Combined Full Collision Attack: Pushing the Limits of Exhaustible Key Spaces

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. Recovering keys efficiently from very deep candidate space is a very important but challenging issue in Side-Channel Attacks (SCA). State-of-the-art combined collision attacks extract specific collisions from the outputs of a divide-and-conquer attack and an analytical attack, thus reducing the large guessing spaces to much smaller collision spaces. However, the inefficient chain detection makes them time-consuming. The very limited collisions exploited and very different performance of two combined attacks also prevent their application in much deeper spaces. In this paper, we propose a Minkowski Distance enhanced Collision Attack (MDCA) with performance close to Template Attack (TA), thus making their combination more practical and meaningful. Moreover, we build a more advanced combined collision attack named Combined Full Collision Attack (CFCA) from TA and MDCA to fully exploit collisions. We further incorporate guessing theory into CFCA to enable the determination of suitable thresholds and optimize search orders of sub-keys. Finally, to set the thresholds as small as possible while guaranteeing a high success probability of key recovery, we propose Block based Fault-Tolerant CFCA (BFT-CFCA). We further exploit the Fault-Tolerant Vector (FTV) to provide a reference for its chain space adjustment. Experimental results show that BFT-CFCA notably outperforms the existing methods and CFCA.

Keywords: CFCA · BFT-CFCA · MDCA · collision attack · divide and conquer · key enumeration · side-channel attack

1 Introduction

Implementations of cryptographic algorithms on devices produce unintentional leakages from various channels such as time [Koc96, YGH16], power consumption [KJJ99, WYS+18], electromagnetic [GPPT16, EFGT17], cache patterns [MIE17, DKPT17], acoustic [GST14]. They can be statistically analyzed for key recovery, which poses serious threats to the security of cryptographic devices. Power consumption is one of the most widely used channels in SCA, on which side-channel attacks can be classified as divide-and-conquer and analytical. Divide-and-conquer attacks, such as Correlation Power Analysis (CPA) [BCO04], Template Attack (TA) [CRR02] and Mutual Information Analysis (MIA) [GBTP08], divide the huge key candidate space into small sub-keys and conquer them independently. Then, the attacker recombines the information e.g. via key enumerations [PSG16, LWWW17, VGS13]. However, key enumeration is limited by the computing power of the attacker, and can only be performed on implementations that are "practically insecure" (for which the leakage allows for key enumeration). Key rank estimation tools such as histogram [GGP+15], can bound the security level of AES-128 with an error of less than 1 bit within 1 second. However, they require the knowledge

of the key, and hence can only be used for evaluation. Analytical attacks such as collision attacks [MME10, SLFP04, LMV04], aim at recovering the full key together e.g. by solving systems of equations. Divide-and-conquer approaches have the benefits of fast implementation and low knowledge requirements, while analytical strategies exploit more leaky information.

The concept of combined collision attack was given in [BK12], which exploited key related information from the outputs of both collision attacks (e.g. Correlation enhanced Collision Attack (CCA) [MME10]) and divide-and-conquer attacks (e.g. TA), aiming to recover the full key from them under given thresholds. In other words, it only guesses a part of candidate space and tries to recover the key from it. A concept named Test of Chain(TC) was given in [BK12], and another two practical works named Fault-Tolerant Chain(FTC) and Group Collision Attack (GCA) in [WWZ14, OWSZ19] followed it, of which the ability of key recovery was fully demonstrated. Unlike the enhanced single collision distinguishers such as the optimized linear collision attacks in [GS13], improved CCA in [WK18] and Stochastic Collision Attack in [BCG+17], they eliminate bad candidates that locate within thresholds but do not satisfy the given collision conditions to build collision chains (i.e. good candidates), so that key recovery can be feasibly applied on the much smaller remaining chain space. Combined collision attacks are independent of specific distinguishers used. Therefore, unlike classical collision attacks, they can be regarded as post-processing key recovery schemes. However, the above-mentioned schemes have difficulties in dealing with larger candidate space. They will be introduced in the next subsection before introducing our contributions.

1.1 Related Works

Bogdanov et al. proposed the first combined collision attack named Test of Chain (TC) in [BK12]. Taking the combination of TA and CCA as an example, TC introduces the XOR-values provided by CCA to the ranks output by TA to construct specific collisions to avoid exhaustion. Let k_j denote the j^{th} sub-key and τ_k denote the threshold of each sub-key, which means only the τ_k most likely key guesses will be considered. Like τ_k in TA, TC also sets a reasonable threshold τ_d for the outputs of CCA. A collision happens if two sub-keys k_{j_1} and k_{j_2} and their collision value $k_{j_1} \oplus k_{j_2}$ are within τ_k and τ_d simultaneously, and a chain includes at least a pair of collision and has a single free variable. Rather than randomly exhaust all candidates of the free variable if only CCA is used, TC only considers τ_k candidates ranked in the front of their lists, and can recover key from them with high probability.

TC [BK12] tries to find a long chain from the first sub-key to the last sub-key. The key recovery fails if at least one collision value exceeds the threshold τ_d. This requires the attacker to either capture more leakages or set a larger threshold. Moreover, no practical scheme on how to efficiently build the chain was given in [BK12]. Wang et al. gave the first feasible scheme named Fault Tolerant Chain (FTC) combining CPA with CCA in [WWZ14], which can find collisions between the first sub-key and the remaining sub-keys. If multiple sub-keys do not collide with the first sub-key k_1, the attacker exhausts them independently in FTC. In this case, FTC will quickly fail since too large remaining candidate space to against. Another important issue is that CCA's performance in both FTC and GCA in [WWZ14, OWSZ19] is worse than CPA's. This is not surprising since the original intention of CCA is to attack flawed masks (e.g. DPA contest $v4.1$ [dpa]). If the performance of two distinguishers is notably different, the key can be recovered efficiently using the one with better performance, since collision information is difficult to exploit in such a case. The significance of their combination lies in obtaining better key recovery capability than their attack separately.

Group Verification Chain (GVC) [OWS+16] uses the collisions between the current sub-key and other sub-keys to verify and rank its candidates within the threshold. In this

case, a sub-key is verified by other sub-keys. The more collisions, the higher probability that the candidates are the correct sub-keys. However, GVC neither changes the output scores or probabilities of candidates, nor deletes a part of them, so it is not conducive to key recovery. This is contrary to the intention of this paper, thus we do not consider this strategy. Group Collision Attack (GCA) divides the full-key space into several big "groups" containing the same number of sub-keys and makes full use of collision information to remove bad candidates within each group. It continuously splices long chains from short chains. Specifically, it makes the two adjacent groups share several sub-keys. If the values of them are the same, a longer chain can be stitched. GCA can utilize collision information several times more than TC and FTC, and thus has much stronger search ability. However, GCA still insufficiently exploits the collision information across different groups (see Section 3.2 for details). Moreover, GCA builds larger groups through constant splicing, which avoids exponential growth in the number of recombined full-key candidates, but requires a lot of computation and significantly increases its runtime.

It is worth mentioning that the purpose of this paper is not to improve the key enumeration or rank estimation. In fact, any improvement of side-channel attacks will be reflected in the more advanced ranking of the key. As we have mentioned above, combined collision attacks eliminate the consideration of bad candidates within thresholds that do not satisfy the given collision conditions, so that the later key recovery(e.g. key enumeration) can be feasibly applied on the remaining chain space. Taking AES-128 as an example, if a pair of candidates of the first and second sub-keys are deleted due to the absence of collisions, then all τ_k^{14} possible full-key candidates consisting of this guessing pair and the subsequent 14 sub-keys within threshold τ_k will be deleted. An attacker can recover the key from the much smaller remaining chain space. The challenge of combined collision attacks lies in the ability to construct the chains efficiently, leaving the attacker with a chain space that is as small as possible to reduce the difficulty of key recovery, while not significantly lowering the probability of key recovery. Therefore, it is obviously an art to construct such a combined collision attack.

1.2 Our Contributions

The main contributions of this paper are as follows:

(i) Combined collision attacks are independent of specific distinguishers used. However, both FTC and GCA combined CCA with significantly higher performance CPA in [BK12, OWSZ19], where τ_d need to be set very large. Otherwise, the collision information is difficult to be exploited. We introduce a Minkowski Distance enhanced Collision Attack (MDCA) with performance close to TA, thus making their combination more practical, meaningful and challenging.

(ii) FTC and GCA exploit only a small part of collisions and waste most of them. They leave us a large number of full-key candidates and are time-consuming when against large spaces. We propose a simple and efficient combined collision attack named Combined Full Collision Attack (CFCA) to efficiently exploit collision information between any two sub-keys, thus having much stronger search capability than FTC and GCA. We further introduce the guessing theory to determine the guessing lengths and optimize search order of sub-keys, thus significantly alleviating the issue where the collision detection load of the existing combined collision attacks is mainly concentrated on the first several sub-keys under given thresholds.

