The Science of Guessing in 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. Recovering keys ranked in very deep candidate space efficiently is a very important but challenging issue in Side-Channel Attacks (SCAs). State-of-the-art Collision Optimized Divide-and-Conquer Attacks (CODCAs) extract collision information from a collision attack to optimize the key recovery of a divide-and-conquer attack, and transform the very huge guessing space to a much smaller collision space. However, the inefficient collision detection makes them time-consuming. The very limited collisions exploited and large performance difference between the collision attack and the divide-and-conquer attack in CODCAs also prevent their application in much larger spaces. In this paper, we propose a Minkowski Distance enhanced Collision Attack (MDCA) with performance closer to Template Attack (TA) compared to traditional Correlation-Enhanced Collision Attack (CECA), thus making the optimization more practical and meaningful. Next, we build a more advanced CODCA named Full-Collision Chain (FCC) from TA and MDCA to exploit all collisions. Moreover, to minimize the thresholds while guaranteeing a high success probability of key recovery, we propose a fault-tolerant scheme to optimize FCC. The full-key is divided into several big "blocks", on which a Fault-Tolerant Vector (FTV) is exploited to flexibly adjust its chain space. Finally, guessing theory is exploited to optimize thresholds determination and search orders of sub-keys. Experimental results show that FCC notably outperforms the existing CODCAs.

Keywords: FCC · fault tolerance · 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 [YGH16], power consumption [KJJ99, WYS+18], electromagnetic [GPPT16], and 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, which can be classified as divide-and-conquer and analytical. Divide-and-conquer attacks, such as Correlation Power Analysis (CPA) [BCO04] and Template Attack (TA) [CRR02], divide the huge key candidate space into small sub-keys and conquer them independently. Then, the attacker recombines the candidates of sub-keys via key enumeration [LWWW17, PSG16, VGS13]. However, key enumeration is limited by the enumeration 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 [LMV04, MME10, SLFP04], aim at recovering

the full key together by solving a system of equations. Divide-and-conquer approaches have the benefits of fast implementation and low knowledge requirements, while analytical strategies exploit more leaky information.

The idea exploiting collision information to optimize divide-and-conquer attacks was first presented in [BK12]. Two practical attacks named Fault-Tolerant Chain (FTC) and Group Collision Attack (GCA) were reported in [OWSZ19, WWZ14]. These Collision Optimized Divide-and-Conquer Attacks (CODCAs) only guess a part of the best candidates of both two attacks and try to recover the key from them. Their advantage is, the candidates of collision attacks (e.g. Correlation-Enhanced Collision Attack (CECA) [MME10]) are exploited in divide-and-conquer attacks (e.g. TA) to find specific collision-pairs, thus transforming the original huge candidate space to a much smaller collision space and significantly lowering the complexity of the latter key recovery. CODCAs are independent of specific distinguishers used, and their key recovery ability was fully demonstrated in [OWSZ19, WWZ14]. However, the above-mentioned CODCAs have difficulties in dealing with much larger candidate spaces. They will be introduced in the next subsection before introducing our contributions.

1.1 Related Works

In this paper, we aim to exploit collision information from CECA to optimize the key recovery of TA. 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 best candidates of each sub-key will be considered. Like τ_k in TA, we also set a reasonable threshold τ_d for the outputs of CECA, and only consider the τ_d best candidates of each collision value (i.e. XOR value $k_{j_1} \oplus k_{j_2}$ between two sub-keys k_{j_1} and k_{j_2}). A collision happens if a pair of candidates of sub-keys k_{j_1} and k_{j_2} and their collision value are within τ_k and τ_d simultaneously, and a chain includes at least a pair of collision.

Bogdanov et al. proposed Test of Chain (TC) in [BK12]. They introduced the best candidates within τ_d in CECA to the best candidates within τ_k in TA to construct specific collisions, and 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. Wang et al. proposed another CODCA named Fault Tolerant Chain (FTC) in [WWZ14], which found collisions between the first sub-key and the remaining sub-keys. Key enumeration still works in FTC, since the remaining candidates of $k_2 \sim k_{16}$ under a guessing k_1 are independent. TC and FTC are very simple and fast (see pros and cons in Table 1). Key verification assumes that the attacker knows plaintexts and the corresponding ciphertexts. It directly encrypts the known plaintexts using the chains in the collision space one by one after CODCAs. A chain is the key if the encryption generates the same ciphertexts. In this case, key verification in CODCAs is much faster than the traditional key enumeration.

There have a total number of 120 pairs of collisions among 16 sub-keys in AES-128, but TC and FTC only exploit 15 pairs resulting in poor utilization. They become too time-consuming when we consider very large τ_d and τ_k, since too many chains are established and there are too many collisions between them and candidates of the next sub-keys to detect. Unfortunately, this happens in CODCAs exploiting CECA to optimize TA, since the traditional CECA achieves performance much lower than TA. This is not surprising since the original intention of CECA is to attack flawed masks (e.g. DPA contest $v4.1$ [dpa]). Collision information is difficult to exploit in such a case. Therefore, we need to set a τ_d that is much larger than τ_k.

Group Collision Attack (GCA) in [OWSZ19] divided the full-key space of AES-128 in TA into 8 big “groups" containing 4 sub-keys and made full use of collisions from CECA to remove candidates that do not satisfy the number of collision required within each group. It continuously splices long chains from short chains. Specifically, it makes the two adjacent groups share two sub-keys. If their candidates are the same, a longer chain

can be stitched. GCA can exploit up to 32 pairs of collisions compared to 15 pairs in TC and FTC, thus having much stronger ability against large thresholds τ_k and τ_d when optimizing divide-and-conquer attacks. However, GCA still wastes the collisions between different groups (see Section 3.4 for details). Moreover, GCA does not introduce any fault tolerance strategy. Any one of the exploited 32 collision values falling outside the threshold τ_d will lead to failure of key recovery. Therefore, its success rate [SMY09] is relatively low. Finally, the repetitive detection of collisions between two groups also brings a lot of computation and significantly increases its runtime.

Table 1: Pros and cons of the existing CODCAs.

CODCAs	Advantages	Disadvantages
TC	a. very quick key verification, b. simple.	a. exploit only 15 collisions, b. no fault tolerant strategy, c. high complexity.
FTC	a. low complexity, b. facilitate key enumeration, c. fault tolerance strategy, d. simple.	a. exploit only 15 collisions.
GCA	a. exploit about 32 collisions, b. quick key verification.	a. no fault tolerant strategy, b. repetitive collision detection, c. high complexity.

It is worth mentioning that the purpose of this paper is not to improve the key enumeration or rank estimation techniques, which excludes CODCAs. In fact, any improvement of side-channel attacks will be reflected in the more advanced rank of the key. As we have mentioned above, CODCAs introduce collisions from CECA to optimize the key recovery of TA. Take AES-128 for an example, if a pair of candidates of the first and second sub-keys are deleted due to the absence of their candidate of collision value, 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 not be considered. In this case, CODCAs can quickly reduce the candidates of TA within τ_k to a much smaller remaining chain space. The challenge of CODCAs 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 a big challenge to construct such a combined attack.

