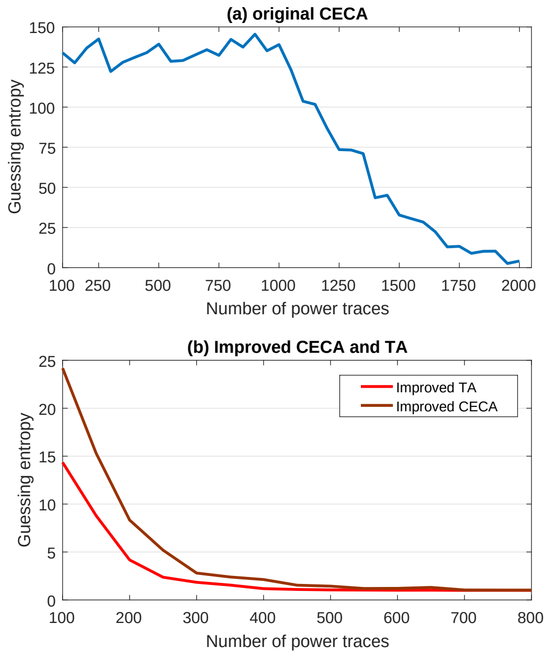

Figure 6: Guessing entropy of the improved TA and CECA under different number of power traces.

7.2 The Influence of τ_d

We set the number of power traces used in each repetition to 240, and investigate how τ_d affects the performance of our MD-CODCA. We only compare the performance of TC, FTC, GCA and our MD-CODCA in this section, and leave the introduction of the original candidate space under 240 power traces in Section 7.3 (see Fig. 9). The success rates under different τ_d-s are shown in Fig. 7. Computing power in this figure means the enumeration power. For example, abscissa 30.0 and ordinate 0.7 indicate that we can achieve a success rate of 0.7 by enumerating at most 2^{30} candidates. With the increase of τ_d, more candidates satisfying the collision conditions of FTC, GCA and MD-CODCA are maintained, and the remaining candidate space gradually increases. Significant changes also occur in the rank of the key.

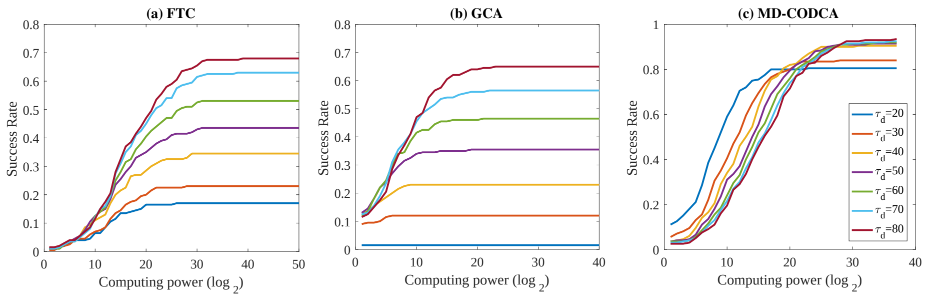

Figure 7: Success rates under different thresholds τ_d.

The smaller the τ_d, the better the rank of the key, and the lower the success rate (as shown in Fig. 7). This indicates that fewer candidates (including the correct and wrong ones) satisfy the collision conditions. This also tells us that smaller τ_d requires more flexible collision conditions and better fault tolerance mechanisms. The success rates of FTC and GCA increase rapidly when τ_d is from 20 to 80, but are still relatively low. They only reach 0.68 and 0.65 at $\tau_d = 80$. MD-CODCA achieves success rate of about 0.80 at $\tau_d = 20$, which fully illustrates its high efficiency. The success rates of MD-CODCA are very close when τ_d reaches 40, so $\tau_d = 40$ is a good threshold when $n = 240$. The success rates of our MD-CODCA are very close to the theoretical probability 0.9372 and significantly higher than FTC and GCA, which fully illustrates its superiority.