(iii) CFCA requires all collision values output by collision attack to be within the threshold, so that τ_d usually needs to be set very large. We find a very important phenomenon that collision values beyond τ_d can be optimized by performing fault tolerance on only a few sub-keys. Based on this, we propose Block Fault-Tolerant CFCA

(BFT-CFCA), which can significantly reduce τ_d of CFCA. We further exploit the Fault-Tolerant Vector (FTV) to provide a reference for its chain space adjustment. These significantly improve CFCA's performance.

1.3 Organization

The rest of this paper is organized as follows: Measurement setups, TA and Collision Attack (CA) including CCA are introduced in Section 2. The combined collision attacks TC, FTC and GCA are presented in Section 3. Our MDCA, CFCA and its optimization based on guessing theory are detailed in Section 4. The collision distribution and BFT-CFCA are discussed in Section 5. Experiments on an AT89S52 micro-controller are presented in Section 6. Finally, we conclude this paper in Section 7.

2 Preliminaries

2.1 Measurement Setup

Our experiments are performed on the power traces leaked from an AT89S52 micro-controller, which facilitates linear collision attacks[1]. Its operating frequency is 12 MHz, and the shortest instructions take 12 clock cycles. We exploit assembly language to implement the AES-128 algorithm, and use the instruction "MOVC A, @A+DPTR" to perform the S-box operation, which requires 24 clock cycles. The register "DPTR" saves the starting address of the S-box, register "A" saves the offset, and the output of the lookup table is stored back to register "A". Sampling rate of the Picoscope 3000 is set to 125 MS/s. We acquired 50000 power traces and performed CPA to extract the time sample with the highest correlation coefficient for each S-box. We then undertook our experiments using MATLAB *R*2016*b* on a HP desktop computer with 6 Intel(R) Xeon(R) E5-1650 v2 CPUs, 16 GB RAM and a Windows 10 operating system.

2.2 Template Attack

The classical Template Attack(TA) includes two stages. Taking AES-128 on our AT89S52 micro-controller as an example, we encrypt n_j plaintexts $\mathbb{X}_j = \left(x_1, x_2, \ldots, x_{n_j}\right)$ having the same Hamming weight of their S-box outputs, acquire n_j power traces $\mathbb{T}_j = \left(\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{n_j}\right)$. We then exploit their POIs $\mathbb{T}_j^{'} = \left(t_1, t_2, \ldots, t_{n_j}\right)$ to profile the mean:

$$\mathbf{m}_j = \frac{1}{n_j} \sum_{\kappa=1}^{n_j} t_\kappa \tag{1}$$

and covariance matrix:

$$\mathbf{C}_j = \frac{1}{n_j} \sum_{\kappa=1}^{n_j} \left(t_\kappa - \mathbf{m}_j\right) \left(t_\kappa - \mathbf{m}_j\right)^{\mathsf{T}}, \tag{2}$$

for templates $(\mathbf{m}_j, \mathbf{C}_j)$ $(1 \le j \le 9)$. 9 templates with Hamming weights from 0 to 8 are constructed in the profiling stage. Here the symbol "T" denotes matrix transposition. In the attack stage, the probability of a new measurement with POIs t generated by encrypting x having the same Hamming weight with template $(\mathbf{m}_j, \mathbf{C}_j)$ satisfies:

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

Here $|\mathbf{m}_j|$ represents the number of POIs on template $\mathbf{m}_j$.

[1] Linear collision attack is infeasible in parallel implementation as explained in [GS13].

2.3 Collision Attacks

AES-128 performs the "SubBytes" operation (16 parallel S-box applications) in its first round. Let k_j denote the j-th sub-key, and x_j denote the corresponding encrypted plaintext byte. A generalized internal AES-128 linear collision [BK12] occurs if there are two S-boxes in the same AES encryption or several encryptions with the same byte value as their input (as shown in Fig. 1). The attacker finds a collision:

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

and obtains a linear equation:

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

Since the 16 S-boxes of AES-128 are exactly the same in each round, the following collision value is obtained:

$$\delta_{j_1,j_2} = k_{j_1} \oplus k_{j_2} = x_{j_1} \oplus x_{j_2}, \tag{6}$$

which means that although these two sub-keys are unknown, they have a fixed XOR value. In this case, the classical collision detection function can be defined as:

$$\phi\left(x_{j_1} \oplus k_{j_1}, x_{j_1} \oplus k_{j_2}\right) = \begin{cases} 1, & if \;\; \mathcal{D}\left(t_{j_1}, t_{j_2}\right) \leq \tau_{po} \\ 0, & else \end{cases}. \tag{7}$$

Here $\mathcal{D}\left(t_{j_1}, t_{j_2}\right)$ is the matching function of two traces (e.g. Euclidean distance) and τ_{po} is the corresponding threshold.

Suppose that we have detected a total number of ν collision values, then a collision system is obtained:

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

of which each pair of collision is a step of chains. We may not find all collision values in an attack, several chains rather than a long chain including all sub-keys are obtained in this case. Each chain includes one free variable to exhaust. In this case, collision attacks establish relationships among multiple sub-keys. The complexity of key recovery depends on the number of chains (i.e., the number of free variables).

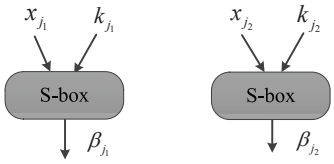

Figure 1: A linear collision between the j_1-th and the j_2-th S-boxes happens if $\beta_{j_1} = \beta_{j_2}$.

Correlation enhanced Collision Attack (CCA) [MME10] can be used to detect δ-s. A specific implementation of typical CCA was given in Algorithm 2 in [WWZ14]. Taking the collision between the first and second S-boxes in the first round of AES-128 as an example, CCA divides their traces into 256 classes according to their plaintext bytes, calculates the

mean power consumption vector $\mathbf{m}_j (j = 1, 2)$ of each class, and computes the correlation coefficient between them:

$$\rho\left\{\left(\mathbf{m}_1^{\alpha}, \mathbf{m}_2^{\alpha \oplus \delta_{1,2}}\right) | \alpha = 0, 1, 2, \ldots, 255\right\}$$

under a guessing $\delta_{1,2}$ (see Eq. 6). Here α from 0 to 255 is the index of 256 means in each class.

3 Combined Collision Attacks

As we have explained in Section 2.3, CCA provides us the collision values δ-s to build a collision system including several chains and we still have to exhaust the free variables. Rather than doing this, Combined collision attacks like FTC, also exploit the key rank information ξ_j $(1 \leq j \leq 16)$ of 16 sub-keys provided by a divide-and-conquer distinguisher like TA used in this paper to avoid exhaustion. In this case, we can recover the key with a high probability by only guessing a small part of candidates with largest probabilities. Unlike the classical collision detection function defined in Eq. 7, it can be defined as:

$$\phi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right) = \begin{cases} 1, & if \;\; \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \left\{\delta_{j_1,j_2}^{1}, \ldots, \delta_{j_1,j_2}^{\tau_d}\right\} \\ 0, & else \end{cases}. \tag{9}$$

in combined collision attacks $(i_1 \leq \tau_k, i_2 \leq \tau_k)$. Here ξ_j^i denotes the i^{th} candidate on the ranked list ξ_j of k_j and $\delta_{j_1,j_2}^1, \ldots, \delta_{j_1,j_2}^{\tau_d}$ are the ranked δ-s in CCA according to their correlation coefficients. CCA only determines δ-s, while combined collision attacks determine specific collisions between sub-keys. For simplicity, we use $\xi_{j_1}^{i_1} \leftrightarrow \xi_{j_2}^{i_2}$ to represent this pair of collision, which is a candidate of $\underline{k_{j_1} \leftrightarrow k_{j_2}}$. It includes information of $\xi_{j_1}^{i_1}$, $\xi_{j_2}^{i_2}$ and collision value $\delta_{j_1,j_2}^{i_3} = \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2}$. Based on these concepts, we introduce TC, FTC and GCA in the following subsections.

3.1 Fault Tolerant Chain

Bogdanov et al. proposed Test of Chain (TC) in [BK12], which attempted to find a long chain including 15 pairs of collisions $k_1 \leftrightarrow k_2$, $k_2 \leftrightarrow k_3$, ..., $k_{15} \leftrightarrow k_{16}$ from the first sub-key to the 16^{th} sub-key, but failed to provide a practical scheme to construct the chain. Wang et al. provided the first practical scheme for collision chain construction named Fault Tolerant Chain (FTC) in [WWZ14]. FTC tries to find collisions between the first sub-key and the other 15 sub-keys: $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$, ..., $k_1 \leftrightarrow k_{16}$. If there is no collision between k_1 and k_j $(2 \leq j \leq 16)$, which means that all possible guessing values of k_1, k_j and $k_1 \leftrightarrow k_j$ are not within the thresholds τ_k and τ_d simultaneously, then FTC assumes that an error has occurred. Its performance is notably influenced by k_1. Another shortcoming of FTC is that it exhausts the sub-keys identified as errors. Its complexity reaches $2^8 \cdot \left(2^8\right)^{n_o} \cdot \binom{15}{n_o}$ if n_o sub-keys are error (see Section 3.2 in [WWZ14]). For example, if errors happen on two sub-keys, up to 2^{31} candidates have to exhaust. If too many errors occur, FTC has to exhaust a very large space, which is infeasible. From the perspective of collision information utilization, FTC and TC only exploit 15 pairs of collisions. Since there is a pair of collision between any two sub-keys and 120 pairs of collisions in total, they waste most of the collision information. It's noteworthy that although FTC is not very efficient, it triggers new opportunities for combined collision attacks.