1.2 Our Contributions

CODCAs are independent of the specific attacks to be optimized. In this paper, we exploit CECA to optimize the key recovery of TA, and aim to build a simple, high efficient CODCA, which exploits almost all collisions and leaves us the smallest candidates to verify. The main contributions of this paper are as follows:

(i) CECA's performance is significantly worse than TA. τ_d need to be set very large in this case. Otherwise, the collision information is difficult to be exploited. We introduce a Minkowski Distance enhanced Collision Attack (MDCA). It achieves different performance on different orders. Therefore, it can easily be applied to optimize different divide-and-conquer attacks in various attack scenarios, thus making the CODCAs more practical and meaningful.

(ii) FTC and GCA exploit only a small part of collisions and waste most of them. They leave a large number of full-key candidates and are time-consuming when tackling larger spaces. We propose a simple and efficient CODCA named Full-Collision Chain (FCC) 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 number of candidates to be guessed for each

sub-key under a fixed probability and optimize their search order, thus significantly alleviating collision detection load.

(iii) FCC 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 bundle several adjacent sub-keys together and exploit the Fault-Tolerant Vector (FTV) to significantly reduce its threshold. This significantly improves FCC's flexibility and performance.

1.3 Organization

The rest of this paper is organized as follows: Measurement setups, TA and Collision Attack (CA) including CECA are introduced in Section 2. Our MDCA, and the existing CODCAs like TC, FTC and GCA, are presented in Section 3. Our FCC and its optimization based on guessing theory are detailed in Section 4. The collision distribution and fault tolerance strategies 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 a Point-Of-Interest (POI) with high 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. Take AES-128 on our AT89S52 micro-controller for 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.

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

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:

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

Here $|\mathbf{m}_j|$ represents the number of POIs on template $\mathbf{m}_j$. We then normalize and rank the probabilities of each sub-key output by TA in descending order, and obtain $p_j = \left\{p_j^1, p_j^2, \ldots, p_j^{256}\right\}$ satisfying:

$$\sum_{i=1}^{256} p_j^i = 1 \tag{4}$$

as introduced in [CP17], where p_j^i denotes the i^{th} largest probability.

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:

$$\mathsf{Sbox}(x_{j_1} \oplus k_{j_1}) = \mathsf{Sbox}(x_{j_2} \oplus k_{j_2}), \tag{5}$$

and obtains a linear equation:

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

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

$$\delta_{j_1,j_2} = k_{j_1} \oplus k_{j_2} = x_{j_1} \oplus x_{j_2}, \tag{7}$$

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:

$$\phi\left(x_{j_1} \oplus k_{j_1}, x_{j_2} \oplus k_{j_2}\right) = \begin{cases} 1, & if\ \ \mathcal{D}\left(t_{j_1}, t_{j_2}\right) \le \tau_{po} \\ 0, & else \end{cases}. \tag{8}$$

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.

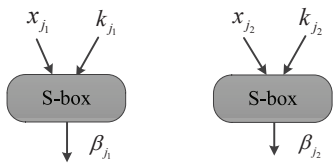

Figure 1: A linear collision between the j_1-th and the j_2-th S-boxes happens if they have the same outputs ($\beta_{j_1} = \beta_{j_2}$).

Correlation-Enhanced Collision Attack (CECA) [MME10] can be used to detect δ-s. Take the collision between the first and second S-boxes in the first round of AES-128 as an example, CECA divides their traces into 256 classes according to their plaintext bytes, calculates the mean power consumption vector $\mathbf{t}_j (j = 1, 2)$ of each class, and computes the correlation coefficient between them:

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

under a guessing $\delta_{1,2}$ (see Eq. 7). Here α from 0 to 255 is the index of 256 means in each class. Suppose that we have detected a total number of ν collision values, then a collision system is obtained:

$$\begin{cases} \delta_{j_1,j_2} = k_{j_1} \oplus k_{j_2}, \\ \delta_{j_3,j_4} = k_{j_3} \oplus k_{j_4}, \\ \qquad \cdots \\ \delta_{j_{2\nu-1},j_{2\nu}} = k_{j_{2\nu-1}} \oplus k_{j_{2\nu}}, \end{cases} \tag{9}$$

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).

3 Collision-Optimized Divide-and-Conquer Attacks

Traditional CECA's efficiency is much lower than TA's. If TA and CECA are directly performed on the POIs described in Section 2.1, their performance will still exhibit a very big gap. Moreover, the traditional CODCAs are not applicable against large τ_k and τ_d as explained in Section 1.1. Therefore, we will first introduce a profiled CECA named Minkowski Distance enhanced Collision Attack (MDCA) with performance close to TA in Section 3.1. Then, we give the examples of the existing CODCAs including Test of Chain (TC), Fault-Tolerant Chain (FTC) and Group Collision Attack (GCA) in the following Sections 3.2, 3.3 and 3.4, separately.

3.1 Minkowski Distance enhanced Collision Attack

Let n denote the number of power traces exploited in attacks, γ denote the order of Minkowski distance applied, and t_1^i $\left(t_2^i\right)$ denote the POIs of the first (second) S-box of the i^{th} power trace. We build a new scheme named Minkowski Distance enhanced Collision Attack (MDCA) as:

$$\sqrt[\gamma]{\sum_{i=1}^{n}\sum_{j=1}^{2}\left(t_j^i - \mathbf{m}_j^{\mathsf{Sbox}\left(x_j^i \oplus k_j\right)}\right)^{\gamma}}. \tag{10}$$

Obviously, the power consumption of POIs t_1^i and t_2^i are matched with profiled mean power consumption vectors $\mathbf{m}_1$ and $\mathbf{m}_2$ of S-boxes 1 and 2 under a pair of guessing sub-keys k_1 and k_2, and MDCA is also a profiled attack like TA.

We randomly extract 12800 traces to profile Hamming weight templates for TA, and 50 traces to profile template of each intermediate value in MDCA. The corresponding ranks of an experiment that TA and MDCA are performed on randomly selected 200 power traces are shown in Tables 2 and 3. The probabilities in Table 2 are normalized as explained in Section 2.2. Thresholds τ_d and τ_k here are set to 12 and 10. The correct sub-keys are 212, 153 and 17 as bolded, and their probabilities are also given in Table 2. The candidates of them are randomly combined, and the probability product of these

3 sub-keys is ranked at 31[2]. There are 7, 11 and 12 candidates for collisions $k_1 \leftrightarrow k_2$, $k_2 \leftrightarrow k_3$ and $k_1 \leftrightarrow k_3$, respectively (as shown in Table 4).