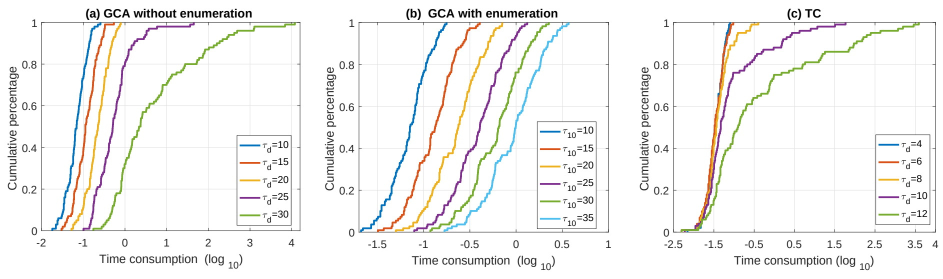

Figure 8: Time consumption (seconds) of TC and GCA (with or without key enumeration).

Table 7: Time consumption (seconds) under different thresholds τ_d.

τ_d	20	30	40	50
FTC	0.037	0.038	0.046	0.058
GCA with enumeration	0.21	0.57	1.28	2.60
MD-CODCA	0.77	1.25	1.36	1.57
τ_d	60	70	80	–
FTC	0.067	0.093	0.091	–
GCA with enumeration	5.41	13.21	37.50	–
MD-CODCA	1.83	1.92	2.08	–

In terms of algorithm runtime, GCA is significantly affected by the threshold τ_d. The time consumption of GCA with or without key enumeration is given in Figs. 8(a) and 8(b). Here GCA without key enumeration means it has to splice long chains from short chains as described in Section VI-B. However, FTC and MD-CODCA do not change much under different thresholds τ_d (see Table 7). In fact, τ_d also has a significant impact on MD-CODCA. However, we can mitigate its impact by selecting 6 sub-keys having the fewest candidates within τ_k in the first differential voting mechanism of MD-CODCA, which makes the number of chains used for voting smaller when $n = 240$ (see Fig. 3), and improves the efficiency of voting. Moreover, Algorithm 3 flexibly exits the rotating fault tolerance on sub-keys when there exists a chain making a candidate satisfy the fault-tolerant conditions, which notably reduces collision detection time.

The chain construction in FTC is quick, but the other 15 sub-keys are still independent of each other under a guessing k_1 and traditional key enumeration is required. TC and GCA construct long chains from short ones. GCA exploits a lot of collision information, and it leaves very small space for key recovery. However, TC exploits very limited collision information, which may produce a large number of chains under very large τ_d and τ_k as we consider in this paper. In fact, a small increase in τ_d and τ_k will bring much huger candidate space. For example, the candidate space $4^{16} = 2^{32}$ becomes $8^{16} = 2^{48}$

if τ_k is improved from 4 to 8. The time consumption increases dramatically when we keep τ_k constant and adjust τ_d from 4 to 12, and randomly extract 240 power traces for experiments (see Fig. 8(c)). About 10% (5%) of experiments consume more than 100 (1000) seconds when $\tau_d = 12$. TC ran for two days under $\tau_d = 20$, but only finished 3 repetitions. Therefore, it is difficult for TC to deal with very large thresholds.

7.3 The Influence of the Number of Traces

The number of power traces plays an important role in attacks, more power traces will provide more leakage information and reduce the number of guesses of sub-keys under a fixed probability α (i.e. μ_α), thus reducing the candidate space. Although $\tau_d = 40$ is a good threshold when the number of power traces $n = 240$ (Fig. 7), fewer power traces need larger threshold. Therefore, we set τ_d to 60, and the experimental results when the number of traces n ranges from 160 to 320 are shown in Fig. 9. We also show the corresponding guessing spaces within τ_k since they are dramatically affected by n. For example, the probability of guessing space within $\tau_k = \mu_\alpha$ larger than 70 bits is almost 1.00 when $n \leq 180$, and only 50% when $n = 240$, and decreases to about 0.00 when n reaches 280. The estimated ranks of keys from the improved TA in Fig. 9 are also much better.