3.2 Group Collision Attack

Group Collision Attack(GCA) [OWSZ19] divides the 16 sub-keys into 4 non-overlapping groups of equal size, and performs the first round of bad candidates deletion within them. It is difficult to make full use of collision information within or among groups. GCA alleviates this by continuously splicing short chains to obtain long chains. The corresponding ranks of an experiment that CCA and guessing theory optimized TA are performed on randomly selected 120 power traces are shown in Tables 1 and 2 (see Section 4). Thresholds τ_d and τ_k are set to 10, and the correct sub-keys are 212, 153 and 17 as bolded, and their probabilities are also given in Table 1. The rank of them is 167.

Table 1: The ranked candidates of 3 sub-keys in TA.

	k_1		k_2		k_3	
rank	value	p	value	p	value	p
1	161	0.3843	**153**	0.2536	178	0.1465
2	104	0.1512	194	0.1106	162	0.1256
3	25	0.1135	187	0.1038	236	0.1145
4	**212**	0.0918	85	0.0873	180	0.1041
5	231	0.0632	35	0.0790	201	0.0959
6	133	0.0468	110	0.0594	35	0.0721
7	17	0.0420	185	0.0424	97	0.0486
8	211	0.0318	226	0.0284	173	0.0314
9	131	0.0106	83	0.0227	186	0.0271
10	243	0.0032	37	0.0213	**17**	0.0235

Table 3: The collisions in TA combined with CCA.

$\delta_{1,2}$	$\delta_{1,3}$	$\delta_{2,3}$
77	121	**136**
154	188	128
139	29	80
169	205	106
142	94	218
22	**197**	118
75	110	198
234	143	241
242	39	187
99	218	170

$k_1 \leftrightarrow k_2$		$k_2 \leftrightarrow k_3$		$k_1 \leftrightarrow k_3$	
161	194	153	201	104	178
161	83	**153**	**17**	104	173
104	35	194	180	104	17
104	37	187	97	212	201
212	**153**	187	17	212	173
212	194	85	35	212	186
		110	180	**212**	**17**
		226	178	133	162
		83	162	17	173
		37	173	243	173

There are 6, 10 and 10 chains for collisions $k_1 \leftrightarrow k_2$, $k_2 \leftrightarrow k_3$ and $k_1 \leftrightarrow k_3$ (as shown in Table 3). For simplify, the scores are the products of their corresponding probabilities in this example[2]. The long chain $k_1 \leftrightarrow k_2 \leftrightarrow k_3$ splices from $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$ if they have the same k_2. In this case, there have 6 candidates satisfying the collision conditions(as shown in Table 4). If collision $k_1 \leftrightarrow k_3$ is also considered, then only 3 candidates remain (as shown in Table 5). In these two cases, the size of chain space drops from 60 ($6 \cdot 10$) in Table 3 to 6 and 3 in Tables 4 and 5, respectively. Thus, combined collision attacks transform the original guessing space to much smaller collision chain space and make the key recovery easier. This is intuitively embodied in the more advanced of key ranking(i.e. the rank of the correct chain $212 \leftrightarrow 153 \leftrightarrow 17$ drops from 167 in TA to 5 and 2).

GCA splices two chains $k_1 \leftrightarrow k_2 \leftrightarrow k_3$ and $k_2 \leftrightarrow k_3 \leftrightarrow k_4$ to get a longer chain $k_1 \leftrightarrow k_2 \leftrightarrow k_3 \leftrightarrow k_4$ if they have the same guessing values on k_2 and k_3. In this case, a group in GCA is successfully built. In order to facilitate the second round of deletion, two sub-keys overlap between two adjacent groups. GCA builds other 2 groups $k_3 \leftrightarrow k_4 \leftrightarrow k_5 \leftrightarrow k_6$ and $k_5 \leftrightarrow k_6 \leftrightarrow k_7 \leftrightarrow k_8$. It splices groups $k_1 \leftrightarrow k_2 \leftrightarrow k_3 \leftrightarrow k_4$ and $k_5 \leftrightarrow k_6 \leftrightarrow k_7 \leftrightarrow k_8$ if the overlapping values from k_3 to k_6 are also a group, thus

[2]Multiplication can be converted into addition using logarithm of probabilities (e.g., $\log_2(p_1 \cdot p_2) = \log_2(p_1) + \log_2(p_2)$) in key enumeration. For simplify, the sums instead of products are exploited like [GGP+15, Gro18]. This strategy is also exploited in the rest of this paper.

Table 5: Chains include $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$ and $k_2 \leftrightarrow k_3$.

$k_1 \leftrightarrow k_2 \leftrightarrow k_3$			score
161	194	180	0.004424
212	153	201	0.002232
161	83	162	0.001095
212	194	180	0.001056
212	**153**	**17**	0.000547
104	37	173	0.000101

$k_1 \leftrightarrow k_2 \leftrightarrow k_3$			score
212	153	201	0.002232
212	**153**	**17**	0.000547
104	37	173	0.000101

obtains a bigger group including 8 sub-keys (see [OWSZ19] for details). In this case, GCA uses up to 40 pairs of collisions to remove bad candidates within 8 groups (including $k_1 \leftrightarrow k_2 \leftrightarrow k_{15} \leftrightarrow k_{16}$).

4 CFCA with Guessing Theory

GCA exploits information from up to 40 δ-s for its two rounds of bad key candidates deletion. It leaves us a much smaller chain space than TC and FTC. However, GCA also has its limitations: (1) The algorithm is too time-consuming since GCA needs to constantly perform splicing, and (2) it exploits up to 40 δ-s and most collisions are wasted, which greatly limits its ability to remove the bad candidates. When the sub-keys and collision values are ranked in much deeper spaces, GCA quickly fails. Moreover, as we stated in Section 1.2, combined collision attacks are independent of specific distinguishers used. However, both FTC and GCA combined CCA with significantly higher performance CPA in [BK12, OWSZ19]. In order to exploit collisions from CCA, it is necessary to set τ_d much larger than τ_k. In this section, we propose a new collision attack named Minkowski Distance enhanced Collision Attack (MDCA) with performance close to TA. We further design a more efficient combined collision attack named Combined Full Collision Attack(CFCA) from them, which can quickly exploit the largest amount of collision information to remove the bad candidates.

4.1 Minkowski Distance enhanced Collision Attack

CCA's efficiency is much lower than TA's. If TA and CCA are directly performed on the POIs described in Section 2.1, their performance will still exhibit a big gap. In this case, we use the sum of two γ^{th}-order Minkowski distances:

$$\sqrt[\gamma]{\sum_{\alpha=0}^{255}\left(t_1^{\alpha}-\mathbf{m}_2^{\alpha\oplus\delta_{1,2}}\right)^{\gamma}}+\sqrt[\gamma]{\sum_{\beta=0}^{255}\left(t_2^{\beta}-\mathbf{m}_1^{\beta\oplus\delta_{1,2}}\right)^{\gamma}} \tag{10}$$

to build a new collision attack named Minkowski Distance enhanced Collision Attack (MDCA). Here α and β denote the encrypted first and second plaintext bytes, and power consumption $t_1^{\alpha}\left(t_2^{\beta}\right)$ of the POIs of S-box 1(2) is matched with mean power consumption vector $\mathbf{m}_2$ ($\mathbf{m}_1$) of S-box 2(1) under the guessing $\delta_{1,2}$. MDCA is not a kind of CCA, and its performance is much closer to TA compared to CCA, thus making the combined collision attack built from TA and MDCA more meaningful and practical (see experiments on Section 6.1). Actually, the 2^{nd}-order Minkowski distance is Euclidean distance, which we may be familiar with. In principle, higher-order MDCA will achieve better performance. It achieves performance close to guessing theory optimized TA on the 4^{th}-order ($\gamma = 4$), which makes it more challenging to recover the key from their combination. Therefore, we exploit 4^{th}-order MDCA in this paper.