Table 2: The ranked candidates of 3 sub-keys within τ_k in TA.

	k_1		k_2		k_3	
rank	value	p	value	p	value	p
1	224	0.4273	**153**	0.6867	108	0.1724
2	178	0.2841	204	0.0717	**17**	0.1564
3	217	0.1565	80	0.0639	75	0.1100
4	**212**	0.0384	236	0.0309	160	0.1073
5	97	0.0150	244	0.0214	74	0.0554
6	222	0.0074	188	0.0210	117	0.0342
7	51	0.0068	180	0.0150	35	0.0182
8	249	0.0054	196	0.0139	213	0.0156
9	211	0.0050	12	0.0090	60	0.0146
10	221	0.0043	146	0.0066	12	0.0143
11	134	0.0040	24	0.0064	155	0.0141
12	244	0.0038	222	0.0047	229	0.0140

Table 3: The ranked candidates of δ-s within τ_d in MDCA.

$\delta_{1,2}$	$\delta_{1,3}$	$\delta_{2,3}$
77	197	136
73	236	161
12	193	201
28	132	0
29	148	224
69	149	2
221	77	192
15	173	165
65	232	225
93	79	128

Table 4: The collisions in combined attack TA and MDCA.

$k_1 \leftrightarrow k_2$		$k_2 \leftrightarrow k_3$		$k_1 \leftrightarrow k_3$	
224	236	**153**	**17**	224	117
217	196	153	60	224	12
212	**153**	153	155	**212**	**17**
97	188	204	12	212	60
249	188	236	108	212	155
249	180	236	12	97	160
211	146	188	117	97	229
		188	60	222	75
		180	17	222	74
		180	60	249	108
		12	12	249	17
				249	60

3.2 Test of 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. If we perform TC on the candidates of the 3 sub-keys with their δ-s given in Table 4, the remaining chains are shown in Table 5. It's noteworthy that TC fails to provide a practical scheme under very large thresholds τ_d and τ_k, since too many chains satisfy the collision conditions and the chain construction is too time-consuming in this case as explained in Section 1.1. Therefore, we do not consider it in the rest of this paper.

[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: TC Chains include $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$.

$k_1 \leftrightarrow k_2 \leftrightarrow k_3$			probability
224	236	108	0.4273
224	236	12	0.4273
212	153	155	0.0421
212	**153**	**17**	0.0388
212	153	60	0.0384
97	88	117	0.0156
97	88	60	0.0150
249	88	117	0.0061
249	180	17	0.0058
249	180	60	0.0054
249	88	60	0.0054

Table 6: FTC Chains include $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$.

$k_1 \leftrightarrow k_2 \leftrightarrow k_3$			probability
224	236	117	0.4279
224	236	12	0.4273
212	153	155	0.0421
212	**153**	**17**	0.0388
212	153	60	0.0384
97	188	160	0.0150
97	188	229	0.0150
249	180	17	0.0058
249	188	17	0.0058
249	180	108	0.0054
249	188	108	0.0054
249	180	60	0.0054
249	188	60	0.0054

3.3 Fault Tolerant Chain

Wang et al. provided the first practical CODCA against large τ_k and τ_d 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 a fault has occurred and enumerates this sub-key. The remaining chains that FTC performs on Tables 2 and 3 are shown in Table 6.

3.4 Group Collision Attack

Group Collision Attack(GCA) [OWSZ19] divides the 16 sub-keys of AES-128 into 8 big groups of equal size. Each group shares its first two sub-keys with its former group, and the last two sub-keys with its latter group. 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. For example, it exploits collisions $k_1 \leftrightarrow k_2$, $k_2 \leftrightarrow k_3$ and $k_1 \leftrightarrow k_3$ to build $k_1 \leftrightarrow k_2 \leftrightarrow k_3$ (as shown in Table 7). $k_2 \leftrightarrow k_3 \leftrightarrow k_4$ is constructed in the same way, and GCA obtains the first group $k_1 \leftrightarrow k_2 \leftrightarrow k_3 \leftrightarrow k_4$. It then builds the second group $k_3 \leftrightarrow k_4 \leftrightarrow k_5 \leftrightarrow k_6$. If $k_3 \leftrightarrow k_4$ of these first two groups are the same, then GCA obtains a longer chain $k_1 \leftrightarrow \cdots \leftrightarrow k_6$. CODCAs transform the original guessing space of divide-and-conquer attacks to much smaller collision chain space and make the key recovery easier. This is intuitively embodied in the more advanced key ranking (i.e. the rank of the correct chain $212 \leftrightarrow 153 \leftrightarrow 17$ drops from 31 in TA to 3).

4 Full-Collision Chains

GCA exploits information from up to 32 δ-s for its chain construction. It leaves us a much smaller chain space than TC and FTC. However, GCA also has its limitations as

Table 7: GCA Chains include $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$ and $k_2 \leftrightarrow k_3$.

$k_1 \leftrightarrow k_2 \leftrightarrow k_3$			probability
224	236	12	0.4273
212	153	155	0.0421
212	**153**	**17**	0.0388
212	153	60	0.0384
249	180	17	0.0058
249	180	60	0.0054
249	88	60	0.0054

mentioned in Section 1.1: (1) GCA still wastes the most of collisions between different groups, (2) it does not contain any fault tolerant strategy and success rate is very low, and (3) repetitive detection of collisions makes it very time-consuming. These disadvantages greatly limit its ability against huge candidate space. In this section, we propose a more efficient CODCA named Full-Collision Chain (FCC), which can quickly exploit the largest amount of collision information when constructing chains, and is suitable for very large τ_k and τ_d. Its collision detection mechanism will be introduced in Section 4.1. Moreover, an example is given in Section 4.2 to illustrate its performance compared to the existing CODCAs. Finally, we introduce guessing theory into FCC to make its threshold τ_k in TA flexible.

4.1 Collision Detection Mechanism in FCC

Unlike the classical collision detection function for single collision attacks defined in Eq. 8, it can be defined as:

$$\phi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right) = \begin{cases} 1, & if \ \ \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \left\{\phi_{j_1,j_2}^{1}, \ldots, \phi_{j_1,j_2}^{\tau_d}\right\} \\ 0, & else \end{cases}. \tag{11}$$

in CODCAs ($i_1 \leq \tau_k, i_2 \leq \tau_k$). Here ξ_j^i denotes the i^{th} best candidate of k_j in TA, and $\phi_{j_1,j_2}^{1}, \ldots, \phi_{j_1,j_2}^{\tau_d}$ are the τ_d best candidates of δ_{j_1,j_2}-s in MDCA. Single CECA including MDCA only determines δ-s, while CODCAs 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 $k_{j_1} \leftrightarrow k_{j_2}$. It includes candidates information of sub-keys k_{j_1} and k_{j_2}, and collision value $\delta_{j_1,j_2} = k_{j_1} \oplus k_{j_2}$.