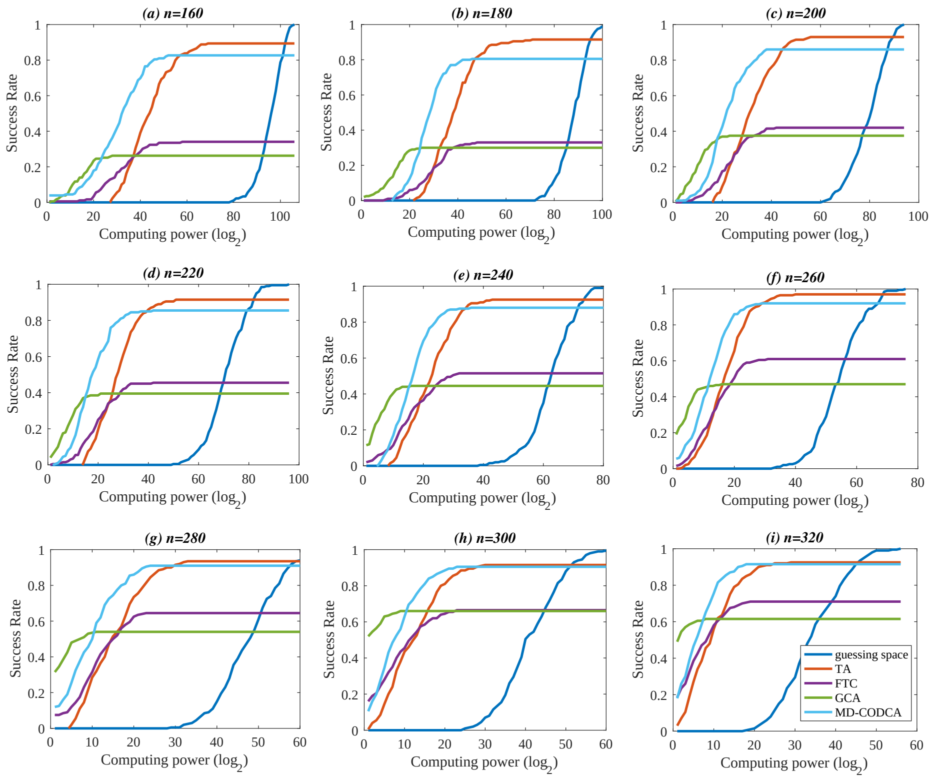

Figure 9: Success rates under different number of traces.

The keys in FTC rank deeper than those in GCA (with enumeration) and MD-CODCA. The success rates of FTC and GCA with enumeration are similar and relatively low. They only reach 0.70 and 0.61 respectively when $n = 320$, and are much lower than 0.82 of MD-CODCA under $n = 160$. This fully demonstrates that our MD-CODCA significantly improves the success rate without notably increasing the considered candidate space and the difficulty of key recovery. More power traces exploited in MD-CODCA will make its success rate closer to the theoretical success probability. The difference between them is about 0.07 at $n = 160$ and decreases to 0.03 when $n = 320$. These results demonstrate the superiority of MD-CODCA over FTC and GCA.

Table 8: Time consumption (seconds) under different number of traces.

n	160	180	200	220	240
FTC	0.66	0.39	0.21	0.29	0.068
GCA with enumeration	707.84	213.92	75.27	17.91	5.95
MD-CODCA	651.77	106.53	17.74	6.083	1.920
n	260	280	300	320	–
FTC	0.030	0.019	0.012	0.007	–
GCA with enumeration	2.89	1.35	0.63	0.35	–
MD-CODCA	0.564	0.146	0.061	0.026	–

The runtime of GCA with enumeration is still most severely affected by the number of power traces (as shown in Table 8). The number of candidates of the selected 6 sub-keys are quite large when n is too small (e.g. about 25 for the second and about 60 for the 6-th re-ranked sub-keys in average as shown in Fig. 3). This also makes it time-consuming for MD-CODCA to perform fault tolerance on sub-keys and collision values δ-s outside the thresholds τ_k and τ_d. It is worth noting that the parameter α can be flexibly adjusted according to our computing power. For example, we can set $\alpha = 0.9$ when $n = 160$, and $\alpha = 0.998$ when $n = 320$. MD-CODCA can easily obtain the success probability of key recovery of TA and determine the candidate space it needs to deal with (see Section 4.3). It also facilitates the choice of a reasonable threshold τ_d.