4.2 The Algorithm CFCA

Actually, the method to exploit all collisions can be very simple. Considering the given thresholds τ_k and τ_d, if the current length of the q^{th} chain (i.e. the number of sub-keys included) is n_l, then the number of collisions between the chain and the ranked candidate ξ_j^i $(1 \le i \le \tau_k)$ of k_j in TA is:

$$\Phi\left(\mathrm{C}_q^{n_l}, \xi_j^i\right) = \sum_{r=1}^{n_l} \phi\left(\mathrm{C}_q^{n_l}[r], \xi_j^i\right). \tag{11}$$

Here $\mathrm{C}_q^{n_l}[r]$ denotes the candidate of the r^{th} sub-key on the chain, and the collision detection function $\phi\left(\mathrm{C}_q^{n_l}[r], \xi_j^i\right)$ satisfies Eq. 9. Let $\left[\mathrm{C}_q^{n_l}, \xi_j^i\right]$ denote the possible chain constituting of ξ_j^i and $\mathrm{C}_q^{n_l}$, then the total number of collisions of it is:

$$\Gamma\left(\left[\mathrm{C}_q^{n_l}, \xi_j^i\right]\right) = \Gamma\left(\mathrm{C}_q^{n_l}\right) + \Phi\left(\mathrm{C}_q^{n_l}, \xi_j^i\right). \tag{12}$$

Following the combined collision attack, we name this scheme exploiting all collision information as Combined Full Collision Attack(CFCA), and the corresponding chains as Full-Collision Chains. CFCA exploits collisions efficiently but without any fault-tolerant mechanism. If any collision value δ between ξ_j^i and $\mathrm{C}_q^{n_l}$ is beyond τ_d, the combination of them is abandoned. Therefore, we need to set a large τ_d. CFCA exploits the largest number of collisions, and leaves us the smallest chain space in theory. Therefore, it has strong key recovery ability when both τ_d and τ_k are large enough to make all sub-keys and δ-s within them.

4.3 Typical Application Scenarios

We aim to conquer the key from very deep space in this paper, another 120 power traces are randomly selected to perform TA and MDCA, and the experimental results are shown in Fig. 2. The rank of the deepest sub-key k_3 is 51. The rank of the deepest collision value is 72 between k_5 and k_{14}. Therefore, in order to ensure that CFCA can recover the key, the thresholds τ_k and τ_d should be set to at least 51 and 72 respectively. In other words, we need to find the correct chain from such two huge spaces. Fig. 2(b) shows that 9 of the 15 δ-s between k_1 and $k_2 \sim k_{16}$ exceed threshold $\tau_d = 1$, which poses a problem for FTC as there are too many sub-keys to be exhaustively searched. This paper considers FTC under threshold $\tau_d > 1$ for the first time.

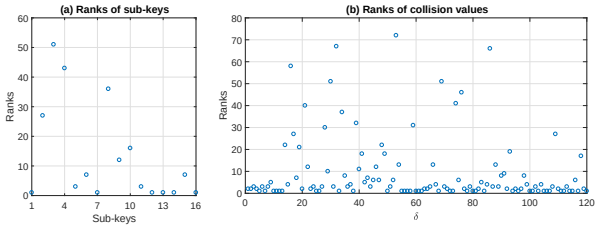

Figure 2: The ranks of sub-keys (a) and correct δ-s (b).

We use the histogram based key rank estimation in Algorithm 1 in [GGP+15] to estimate the security levels, the corresponding lower and upper bounds given in Section 3.2 in [GGP+15] are also exploited. It's worth noting that key rank estimation of combined collision attacks is different from that of TA, since the "key rank" we obtain in combined collision attacks is very different from the one in classical key rank estimation. Specifically,

for FTC, the 15 sub-keys $k_2 \sim k_{16}$ are only related to k_1 and are independent of each other. Therefore, classical key rank estimation can be performed on the scores of chains (see Section 3.2) from each candidate of k_1 separately and the key ranking can be estimated by the sum of these estimations. The 16 sub-keys are divided into 4 big "groups" in GCA, thus scores of each group are computed and we obtain 4 lists of scores. We then perform classical histogram based key rank estimation on them.

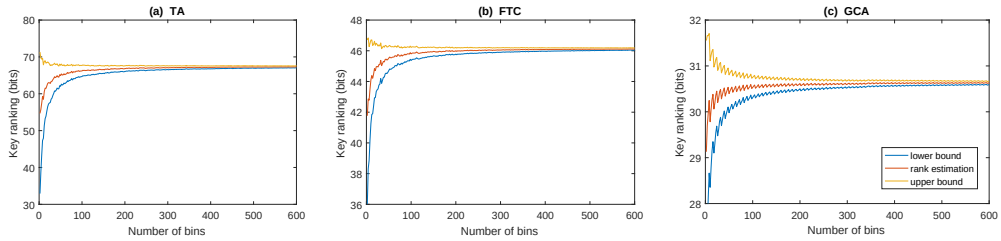

Figure 3: Key rank estimation on the original space of TA, and the remaining chain space of FTC and GCA.

We set τ_k to 52 and τ_d to 72, thus all sub-keys and δ-s are within thresholds. The results of the above described key rank estimation after TA, FTC and GCA are shown in Fig. 3. As there are only about 300 full-collision chains in CFCA, their scores can be ranked directly. Traditional key enumeration, FTC and GCA optimize the computing power from $16 \cdot \log_2 52 = 2^{91.207}$ within threshold $\tau_k = 52$ to $2^{67.51}$, $2^{46.06}$ and $2^{30.65}$. The time consumption of GCA, FTC and CFCA is about 0.444, 9.471 and 1.055 seconds. FTC and GCA reduce the difficulty of key recovery, but the following "key enumeration", which enumerates the scores of full-key candidates from the largest to the smallest, is still very time-consuming. CFCA exhibits its powerful search capability under very large thresholds. It leaves us a very small space when both τ_k and τ_d are reasonably set, makes it possible for the attacker to directly exhaust the chain space.

The number of remaining chains under the thresholds near $\tau_k = 52$ and $\tau_d = 72$ are shown in Fig. 4. Due to the fact that only a few δ-s are among the first several sub-keys (e.g. the j^{th} $(2 \leq j \leq 16)$ subkey has $j-1$ δ-s between it and the first $j-1$ sub-keys), a very limited number of bad candidates can be removed. Less chains satisfy collision condition in the next sub-keys and reduction on the number of chains happens. This is vividly demonstrated in Fig. 4, where the number of candidates increases rapidly on the 3^{rd} to 5^{th} sub-keys, but only hundreds are remained after the 10^{th} sub-key. This means, construction of longer chains in CFCA becomes faster, which just accounts for only a very small part of the algorithm overhead. The number of remaining chains is almost the same under several close thresholds τ_d and τ_k. This fully illustrates the superiority of CFCA that we do not need to adjust the thresholds deliberately, as small changes of them will not significantly affect the computation overhead.

4.4 Guessing Theory based Optimization

The existing combined collision attacks set a fixed τ_k for all sub-keys. On one hand, the location of some sub-keys is relatively advanced, we only need to guess a small number of candidates before reaching them. In this case, guessing deeper locations wastes computing power. On the other hand, a part of sub-keys are ranked very deeply, we need to set a large τ_k to contain them. Therefore, it is more reasonable to define a τ_k flexibly for each sub-key according to the results of TA, rather than a fixed one for all sub-keys.

We convert multiplication to addition using logarithm of probabilities in TA (see Foot-

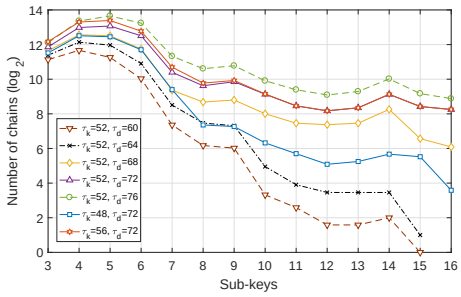

Figure 4: The number of chains under each sub-key.

note 2). We then normalize and rank the probabilities of the j^{th} sub-key (with candidate space $\mathbb{K}_j$) output by TA in descending order, and obtain $p_j = \left\{p_j^1, p_j^2, \ldots, p_j^{|\mathbb{K}_j|}\right\}$ satisfying:

$$\sum_{i=1}^{|\mathbb{K}_j|} p_j^i = 1 \tag{13}$$

as introduced in [CP17], where p_j^i is the i^{th} largest probability. A metric considers only the candidates that satisfy the given probability value $\alpha \in (0, 1)$:

$$\mu_\alpha\left(\mathbb{K}_j\right) = \min\left\{\tau_k \Big| \sum_{i=1}^{\tau_k} p_j^i \geq \alpha\right\}, \tag{14}$$

which is named as α-$work$-$factor$ in [Pli00]. Here τ_k is our guessing length on k_j. Compared with setting a unified threshold τ_k for all sub-keys, guessing theory enables us to define a more flexible and reasonable threshold for each sub-key separately based on our computing power. The success probability and the candidate space:

$$\Omega = \prod_{j=1}^{16} \mu_\alpha\left(\mathbb{K}_j\right) \tag{15}$$

under the given thresholds can be estimated quickly. For example, we set $\alpha = 0.996$ in our paper, the corresponding theoretical success probability is $\alpha^{16} = 0.996^{16} = 0.9379$. We rank sub-keys in ascending order according to their guessing lengths $\mu_\alpha\left(\mathbb{K}_j\right)$ $(1 \leq j \leq 16)$ to lower computational complexity. The average number of candidates of the re-ranked sub-keys under different numbers of power traces(denoted as n) with 400 repetitions is shown in Fig. 5.