Actually, the collision detection mechanism exploiting all collisions can be very simple. Considering the given thresholds τ_k and τ_d, if the length of the current q^{th} chain C_q (i.e. the number of sub-keys included) is n_l, then the number of collisions between it and the i^{th} best candidate ξ_j^i ($1 \leq i \leq \tau_k$) of k_j in TA is:

$$\Phi\left(\mathrm{C}_q, \xi_j^i\right) = \sum_{r=1}^{n_l} \phi\left(\mathrm{C}_q^r, \xi_j^i\right). \tag{12}$$

Here C_q^r denotes the candidate of the r^{th} sub-key on the chain, and the collision detection function $\phi\left(\mathrm{C}_q^r, \xi_j^i\right)$ satisfies Eq. 11. Let $\left[\mathrm{C}_q, \xi_j^i\right]$ denote the possible chain constituting of ξ_j^i and C_q, then the total number of collisions of the new chain can be updated by:

$$\Gamma\left(\left[\mathrm{C}_q, \xi_j^i\right]\right) = \Gamma\left(\mathrm{C}_q\right) + \Phi\left(\mathrm{C}_q^{n_l}, \xi_j^i\right). \tag{13}$$

Here $\Gamma\left(\mathrm{C}_q\right)$ denotes the number of collisions on the current chain C_q, and it satisfies:

$$\Gamma\left(\mathrm{C}_q\right) = \sum_{r_1=1}^{n_l-1} \sum_{r_2=r_1+1}^{n_l} \phi\left(\mathrm{C}_q^{r_1}, \mathrm{C}_q^{r_2}\right). \tag{14}$$

Following the CODCAs like TC and FTC, we name this CODCA exploiting all collision information as Full-Collision Chain (FCC). FCC exploits the largest number of collisions, thus leaving 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 fall within them.

4.2 Typical Application Scenarios

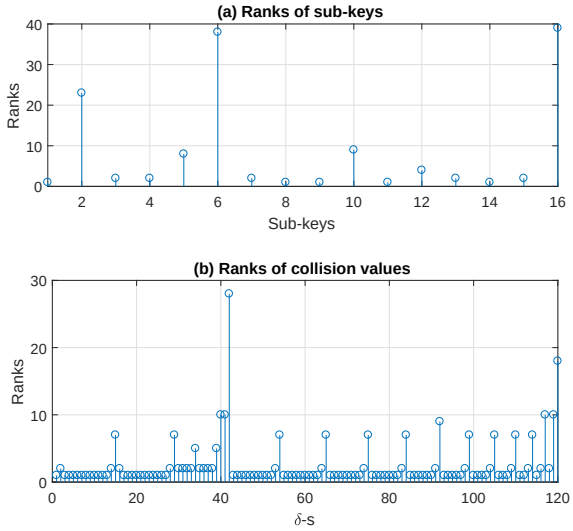

Figure 2: The ranks of sub-keys (a) and correct δ-s (b).

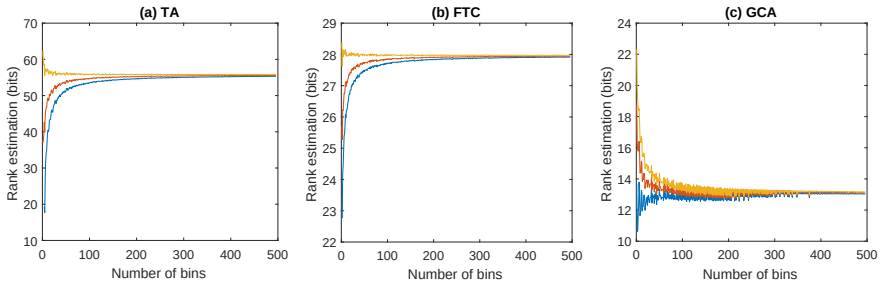

Figure 3: Key rank estimation on the original space of TA, and the remaining chain space of FTC and GCA.

We aim to conquer the key ranked at very deep space in this paper. In other words, we need to set both τ_d and τ_k very large. We perform TA and the 5^{th}-order MDCA ($\gamma = 5$) on the power traces extracted in Section 3. We rank the candidates from the best one to the worst one for each sub-key and each δ, and the experimental results are shown in Fig. 2. For example, the worst ranked sub-key is k_{16} and its rank is 39 in Fig. 2(a), and the worst ranked δ is the one between k_3 and k_{16}, and its rank is 28 in Fig. 2(b). Therefore, to ensure that FCC can recover the key, the thresholds τ_k and τ_d should be set to at least 39 and 28 respectively. In other words, we need to find the correct chain from such two huge spaces in CODCAs.

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" on CODCAs is different from that on TA. 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 still be performed on them under each candidate of k_1, and the rank of the key can be estimated by the sum of these estimations. The 16 sub-keys are divided into 8 big "groups" in GCA as introduced in Section 3.4. To facilitate the key recovery, we exploit the remaining candidates of 4 non-overlapping groups and ignore their inter-group collisions, thus making them independent. Probability products of each group are computed, and classical histogram based key rank estimation is performed on them.

We set τ_k to 40 and τ_d to 40, thus all sub-keys and δ-s are within such huge 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 FCC, their probability products can be ranked directly. Traditional key enumeration, FTC and GCA optimize the computing power from unconquerable huge space $\tau_k^{16} = 40^{16} = 2^{16 \cdot \log_2 40} = 2^{85.1508}$ within threshold $\tau_k = 40$ to $2^{55.7387}$, $2^{27.9627}$ and $2^{13.0338}$. The time consumption of GCA, FTC and FCC is about 4.040, 0.4070 and 3.3360 seconds, which vividly indicates the superiority of CODCAs. It's noteworthy that the time consumption of key recovery in CODCAs includes the runtime of CODCAs themselves and the latter key verification. Key verification verifies each full-key candidate by encrypting the known plaintexts, and checking whether the ciphertexts are the same as the ones encrypted by the correct key. FTC and GCA reduce the difficulty of key recovery, but the following "key enumeration" (including key verification), which enumerates the probabilities of full-key candidates from the largest to the smallest, is still time-consuming. FCC exhibits its capability in this case, and leaves us the smallest space when both τ_k and τ_d are reasonably set.

The number of remaining chains under the thresholds near $\tau_k = 40$ and $\tau_d = 40$ is shown in Fig. 4. The number of remaining chains is almost the same under several close thresholds τ_d and τ_k. This fully illustrates the superiority of FCC where we do not need to adjust the thresholds deliberately, as small changes in them will not significantly affect the computation overhead.

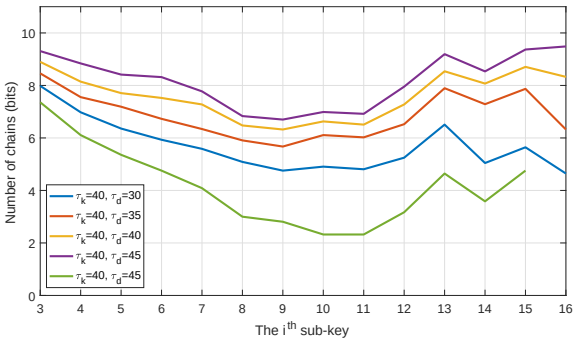

Figure 4: The number of FCC chains under each sub-key.