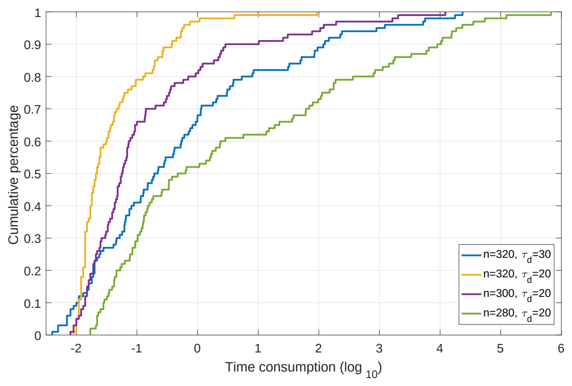

Figure 10: Time consumption (seconds) of TC under different number of power traces.

The time consumed by TC under different number of power traces when $\tau_d = 20$ is shown in Fig. 10. As more power traces are exploited, the number of candidates of each sub-key under $\alpha = 0.996$ in the improved TA is rapidly reduced. Although we keep the threshold τ_d of the improved CECA unchanged, both the number of collision-pairs to be detected and the time consumption are significantly reduced. However, TC is still infeasible under huge threshold $\tau_d = 60$ when n reaches 320. When $\tau_d = 30$, the corresponding success rate is 0.47. About 10% (5%) of experiments take more than 100 (10^3) seconds, and a few experiments even take more than 10^4 seconds. Therefore, the evaluation is very time-consuming. Only one experimental result was obtained in three days when we adjusted τ_d from 30 to 40. It's even worse when $n = 280$, about 10% of the experiments take more than 10^4 seconds. Obviously, TC takes more time than GCA with enumeration, while our MD-CODCA takes the least amount of time. Although FTC is fast, it leaves us a very huge candidate space. Moreover, MD-CODCA's flexible fault tolerance mechanisms enable it to achieve significantly higher success rate while notably reducing the difficulty of key recovery of the existing CODCAs. All these demonstrate the superiority of MD-CODCA.

8 Conclusion

This work introduces guessing theory into TA to optimize the number of guesses of each sub-key, and provide the attacker with quick estimation of the difficulty and success probability of key recovery. Moreover, to better transform a part of the best candidates in divide-and-conquer attacks into smaller collision space, we propose MD-CODCA to exploit collision information from CECA to optimize TA. Our MD-CODCA includes three differential mechanisms, of which the first two exploit several sub-keys with the fewest candidates within threshold to construct chains to vote for the candidates of the remaining sub-keys. Benefiting from its flexibility, MD-CODCA is very suitable for large thresholds. In this case, most of collision values are within the threshold, and MD-CODCA can build another differential mechanism to further divide the 16 sub-keys into 8 big non-overlapping "blocks" and perform collision detection on the remaining candidates within each of them. The collision information between blocks, which may have already been utilized in the first two differential mechanisms, is properly ignored, thus facilitating the later key enumeration. A good CODCA should significantly reduce the original candidate space of divide-and-conquer attacks, without significantly lowering the success rate. Our MD-CODCA well achieves this goal, and has made significant improvements compared to the existing CODCAs.

With the introduction of collision information into divide-and-conquer attacks, CODCAs make it possible to recover the key ranked at very deep space, thus bringing new opportunities for key recovery. Several works in [6,12,13] have provided the upper bound of security level of their improved CECA to show the performance. However, CODCAs require the lower bound rather than the upper bound of CECA to guarantee the remaining collision values except for the fault-tolerant ones, can be within threshold with a very large probability (e.g. close to 1.00). This will facilitate the fault tolerance on collision values and its corresponding threshold setting, ad makes the CODCAs more flexible. Our future work will focus on solving this issue. In addition, we also plan to optimize the collision detection algorithms discussed in Sections 5 and 6.1 to make them feasible in much larger guessing space.