The sub-keys are re-ranked before recovery according to their number of candidates provided by guessing theory, which makes CFCA more efficient. On one hand, although all collisions of the sub-keys re-ranked in the front is fully utilized by our CFCA, the candidates that can be excluded are limited, but the number of candidates of them is also small(see Fig. 5). In this case, although the candidates will be left with a high probability, the number of remaining chains is still very small. This can lower the heights of the first several sub-keys in Fig. 4 and make CFCA work faster. On the other hand, there are a lot of candidates for sub-keys re-ranked backward. CFCA can exploit more collisions to remove bad candidates from them, so that the number of remaining chains is not too large. This significantly reduces the computational complexity of CFCA, and leaves us a small chain space, which facilitates the future key recovery.

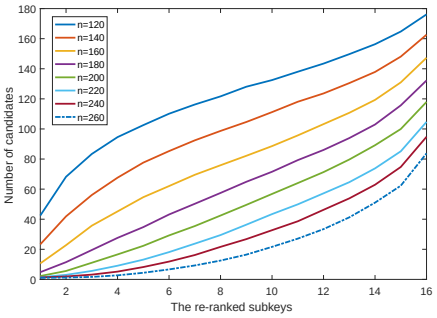

Figure 5: The number of candidates of the re-ranked sub-keys under $\alpha = 0.996$.

5 Error Tolerance

5.1 Distribution of Collision Values Beyond τ_d

CFCA can efficiently remove bad key candidates by maximizing collision exploitation, thus quickly reduces candidate space. Therefore, the thresholds τ_k and τ_d can be set quite large under CFCA, and the number of remaining chains will be much smaller compared to FTC and GCA. If the key is within the threshold, it can be recovered easily. However, CFCA requires all δ-s to fall within threshold τ_d. There may be a few δ-s in much deeper space than we have discussed in Section 4.3, which may make the runtime of CFCA unbearable. In this case, it requires fault tolerance on CFCA that allows a small number of δ-s to be beyond threshold τ_d. In other words, we need to consider the chains that are not so "full", but "partially full".

We delimit the collision $k_{j_1} \leftrightarrow k_{j_2}$ on k_{j_2} for fault tolerance. An interesting phenomenon is that the δ-s outside the threshold τ_d do not occur randomly, but are related to only one or several sub-keys. Take experimental results shown in Fig. 2 for example, δ-s beyond different τ_d-s are shown in Table 6. There are only 1,1,1,1,2 δ-s for k_3, k_4, k_5, k_6 and k_{10} beyond $\tau_d = 50$. Only 4 δ-s should be fault tolerated if τ_d reaches 52. In fact, this can be analyzed from Signal-to-Noise Ratio(SNR) of the selected power traces and the distinguisher used. These effects come from many aspects, such as the difficulty to align the power traces of all S-boxes strictly, the noise from chip and sampling equipments, and different performance of distinguishers. We randomly select 160 of the 50,000 power traces to perform MDCA, count the number of δ-s beyond τ_d from 10 to 80, and analyze success probability if these δ-s can be well concentrated on the given number of sub-keys for their fault-tolerance. The experimental results are shown in Figs. 6 and 7.

Table 6: δ-s beyond different τ_d-s.

τ_d	collisions beyond τ_d
$20 \sim 30$	$k_1 \leftrightarrow k_{15}$, $k_2 \leftrightarrow k_4$, $k_2 \leftrightarrow k_6$, $k_2 \leftrightarrow k_{15}$, $k_4 \leftrightarrow k_{10}$, $k_{11} \leftrightarrow k_{15}$
$30 \sim 40$	$k_2 \leftrightarrow k_8$, $k_3 \leftrightarrow k_8$, $k_3 \leftrightarrow k_{13}$, $k_5 \leftrightarrow k_{10}$
$40 \sim 50$	$k_6 \leftrightarrow k_{15}$, $k_7 \leftrightarrow k_8$
$50 \sim 60$	$k_2 \leftrightarrow k_3$, $k_3 \leftrightarrow k_4$, $k_6 \leftrightarrow k_{10}$
$60 \sim 70$	$k_3 \leftrightarrow k_6$, $k_8 \leftrightarrow k_{10}$
$70 \sim 80$	$k_4 \leftrightarrow k_5$

To achieve a success rate 0.90, we need to perform fault tolerance on δ-s of average 11, 9, 7, 6, 5.5 and 5 sub-keys so that all remaining δ-s are within τ_d (as shown in Fig. 6). It can also be seen from Fig. 7 that the number of δ-s beyond τ_d is smaller than 40, 30, 20,

15, 10, 10 when τ_d is 10, 20, 30, 40, 50 and 60, which also fully illustrates that δ-s beyond the given τ_d are not distributed randomly, but can be concentrated on a few sub-keys for their fault tolerance.

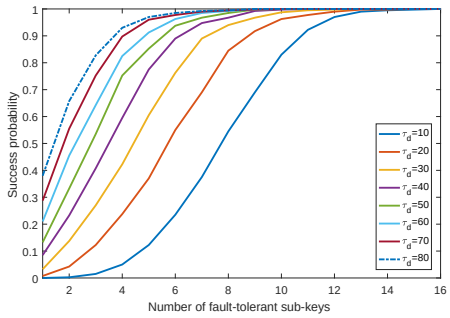

Figure 6: Success probabilities if δ-s beyond τ_d can be fully concentrated on the given number of sub-keys for their fault tolerance.

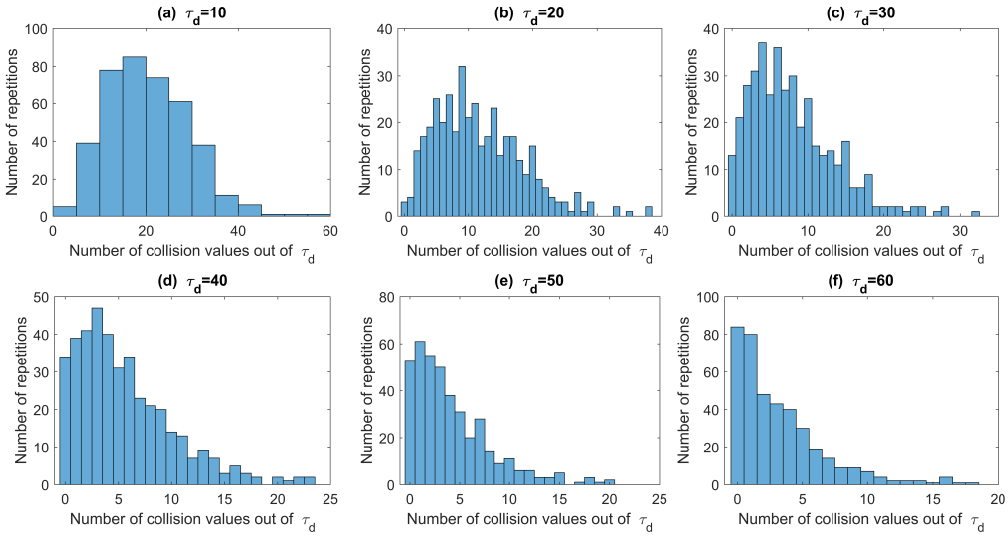

Figure 7: Distributions of δ-s beyond threshold τ_d when $n = 160$.

5.2 Optimization

δ-s beyond τ_d can be concentrated on several sub-keys for fault tolerance. For example, if we perform rotated fault-tolerance on δ-s beyond τ_d on 3 (4) sub-keys in Fig. 6, the success rate can reach up to 0.75 (0.90) when $\tau_d = 70$ (80). Rotated fault-tolerance here divides sub-keys into full-collision and fault-tolerant parts. Here δ-s between any two sub-keys in the full-collision part are within τ_d, and others in fault-tolerant part or between two parts are not strictly required. Therefore, the rotated fault-tolerance on δ-s on 2 (3) sub-keys requires $\binom{16}{2} = 120$ $\left(\binom{16}{3} = 560\right)$ rounds of computation respectively. Although it

can significantly make τ_d smaller, it is still very time-consuming and a large number of collisions are detected repeatedly. Here we propose a new scheme named Block based Fault-Tolerant CFCA (BFT-CFCA), in which several sub-keys are "bundled" together for fault tolerance, so that each large "block" allows a fixed number of δ-s beyond τ_d without strictly limiting the specific sub-key of faults, so as to achieve good fault-tolerant performance.