4.3 Guessing Theory Optimization

The existing CODCAs set a fixed τ_k for all sub-keys. On one hand, the locations of some sub-keys are 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.

A metric considers only the candidates that satisfy the given probability value $\alpha \in (0, 1)$:

$$\mu_\alpha\left(\mathbb{K}_j\right) = \min\left\{\tau_k \Big| \sum_{i=1}^{\tau_k} p_j^i \geq \alpha\right\}, \tag{15}$$

which is named as α-*work-factor* in [Pli00]. Here τ_k is the guessed number of candidates of k_j, p_j^i is the probability of the i-th best candidate of k_j as introduced in Section 2.2, and $\mathbb{K}_j = \{0, 1, \ldots, 255\}$ is its possible candidates. Compared to set 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:

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

under the given thresholds can be estimated quickly. For example, when 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 number of guesses $\mu_\alpha\left(\mathbb{K}_j\right)$ $(1 \leq j \leq 16)$ to lower computational complexity. The average number of candidates of the re-ranked sub-keys under different number of power traces (denoted as n) with 200 repetitions is shown in Fig. 5.

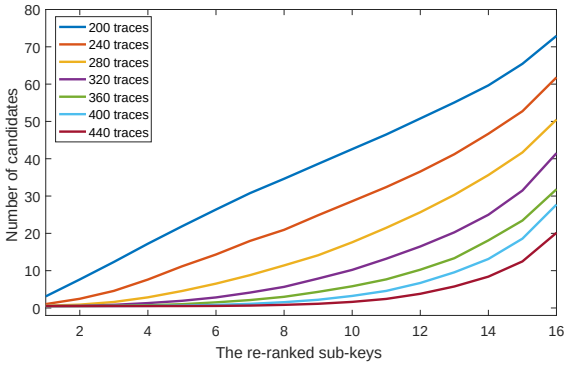

Figure 5: The number of candidates of the re-ranked sub-keys under $\alpha = 0.996$.

The sub-keys of TA are re-ranked before recovery according to their number of candidates provided by guessing theory, which makes FCC more efficient. On one hand, although all collisions of the sub-keys re-ranked in the front is fully utilized by our FCC, the candidates that can be excluded are limited, but their number of candidates 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 FCC work faster. On the other hand, there are a lot of candidates for sub-keys re-ranked backward. FCC also requires more collisions on them, so that the number of remaining chains is not too large. This significantly reduces the runtime of FCC, and leaves us a small chain space, which facilitates the future key recovery.

5 Fault Tolerance

FCC can efficiently reduce the candidates within threshold τ_k by maximizing collision exploitation. Therefore, the thresholds τ_k and τ_d can be set quite large under FCC, 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, FCC requires all δ-s to fall within τ_d. There may be a few δ-s ranked much deeper than we have discussed in Section 4.2, which may make it too time-consuming or impossible for FCC to conquer the key. In this case, it requires fault tolerance on FCC 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 will first discuss the distribution of δ-s beyond τ_d in Section 5.1. In principle, to allow a fixed number of δ-s beyond τ_d, rotative fault tolerance is required. We will give a second-best strategy to bundle several adjacent sub-keys together as big "blocks" for fault tolerance to avoid this in Section 5.2. Finally, as it's difficult to set the number of δ-s beyond τ_d for each block, we exploit additional constraints and provide a good reference in Section 5.3.

5.1 Distribution of Collision Values Beyond τ_d

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 within different τ_d ranges are shown in Table 8. There are only 1, 1 and 4 δ-s for k_{14}, k_{15}, and k_{16} beyond $\tau_d = 10$, respectively. Only 2 δ-s should be faultily tolerated if τ_d reaches 15. In fact, this can be analyzed from Signal-to-Noise Ratio(SNR) of the selected power traces and MDCA 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.

Table 8: δ-s within different τ_d ranges.

Ranges	Collisions
$10 \leq \tau_d < 15$	$k_3 \leftrightarrow k_{14}$, $k_3 \leftrightarrow k_{15}$, $k_{13} \leftrightarrow k_{16}$, $k_{14} \leftrightarrow k_{16}$
$15 \leq \tau_d < 20$	$k_{15} \leftrightarrow k_{16}$
$20 \leq \tau_d < 25$	
$25 \leq \tau_d < 30$	$k_3 \leftrightarrow k_{16}$

We randomly select 320 out of the 50,000 power traces to perform MDCA (1000 repetitions), count the number of δ-s beyond τ_d from 10 to 60, 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. To achieve a success rate 0.80, we need to perform fault tolerance on δ-s of more than average 7 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 almost smaller than 30, 20, 15, 15, 15, 14 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. It's noteworthy that we give the above phenomena only to better illustrate the basis of our fault tolerance strategy. In fact, our fault tolerance strategy is very flexible, it can be dynamically adjusted according to the specific attack situations, the knowledge of collision distribution is not required (see Sections 5.2 and 5.3 for details).

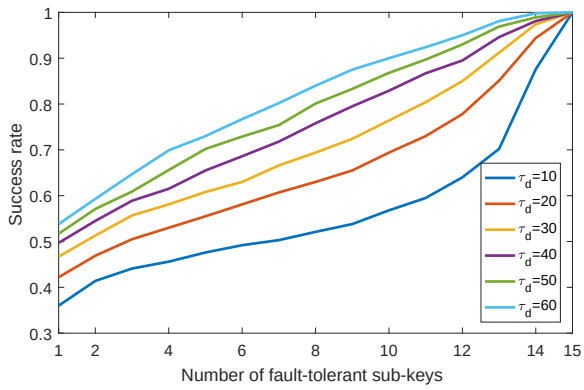

Figure 6: Success probabilities if δ-s beyond τ_d can be fully concentrated on the given number of sub-keys for their fault tolerance.

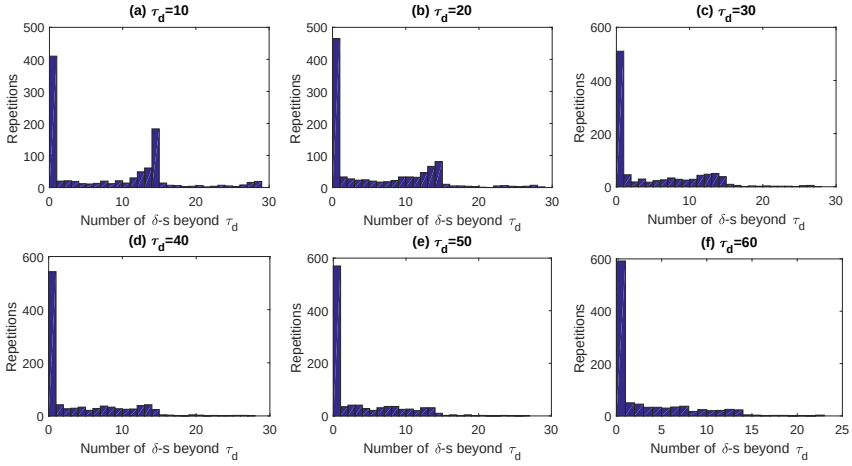

Figure 7: Distributions of δ-s beyond threshold τ_d when $n = 320$.