References

[1] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi. The EM Side-Channel(s). In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 29–45, 2002.

[2] A. Bogdanov. Multiple-Differential Side-Channel Collision Attacks on AES. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 30–44, 2008.

[3] A. Bogdanov and I. Kizhvatov. Beyond the Limits of DPA: Combined Side-Channel Collision Attacks. *IEEE Trans. Computers*, 61(8):1153–1164, 2012.

[4] J. Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 538–552, 2012.

[5] E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29, 2004.

[6] N. Bruneau, C. Carlet, S. Guilley, A. Heuser, E. Prouff, and O. Rioul. Stochastic Collision Attack. *IEEE Trans. Information Forensics and Security*, 12(9):2090–2104, 2017.

[7] S. Chari, J. R. Rao, and P. 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.

[8] M. O. Choudary and P. G. Popescu. Back to Massey: Impressively Fast, Scalable and Tight Security Evaluation Tools. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 367–386, 2017.

[9] F. Durvaux and F. Standaert. From Improved Leakage Detection to the Detection of Points of Interests in Leakage Traces. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 240–262, 2016.

[10] C. H. Gebotys, S. Ho, and C. C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, pages 250–264, 2005.

[11] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer. Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 207–228, 2015.

[12] B. Gérard and F. Standaert. Unified and Optimized Linear Collision Attacks and Their Application in a Non-profiled Setting. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 175–192. Springer, 2012.

[13] C. Glowacz and V. Grosso. Optimal Collision Side-Channel Attacks. In S. Belaïd and T. Güneysu, editors, *Smart Card Research and Advanced Applications - 18th International Conference, CARDIS 2019, Prague, Czech Republic, November 11-13, 2019, Revised Selected Papers*, volume 11833 of *Lecture Notes in Computer Science*, pages 126–140. Springer, 2019.

[14] M. S. Inci, B. Gülmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar. Cache Attacks Enable Bulk Key Recovery on the Cloud. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 368–388, 2016.

[15] P. C. Kocher, J. Jaffe, and B. 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.

[16] Y. Li, S. Wang, Z. Wang, and J. Wang. A strict key enumeration algorithm for dependent score lists of side-channel attacks. In T. Eisenbarth and Y. Teglia, editors, *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, volume 10728 of *Lecture Notes in Computer Science*, pages 51–69. Springer, 2017.

[17] A. Moradi, O. Mischke, and T. Eisenbarth. Correlation-Enhanced Power Analysis Collision Attack. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 125–139, 2010.

[18] C. Ou, Z. Wang, D. Sun, and X. Zhou. Group Collision Attack. *IEEE Trans. Information Forensics and Security*, 14(4):939–953, 2019.

[19] C. Ou, Z. Wang, D. Sun, X. Zhou, and J. Ai. Group Verification Based Multiple-Differential Collision Attack. In *Information and Communications Security - 18th International Conference, ICICS 2016, Singapore, November 29 - December 2, 2016, Proceedings*, pages 145–156, 2016.

[20] J. O. Pliam. On the Incomparability of Entropy and Marginal Guesswork in Brute-Force Attacks. In *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, pages 67–79, 2000.

[21] R. Poussier, F. Standaert, and V. 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.

[22] F. Standaert, T. Malkin, and M. Yung. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[23] D. Wang, A. Wang, and X. Zheng. Fault-Tolerant Linear Collision Attack: A Combination with Correlation Power Analysis. In *Information Security Practice and Experience - 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*, pages 232–246, 2014.

[24] Y. Yarom, D. Genkin, and N. Heninger. CacheBleed: A Timing Attack on OpenSSL Constant Time RSA. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 346–367, 2016.

[25] H. Zhang. How to Effectively Decrease the Resource Requirement in Template Attack? In *Advances in Information and Computer Security - 9th International Workshop on Security, IWSEC 2014, Hirosaki, Japan, August 27-29, 2014. Proceedings*, pages 119–133, 2014.