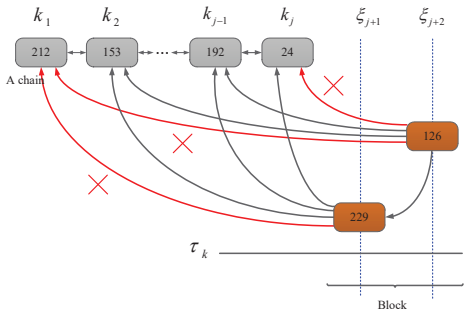

Figure 8: BFT-CFCA performs fault tolerance on a block with 3 δ-s beyond τ_d.

We only consider the case that each block includes two sub-keys in this paper. For a chain shown in Fig. 8, the total number of collisions between it and the candidates $\left[\xi_{j+1}^{i_{j+1}}, \xi_{j+2}^{i_{j+2}}\right]$ of the two sub-keys within the block $[k_{j+1}, k_{j+2}]$ is:

$$n_t = \sum_{\kappa=j+1}^{j+2} \sum_{r=1}^{n_l} \phi\left(\mathrm{C}_q^{n_l}[r], \xi_\kappa^{i_\kappa}\right) + \phi\left(\xi_{j+1}^{i_{j+1}}, \xi_{j+2}^{i_{j+2}}\right). \tag{16}$$

Here $n_l = j$, $1 \le i_j \le \mu_\alpha(\mathbb{K}_j)$ and $\phi\left(\xi_{j+1}^{i_{j+1}}, \xi_{j+2}^{i_{j+2}}\right)$ is the collision detection between $\xi_{j+1}^{i_{j+1}}$ and $\xi_{j+2}^{i_{j+2}}$. It is noteworthy that blocks can be either larger or smaller, or different from fault tolerance on δ-s of independent sub-key. If we allow a total number of 5 δ-s to be beyond τ_d on this block, this covers all cases where at most 5 δ-s for all sub-keys in the block are out of τ_d. As mentioned in Section 5.1, δ-s beyond τ_d only occur on some sub-keys and the distribution of them is relatively scattered if τ_d is set reasonably. Therefore, the number of δ-s allowed to be beyond τ_d in BFT-CFCA does not need to be very large. Moreover, BFT-CFCA avoids rotated fault tolerance and effectively reduces the amount of computation, which can be verified through the experiments in Section 6.

5.3 Fault-Tolerant Vector and Its Adjustment

We illustrate the distributions of δ-s beyond different τ_d-s in Fig. 7. For effective fault tolerance, we can profile a large number joint distributions of these δ-s of several sub-keys. In fact, the adjustment of fault-tolerant threshold can be far less complicated, since the power traces corresponding to the first several re-ranked sub-keys usually have high SNR and MDCA performs well in the guessing theory optimized CFCA. In addition, the rapid growth in the number of collisions mainly occurs in the last several sub-keys. As such, it is not complex to set a good fault-tolerant threshold. Therefore, we can allow different numbers of δ-s to be beyond τ_d for each block, and the fault-tolerant δ-s of these blocks together constitute the Fault-Tolerant Vector (FTV), which is very "thin" in front and "fat" in backward. For example, if we allow 1, 2, 3, 4 ,5, 6 and 7 δ-s to be beyond τ_d for the 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} and 8^{th} blocks independently, then we

get a FTV= $(1, 2, 3, 4, 5, 6, 7)$. Since there is only one δ in the first block built on the re-ranked first and second sub-keys. To avoid exhaustion, we do not consider its fault-tolerance in FTV. Similarly, in order to avoid exhaustion, it would be better if the number of fault-tolerant δ-s is smaller than the number of δ-s of the first sub-key in each block.

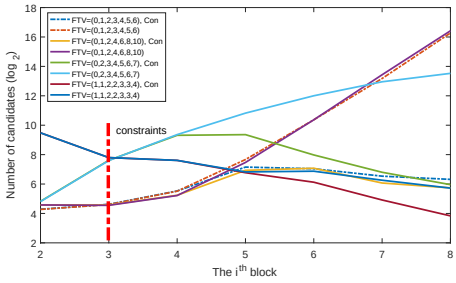

Figure 9: The number of chains under different FTV-s with and without constraints.

An important advantage of BFT-CFCA is that it enables us to decide whether a block should be "fat" or "thin" according to our computing power. To make a block "thin"(i.e. fewer chains on the block), we just need to set the corresponding fault-tolerant threshold smaller in FTV. Four FTVs in Fig. 9 are selected to illustrate the adjustment and their runtime are given in Table 7. The total number of chains under FTV=$(1, 1, 2, 2, 3, 3, 4)$ decreases and BFT-CFCA consumes very little time for each block. The following several blocks have more chains to detect when one δ in the first block $[k_3, k_4]$ is allowed to be beyond τ_d. The number of chains under FTV=$(0, 2, 3, 4, 5, 6, 7)$ and FTV=$(0, 1, 2, 4, 6, 8, 10)$ is close. The latter uses larger thresholds, thus having more chains and consuming more time. FTV=$(0, 2, 3, 4, 5, 6, 7)$ shows more chains in the middle of the curve, then the number decreases. In this case, collision detection is performed on a large number of chains when each block is added, which makes BFT-CFCA very time-consuming.

Table 7: Mean time consumption of 4 FTV-s with or without constraints (seconds).

FTV	with constraints	without constraints
FTV=(1,1,2,2,3,3,4)	22.40	26.73
FTV=(0,1,2,3,4,5,6)	66.838	778.826
FTV=(0,2,3,4,5,6,7)	25.2723	1505.44
FTV=(0,1,2,4,6,8,10)	25.1757	1974.58

The correct δ-s can fall into τ_d with a high probability under a large τ_d, thus the full-key contains almost the largest number of collisions. Moreover, the FTV we set considers the case that many δ-s are beyond the threshold τ_d randomly. In fact, only a few δ-s are beyond τ_d and the distribution is not so dispersed. This goes back to our original conclusion that δ-s beyond τ_d can be concentrated on a few sub-keys for their fault tolerance. Therefore, we further add a constraint: a chain has almost $\tau_c = 5$ δ-s less than the chain with the largest number of collisions are allowed to be beyond τ_d after the third block. Otherwise, it will be discarded. Since it is hard to find such chains directly, each chain satisfying such collision condition in current block is saved and deletion performs when considering the next block. In this case, the two candidates in current block $[k_{j+1}, k_{j+2}]$ with the largest number of collisions satisfy:

$$n_{t_{max}} = \max \left\{ \sum_{\kappa=j+1}^{j+2} \Phi\left(\mathrm{C}_q^{n_l}, \xi_\kappa^{i_\kappa}\right) \middle| i_\kappa = 1, 2, \ldots, \mu_\alpha\left(\mathbb{K}_\kappa\right) \right\}, \tag{17}$$

and the maintained chains should satisfy:

$$n_t \geq n_{t_{max}} + \Gamma\left(\mathrm{C}_{q_{max}}^{n_l}\right) - \tau_c. \tag{18}$$

Here $n_l = j$, the constraint $\tau_c = 5$,

$$q_{max} \leftarrow \arg\max_q \left\{\Gamma\left(\mathrm{C}_q^{n_l}\right) | q = 1, 2, \ldots, n_c\right\}, \tag{19}$$

and n_c is the current number of chains. The corresponding experimental results have been shown in Fig. 9 and Table 7. The number of chains can be reduced from thousands to hundreds, thus significantly improving efficiency. The experimental results in Section 6 will show that the success rate under the constraints $\tau_c = 5$ is close to that without constraints. This also indicates that the sub-keys and their δ-s are mostly within τ_d and τ_k when both thresholds are relatively large, and the key may become one of the chains with almost the largest number of collisions as we stated before.

6 Experimental Results

We compare the performance of FTC, GCA, CFCA and BFT-CFCA under different thresholds τ_d and different numbers of traces n, respectively. Success Rate (SR) [SMY09], one of the most widely used criteria to evaluate the performance of side-channel distinguishers, is exploited here. As we have explained in Section 5.3, our BFT-CFCA with constraints performs very well under very large FTV. Therefore, we set a very large FTV= $(0, 1, 2, 4, 6, 8, 10)$ and allow at most 5 collisions less than the chain with the largest number of collisions. Since some of the 16 sub-keys of AES-128 within thresholds but unsatisfying the given collision conditions are removed under FTC, GCA, CFCA and BFT-CFCA, and we only consider key recovery from theoretical success probability $\alpha^{16} = 0.996^{16} = 0.9379$, SR will be smaller than this.