5.2 Fault Tolerance Strategy

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 0.60 (0.75) when $\tau_d = 50$ (60). 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. In fact, δ-s beyond τ_d are limited and scattered on different sub-keys. In this case, only a small number of them happen on several adjacent sub-keys and are convenient for fault-tolerance. Here we give a fault tolerance strategy for our FCC, in which several sub-keys are “bundled" together for fault tolerance, so that each large “block" allows a small number of δ-s beyond τ_d without strictly limiting the specific sub-key of faults, so as to achieve good fault-tolerant performance.

We only consider the case that each block includes two sub-keys in this paper. Let $k_j^{'}$ denote the sub-key with the j^{th} smallest number of candidates within τ_k after guessing theory optimization. Here we exploit ξ_j^i to denote the i-th best candidate of $k_j^{'}$ as explained in Section 4.1. For a chain $212 \leftrightarrow 24 \leftrightarrow \cdots \leftrightarrow 153$ including candidates of a total of j sub-keys 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 $\left\{k_{j+1}^{'}, k_{j+2}^{'}\right\}$ is:

$$n_t = \sum_{\kappa=j+1}^{j+2} \sum_{r=1}^{n_l} \phi\left(\mathrm{C}_q^r, \xi_\kappa^{i_\kappa}\right) + \phi\left(\xi_{j+1}^{i_{j+1}}, \xi_{j+2}^{i_{j+2}}\right). \tag{17}$$

Here $n_l = j$, $1 \le i_j \le \mu_\alpha\left(\mathbb{K}_j^{'}\right)$ 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 6 δ-s to be beyond τ_d on this block, this covers all cases where at most 6 δ-s for all sub-keys in the block are out of τ_d. As mentioned before, 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 FCC with fault tolerance does not need to be very large. Moreover, the optimized FCC avoids rotative fault tolerance and effectively reduces the complexity of chain construction, which can be verified through the experiments in Section 6.

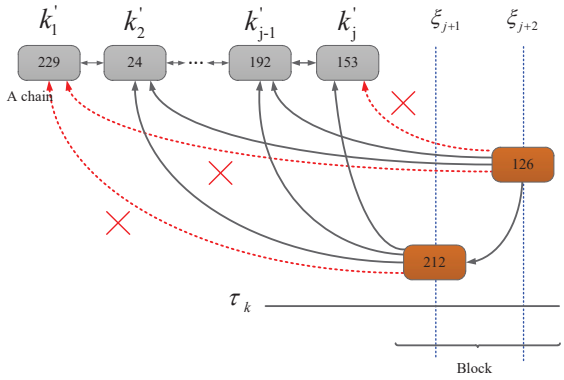

Figure 8: FCC performs fault tolerance on a block with 3 δ-s beyond τ_d.

5.3 Constraints

It is not complex to set a good fault-tolerant threshold for each big "block" according to computing power. The j^{th} re-ranked sub-key $k_j^{'}$ has a total of $j-1$ δ-s in our FCC (see Section 4.1). Therefore, we can allow different number 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 0, 1, 2, 3, 4, 5, 6 and 7 δ-s to be beyond τ_d for the 1^{st}, 2^{nd}, 3^{rd}, 4^{th}, 5^{th}, 6^{th}, 7^{th} and 8^{th} blocks independently, then we get a FTV as $(0, 1, 2, 3, 4, 5, 6, 7)$. 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.

An important advantage of FCC with fault tolerance is that it enables us to decide whether a block should be "fat" or "thin" according to our computing power. To make

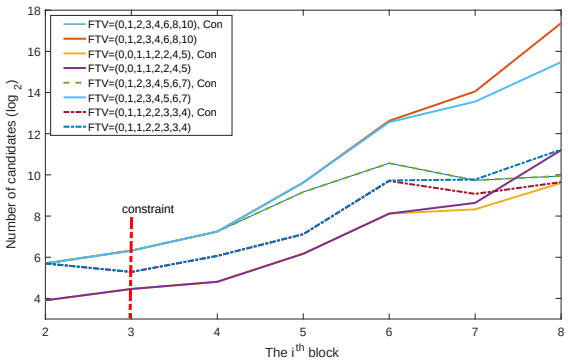

Figure 9: The number of chains under different FTV-s with and without constraints.

a block "thin" (i.e. fewer chains when considering the block), we just need to set the corresponding fault-tolerant threshold smaller in FTV. This makes our FCC flexible and easy to be extended to different attack scenarios. Four FTVs in Fig. 9 are selected to illustrate the adjustment and their runtime is given in Table 9. The total number of chains under vector $(0, 1, 1, 2, 2, 3, 3, 4)$ increases slowly and FCC consumes very little time for each block. The number of chains under $(0, 0, 1, 2, 4, 6, 8, 10)$ is larger than that of $(0, 1, 2, 3, 4, 5, 6, 7)$, since the former uses larger thresholds. However, it also consumes more time.

Table 9: Mean time consumption of 4 FTV-s with or without constraints under $\tau_d = 10$ (seconds).

FTV	with constraints	without constraints
$(0, 0, 1, 1, 2, 2, 4, 5)$	4.72	5.16
$(0, 1, 1, 2, 2, 3, 3, 4)$	10.05	12.19
$(0, 1, 2, 3, 4, 5, 6, 7)$	20.63	139.81
$(0, 1, 2, 3, 4, 6, 8, 10)$	20.72	536.56

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 τ_c δ-s fewer than the chain with the largest number of collisions is allowed. 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 $\left[k'_{j+1}, k'_{j+2}\right]$ with the largest number of collisions satisfy:

$$n_{t_{max}} = \max\left\{\Gamma\left(\mathrm{C}_q\right) + \sum_{\kappa=j+1}^{j+2} \Phi\left(\mathrm{C}_q, \xi_{\kappa}^{i_\kappa}\right)\right\}. \tag{18}$$

Here $i_\kappa = 1, 2, \ldots, \mu_\alpha\left(\mathbb{K}'_\kappa\right)$. The maintained chains should satisfy:

$$n_t \geq n_{t_{max}} - \tau_c. \tag{19}$$

The corresponding experimental results under $\tau_c = 6$ have been shown in Fig. 9 (labeled by "**Con**") and Table 9. The number of chains can be reduced from thousands to hundreds,

thus significantly improving efficiency. Actually, our experimental results show that the success rate under the constraint $\tau_c = 6$ 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

The key recovery in CODCAs includes two stages as we have explained in Section 4.2. The CODCAs post-process the candidates of optimized divide-and-conquer attack and the collision attack, thus achieving the remaining chain space in the first stage. The chains are ordered according to certain rules in the second stage, and verified to achieve the final key recovery. Therefore, their performance is not only reflected in the success rate [SMY09], but also in the number of chains to be verified in the second stage. The candidate space can be significantly reduced after second 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. GCA and FTC often remain a large number of candidates, which may be not exhaustible. Therefore, we exploit histogram based key rank estimation after them as introduced in Section 4.2.