It is worth noting that comparing success rates of different combined collision attacks alone may not be enough. The candidate space can be significantly reduced after these attacks, but the key may still rank very deeply, since τ_d and τ_k are set largely, and very large space is considered in this paper. Therefore, we exploit success rate under given computing power(i.e. the enumeration power) to compare the performance of these attacks, similar evaluation metric has been used in [PGS15]. Histogram based key rank estimation after attacks introduced in Section 4.3 is used to illustrate the superiority of our CFCA and BFT-CFCA. As mentioned in Section 4.3, the key rank under combined collision attacks is very different from that under the classic divide-and-conquer distinguishers. Key recovery like "key enumeration" on CFCA and FTC are the fastest, GCA follows, and TA is the slowest. Since CFCA, FTC and GCA have sub-chains with lengths 16, 16 and 4 respectively, thus only 1, 1 and 4 lists are considered when "enumeration", compared with 16 lists in TA. The evaluations under large thresholds τ_d and different numbers of traces are very time-consuming, we only repeat each experiment 200 times.

6.1 Performance of MDCA

We use 1^{st}-, 10^{th}- and 20^{th}-order Success Rate [SMY09] to evaluate the performance of TA, CCA and our MDCA under different numbers of traces, and the experimental results under 400 repetitions are shown in Fig. 10. TA's first-order success rate is even significantly higher than CCA's 20^{th}-order success rate. The success rates of MDCA and CCA are about 1 and 0.1 under 600 traces, respectively. The success rate of CCA reaches about 1 under 2000 traces. The performance of our MDCA is very close to that of TA, and their combination will be more challenging and meaningful as we stated before. We try to exploit the combined collision attacks to achieve significantly higher success rate than performing them separately under the given computing power.

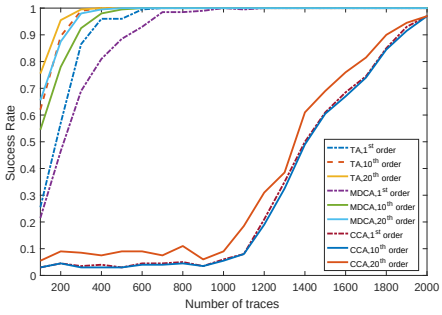

Figure 10: The success rates of TA, CCA and our MDCA.

6.2 Experiments on Different Thresholds τ_d

The success rate under different thresholds τ_d when $n = 160$ is shown in Fig. 11. Wang et al. only discussed the case $\tau_d = 1$ in [WWZ14], and ignored the complex case $\tau_d > 1$. Ou et al. compared GCA considering $\tau_d > 1$ with FTC in [OWSZ19]. However, τ_d was always set to 1. If none of the τ_k guesses of sub-key k_j $(2 \le j \le 16)$ collide with the τ_k guesses of k_1, then there is no collision between k_1 and k_j. In this case, the attacker has to perform brute-force search on k_j. However, this probability becomes very small when τ_d is large. Since compared with τ_k pairs, a total of $\tau_k \cdot \tau_d$ detections make it easier to find collisions. In this case, it is almost impossible for FTC to collide with any guessed values of k_j within the threshold τ_k. This conclusion can also be drawn from Fig. 13.

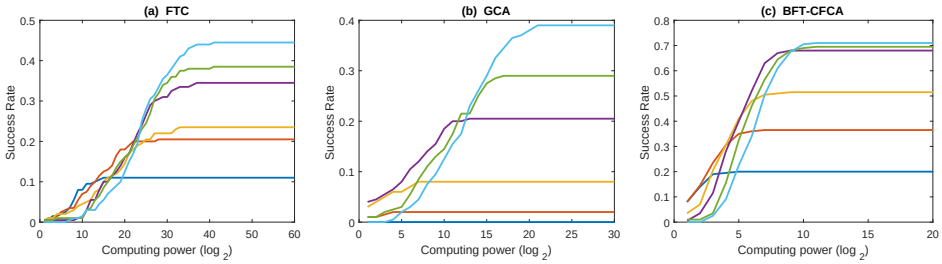

Figure 11: Success rate under different thresholds τ_d.

The growth of τ_d leads to more δ-s falling within it, making the collision conditions easier to be satisfied and more chains are remained. However, the success rate of CFCA under $\tau_d = 30$ is only 0.045, not even comparable to that of BFT-CFCA under $\tau_d = 5$ in Fig. 11. This indicates that BFT-CFCA significantly improves the performance of CFCA. Table 8 also shows that the runtime of FTC and CFCA is shorter than GCA and BFT-CFCA. GCA spends most of the time on its splicing, while BFT-CFCA spends most of the time on its fault-tolerance. Since τ_d from 5 to 30 is too small and there is no chain satisfying the collision conditions after several blocks, CFCA terminates quickly (see Table 8).

Table 8: Time consumption (seconds) under different τ_d.

τ_d	5	10	15	20	25	30
FTC	0.282	0.385	0.427	0.531	0.431	0.759
GCA	0.315	0.609	1.354	2.69	5.006	9.660
CFCA	0.018	0.026	0.039	0.058	0.089	0.148
BFT-CFCA	0.309	1.746	7.673	26.56	83.361	250.170

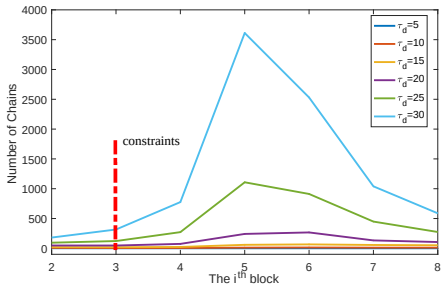

Figure 12: The number of chains under different τ_d.

The selected FTV and τ_d affect the experimental results, increasing them can improve the success rate of BFT-CFCA. However, this increases the number of remaining chains and brings a lot of extra computation(see Table 8). Since larger τ_d makes more δ-s satisfy the given conditions while the corresponding FTV($= (0, 1, 2, 4, 6, 8, 10)$) is not adjusted accordingly, too many chains with fewer collisions are formed (see Fig. 12). The choice of τ_d and FTV reflects the tradeoff between fault-tolerant time and "enumeration" time. If they are larger, BFT-CFCA takes more time to build chains but the key can be recovered from a smaller chain space. For GCA, when the number of power traces is small, such as $n = 160$ in Fig. 11, the number of remaining chains is still large. This is especially for FTC, the space is even close to 2^{40}. It will take about one day to enumerate such a large space of TA on our desktop computer. However, our BFT-CFCA can recover the key quickly from the remaining space less than 2^{10}, and it will not bring much computation. When τ_d increases to a certain height, further increasing it will not significantly improve the success rate(see Fig. 11(c)).

6.3 Experiments on Different Numbers of Traces

We also compare the performance of FTC, GCA, CFCA and BFT-CFCA under different numbers of traces n. Since the success rate of BFT-CFCA approaches the theoretical success probability under $\tau_d = 20$ and FTV$= (0, 1, 2, 4, 6, 8, 10)$, larger τ_d will not significantly improve the success rate (see Fig. 11(c)). Therefore, we set τ_d to 20. Compared with the growth of threshold τ_d, increasing the number of traces used makes the sub-keys and the corresponding δ-s fall within the thresholds more quickly. The success rates of different schemes under $100 \sim 260$ traces are shown in Fig. 13. With the increase of n, the rank of key is more advanced, and the guessing length τ_k of each sub-key becomes shorter under the same probability $\alpha = 0.996$, resulting in a rapid reduction in candidate space and runtime (see Table 9). The success rate of GCA grows slowly and finally gets close to FTC, while the success rate of BFT-CFCA is significantly higher than that of FTC. It is worth noting that although it's a very rigorous requirement that all δ-s are within τ_d, this condition becomes easier to be satisfied with the increase of n, resulting in a significantly higher success rate of CFCA. However, its success rate is almost 0 when $n = 160$, so we don't illustrate this in Fig. 11.

The success rate of our BFT-CFCA tends to be stable when the threshold τ_d reaches a certain height, and the increase in the number of power traces will also has similar effect. It can also be seen from Fig. 13 that the success rate increases slowly from 0.8 to 0.9 when $n \geq 180$, which is close to the theoretical success probability $0.996^{16} = 0.9379$. More intuitively, the success rate of BFT-CFCA is significantly higher than FTC, GCA and CFCA in Fig. 13. This fully illustrates the superiority of its fault-tolerant technique. Due to the fact that the distribution of δ-s beyond τ_d does not always satisfy the fault-tolerant vector, its success rate is still lower than the theoretical success probability. Fortunately,

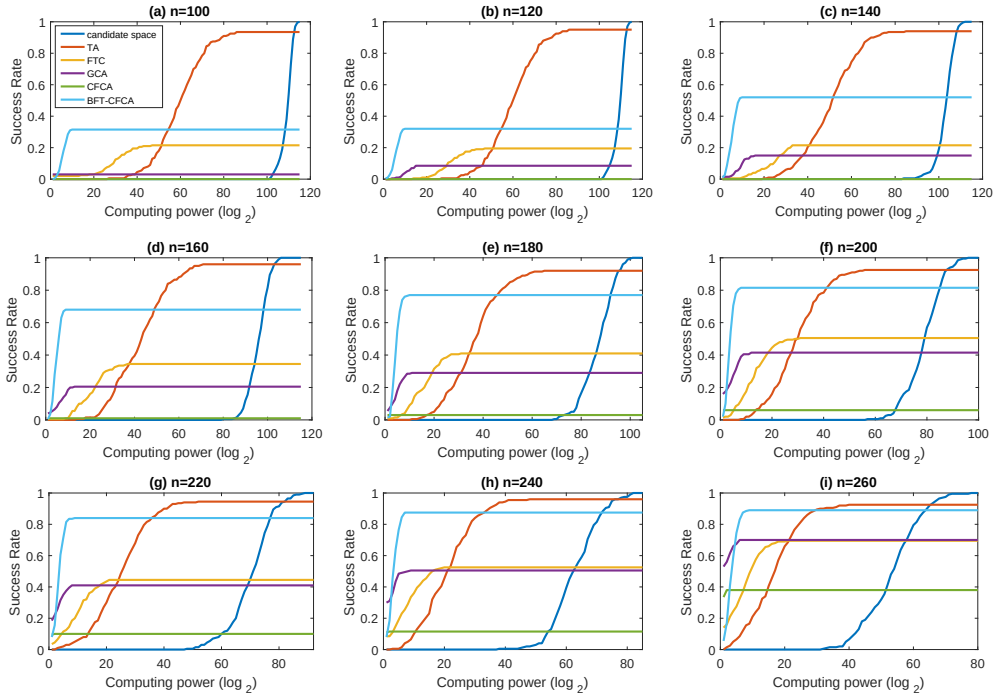

Figure 13: Success rates under different numbers of traces.

the number of chains in our BFT-CFCA does not grow explosively under reasonably restrictive conditions as shown in Fig. 14. Therefore, in order to improve the success rate, we can further adjust FTV, such as from $(0, 1, 2, 4, 6, 8, 10)$ to $(1, 1, 2, 4, 6, 8, 10)$, which can further improves the success rate without bringing too much computation.

Table 9: Time consumption (seconds) under different numbers of traces.

n	100	120	140	160	180
FTC	1.26	1.28	0.92	0.53	0.309
GCA	15.55	9.2275	5.26	2.69	1.38
CFCA	1.17	0.417	0.162	0.058	0.035
BFT-CFCA	115.21	62.19	43.26	26.56	16.36
n	200	220	240	260	–
FTC	0.116	0.071	0.037	0.028	–
GCA	0.776	0.445	0.230	0.359	–
CFCA	0.030	0.028	0.029	0.037	–
BFT-CFCA	10.35	6.62	3.70	2.96	–

CFCA discards a large number of key candidates due to its strict collision conditions, and its success rate is much lower than BFT-CFCA's. However, CFCA is simpler and we don't need to adjust FTV and the constraints(i.e. τ_c in Eq. 18). It also has a very strong key recovery ability and can reduce the same search space and the rank of key to the smallest ones compared to FTC, GCA and BFT-CFCA. These advantages of CFCA are more prominent under properly large thresholds, such as the one we discussed in Section 4.3. Therefore, we can first exploit CFCA to perform a deep search. If the key and the collision values are in the unrecoverable spaces, we can then deploy BFT-CFCA for another search.

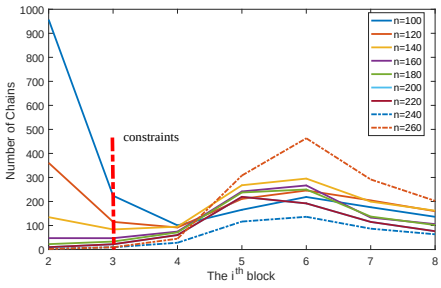

Figure 14: The number of chains under different numbers of traces.

7 Conclusion

Several existing combined collision attacks named TC, FTC and GCA, aim to recover the key from the outputs of a divide-and-conquer distinguisher and an analytical distinguisher to achieve better performance than performing them independently, which can be regarded as post-processing techniques. They discard a large number of key candidates that do not satisfy the collision conditions and leave us a collision chain space much smaller than the original one for future key recovery. They are simple, efficient, and significantly shorten key recovery time. However, the collision information exploited by FTC and GCA is very limited, and they quickly become infeasible when the key and the collision values are located in much deeper spaces.

To enable efficient key recovery from a deep space, we propose a simple yet efficient algorithm named CFCA, which is able to make full use of all collision information and further push the reachable space. We further introduce guessing theory to optimize the search order of sub-keys according to their guessing lengths. This obviously alleviates the problem that the computation is mainly focused on the first several sub-keys for the existing combined collision attacks under fixed thresholds. Benefiting from the high efficiency of its collision detection and stronger exclusion capability of bad candidates within thresholds, CFCA can search much larger candidate spaces than FTC and GCA. It applies to situations where the sub-keys and δ-s are very deep but still acceptable just like the example given in Fig. 2. However, CFCA has a restriction that all collisions must fall within the given thresholds, τ_d has to be set very large.

Based on the fact that collision values beyond the thresholds can be concentrated on several sub-keys for their fault-tolerance, we further propose a fault-tolerant scheme BFT-CFCA to improve CFCA, and exploit the fault-tolerant vector to provide a reference for its chain space adjustment. BFT-CFCA significantly reduces the thresholds used in CFCA. Hence, CFCA and BFT-CFCA provide new ways to further push the limits of reachable key space. The experimental results also demonstrate their superiority. This paper only considers fault-tolerance on collision values, and leaves the optimization of BFT-CFCA and fault-tolerance on sub-keys as open problems. It is also our future work to apply our attacks to more complex hardware devices.

References

[1] Dpa Contest. http://www.dpacontest.org/home/.

[2] M. Alioto, M. Poli, and S. Rocchi. Differential Power Analysis Attacks to Precharged Buses: A General Analysis for Symmetric-Key Cryptographic Algorithms. *IEEE Trans. Dependable Sec. Comput.*, 7(3):226–239, 2010.

[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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] C. Disselkoen, D. Kohlbrenner, L. Porter, and D. M. Tullsen. Prime+Abort: A Timer-Free High-Precision L3 Cache Attack Using Intel TSX. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017.*, pages 51–67, 2017.

[9] T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi. Side-Channel Attacks on B-LISS Lattice-Based Signatures: Exploiting Branch Tracing against Strongswan and Electromagnetic Emanations in Microcontrollers. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1857–1874, 2017.

[10] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer. ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 219–235, 2016.

[11] D. Genkin, A. Shamir, and E. Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 444–461, 2014.

[12] B. Gérard and F. Standaert. Unified and Optimized Linear Collision Attacks and Their Application in A Non-Profiled Setting: Extended Version. *J. Cryptographic Engineering*, 3(1):45–58, 2013.

[13] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 426–442, 2008.

[14] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F. Standaert. Simpler and More Efficient Rank Estimation for Side-Channel Security Assessment. In *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, pages 117–129, 2015.

[15] V. Grosso. Scalable Key Rank Estimation (and Key Enumeration) Algorithm for Large Keys. In *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers.*, pages 80–94, 2018.

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

[17] 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.

[18] H. Ledig, F. Muller, and F. Valette. Enhancing Collision Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 176–190, 2004.

[19] Y. Li, S. Wang, Z. Wang, and J. Wang. A Strict Key Enumeration Algorithm for Dependent Score Lists of Side-Channel Attacks. In *Smart Card Research and Advanced Applications - 16th International Conference, CARDIS 2017, Lugano, Switzerland, November 13-15, 2017, Revised Selected Papers*, pages 51–69, 2017.

[20] A. Moghimi, G. Irazoqui, and T. Eisenbarth. Cachezoom: How SGX Amplifies the Power of Cache Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 69–90, 2017.

[21] 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.

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

[23] 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.

[24] 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.

[25] R. Poussier, V. Grosso, and F. Standaert. Comparing Approaches to Rank Estimation for Side-Channel Security Evaluations. In *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pages 125–142, 2015.

[26] 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.

[27] K. Schramm, G. Leander, P. Felke, and C. Paar. A Collision-Attack on AES: Combining Side Channel- and Differential-Attack. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 163–175, 2004.

[28] 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.

[29] N. Veyrat-Charvillon, B. Gérard, and F. Standaert. Security Evaluations Beyond Computing Power. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 126–141, 2013.

[30] 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.

[31] W. Wang, Y. Yu, F. Standaert, J. Liu, Z. Guo, and D. Gu. Ridge-Based DPA: Improvement of Differential Power Analysis for Nanoscale Chips. *IEEE Trans. Information Forensics and Security*, 13(5):1301–1316, 2018.

[32] A. Wiemers and D. Klein. Entropy Reduction for the Correlation-Enhanced Power Analysis Collision Attack. In *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, pages 51–67, 2018.

[33] 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.