We will first compare the performance of TA and classic CCA with our MDCA to illustrate its superiority in Section 6.1. Then, we compare the performance of FTC, GCA, FCC (with and without fault tolerance) under different thresholds τ_d and different number of traces n in Sections 6.2 and 6.3, respectively. As we have explained in Section 5.3, our FCC with fault tolerance performs very well under very large FTV. Therefore, we set a very large FTV $(0, 1, 2, 3, 4, 6, 8, 10)$ and only consider the chains with at most $\tau_c = 6$ collisions fewer than the chain with the largest number of collisions. Here we only consider key recovery from theoretical success probability $\alpha^{16} = 0.996^{16} = 0.9379$. The evaluations under large thresholds τ_d and different number of traces are very time-consuming, we only repeat each experiment 200 times.

6.1 Performance of MDCA

MDCA is not a kind of CECA, and its performance is much closer to TA compared to CECA. Its performance under different orders evaluated by guessing entropy [SMY09] is shown in Fig. 10. This makes the CODCAs built from TA and MDCA more meaningful and practical. MDCA achieves better performance when $\gamma \leq 4$ (see Fig. 10). However, the performance is very close when the order of MDCA γ is 2, 3 and 4. Finally, it decreases in the 5^{th} and 6^{th} orders. It's noteworthy that we do not need to balance the two attacks in CODCAs strictly. Therefore, we exploit 5^{th}-order MDCA in this paper.

We use 1^{st}-, 10^{th}- and 20^{th}-order Success Rate [SMY09] to evaluate the performance of TA, CECA and our MDCA (5^{th}-order) under different number of traces, and the experimental results under 500 repetitions are shown in Fig. 11. TA's first-order success rate is even significantly higher than CECA's 20^{th}-order success rate. The success rates of 5^{th}-order MDCA and TA are about 1.00 under 600 traces. The success rate of CECA reaches about 1.00 under 2000 traces. The performance of our 5^{th}-order MDCA is very close to that of TA, and their combination will be more challenging, meaningful and practical in more attack scenarios. We try to exploit the CODCAs to achieve significantly higher success rate than performing them separately under the given computing power.

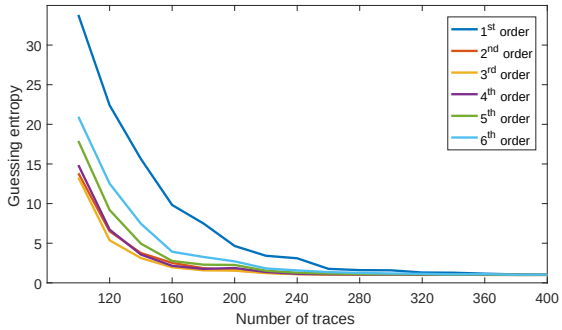

Figure 10: The guessing entropy of different orders of MDCA under different number of traces.

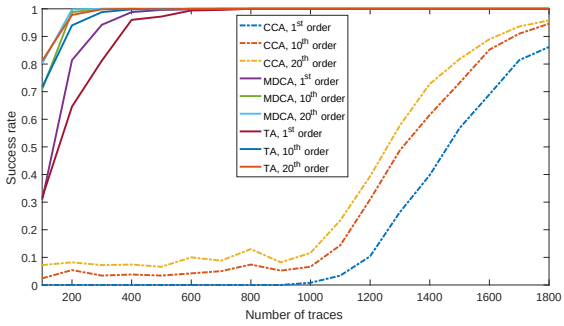

Figure 11: The success rates of TA, CECA and our 5^{th}-order MDCA.

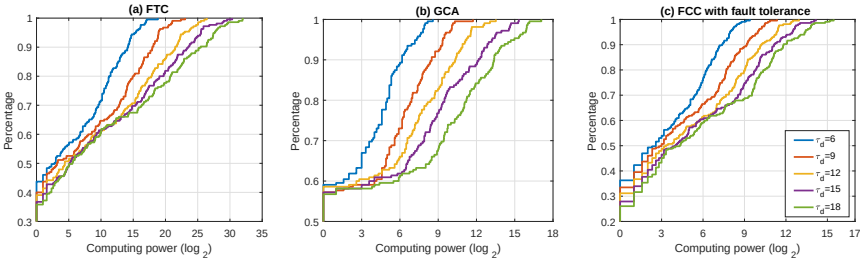

Figure 12: Number of chains to be verified in key recovery under different thresholds τ_d from 6 to 18.

6.2 Experiments on Different Thresholds τ_d

The success rate under different thresholds τ_d when $n = 320$ is shown in Table 10, and the corresponding number of remaining chains to be verified is shown in Fig. 12. The growth of τ_d makes more δ-s fall within it. This makes the collision conditions easier to be satisfied and more chains are constructed. However, the success rate of FCC without optimization under $\tau_d = 18$ is only 0.350, not even comparable to that of FCC with fault tolerance under $\tau_d = 6$ in Table 10. This indicates that the fault tolerance strategy significantly improves the performance of FCC. Table 11 also shows that the runtime of FTC and FCC is shorter than GCA and FCC with fault tolerance. GCA spends most of its runtime on

splicing, while FCC spends most of its runtime on fault-tolerance. Since τ_d from 6 to 18 is too small and there is no chain satisfying the collision conditions after first several blocks, FCC terminates quickly (see Table 11). FTC achieves success rate very close to the FCC with fault tolerance. However, Its remaining chains to be verified are many more than FCC's in Fig. 12. This situation will get worse when we further reduce the power traces to fewer than 280. FTC and GCA without enumeration may be infeasible in this case, since too many chains to splice before key verification.

Table 10: Success rates under different thresholds τ_d.

τ_d	6	9	12
FTC	0.420	0.455	0.475
GCA	0.335	0.360	0.365
FCC(original)	0.270	0.305	0.320
FCC(fault tolerance)	0.400	0.450	0.490
τ_d	15	18	
FTC	0.510	0.515	
GCA	0.400	0.410	
FCC(original)	0.315	0.350	
FCC(fault tolerance)	0.520	0.535	

Table 11: Time consumption (seconds) under different thresholds τ_d.

τ_d	6	9	12
FTC	0.0087	0.0086	0.0088
GCA	0.0322	0.0405	0.0582
FCC(original)	0.0542	0.0716	0.0925
FCC(fault tolerance)	0.12426	3.8847	11.8038
τ_d	15	18	
FTC	0.0086	0.0089	
GCA	0.0854	0.1233	
FCC(original)	0.1261	0.1614	
FCC(fault tolerance)	50.4653	251.7807	

The selected FTV and τ_d affect the experimental results, increasing them can improve the success rate of FCC. However, this increases the number of remaining chains and brings a lot of extra computation (see Table 11). Since larger τ_d makes more δ-s satisfy the given conditions while the corresponding FTV $(0, 1, 2, 3, 4, 6, 8, 10)$ is not adjusted accordingly, too many chains with fewer collisions are formed as shown in Fig. 13. Here we add the constraint that a chain has almost 6 δ-s fewer than the chain with the largest number of collisions is allowed after the third block as explained in Section 5.3. The choice of τ_d and FTV reflects the tradeoff between fault-tolerant time and "enumeration" time. If they are larger, FCC (with fault tolerance) 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 $n = 320$ in Fig. 12, the number of remaining chains is still large. This is especially for FTC, the space is close to 2^{32}. It will take about one day to enumerate 2^{40} of TA on our desktop computer. However, our FT-FCC can recover the key quickly from the remaining space that is smaller than 2^{15}. We can draw a conclusion from Fig. 12(c) that further increasing it will not significantly improve the success rate when τ_d increases to a certain height.

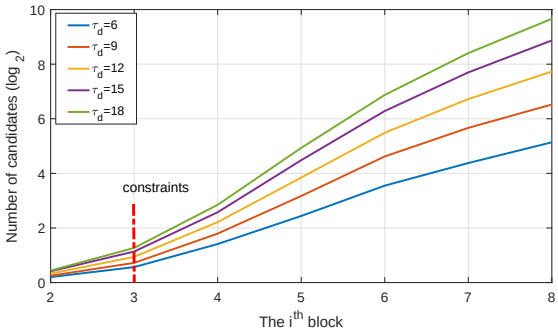

Figure 13: The number of FCC (with fault tolerance) chains under different τ_d-s.

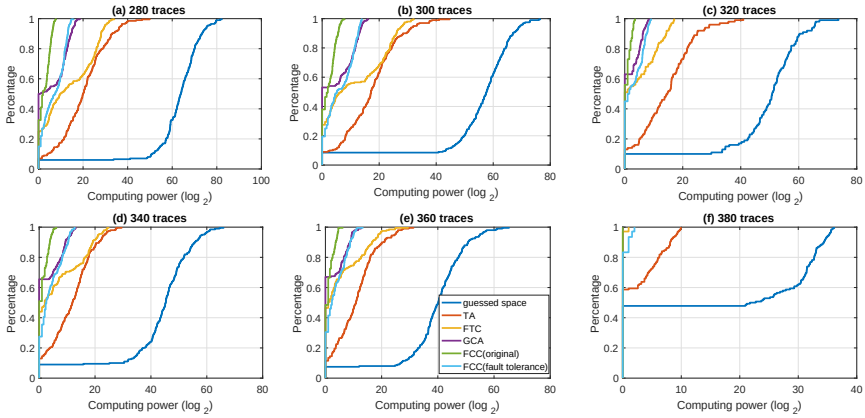

Figure 14: Number of chains to be verified in key recovery under different number of traces.

6.3 Experiments on Different Number of Traces

We also compare the performance of FTC, GCA and our FCC (with and without fault tolerance) under different number of traces n. We set τ_d to 15 and FTV to $(0, 1, 2, 3, 4, 6, 8, 10)$. Compared to the growth of threshold τ_d, using more traces makes the sub-keys and the corresponding δ-s fall within the thresholds more quickly. The success rates of different schemes under $280 \sim 380$ traces are shown in Table 12. With more power traces being exploited, the rank of key is more advanced, and the threshold τ_k of each sub-key becomes shorter under the same probability $\alpha = 0.996$, thus resulting in a rapid reduction in candidate space and runtime (see Fig. 14 and Table 13). The success rate of GCA grows slowly and is lower than FTC and our FCC with fault tolerance. It is worth noting that although it's a very strict requirement for all δ-s to be within τ_d, this condition becomes easier to satisfy by increasing n, resulting in a significantly higher success rate of FCC.

The success rate of our FCC increases slowly when the threshold τ_d reaches a certain height, and the increase in the number of power traces will also have similar effect. It can also be seen from Fig. 14 that the success rate of our FCC with fault tolerance increases slowly from 0.465 to 0.640 when $n \geq 280$, which is close to the success rate of FTC, and significantly higher than that of GCA. Moreover, it achieves a remaining chain space that is much smaller than FTC as shown in Fig. 14, which 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, the number of chains in our FCC does not grow explosively under reasonably restrictive conditions as shown in Fig. 15. Therefore, we can further adjust FTV, such as from $(0, 1, 2, 3, 4, 6, 8, 10)$ to $(1, 2, 4, 6, 8, 10, 12)$, or enlarge the parameter τ_d given in Section 5.3, which further improves the success rate without incurring too much computation. Similarly, if it is set too large and the runtime is too long, we can reduce it appropriately.

Table 12: Success rates under different number of traces.

Number of traces	280	300	320
FTC	0.505	0.490	0.510
GCA	0.370	0.385	0.400
FCC(original)	0.295	0.285	0.320
FCC(fault tolerance)	0.465	0.485	0.520
Number of traces	340	360	380
FTC	0.585	0.605	0.670
GCA	0.450	0.480	0.565
FCC(original)	0.385	0.440	0.500
FCC(fault tolerance)	0.570	0.590	0.640

Table 13: Time consumption (seconds) under different number of traces.

Number of traces	280	300	320
FTC	0.0365	0.0125	0.0086
GCA	0.1911	0.1233	0.0854
FCC(original)	0.2520	0.1575	0.1261
FCC(fault tolerance)	318.7231	97.6415	60.4653
Number of traces	340	360	380
FTC	0.0057	0.0103	0.0041
GCA	0.0506	0.0306	0.0258
FCC(original)	0.0821	0.0281	0.0474
FCC(fault tolerance)	13.9931	2.8876	1.864

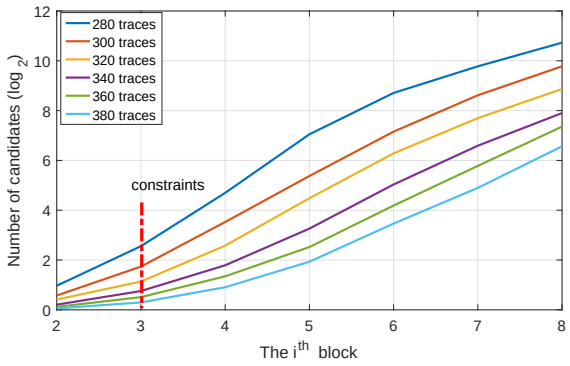

Figure 15: The number of FCC chains (with fault tolerance) under different number of traces.

7 Conclusions

The existing CODCAs exploit collision information to optimize the key recovery of divide-and-conquer attacks. They transform the huge candidate space into a much smaller collision space, wherein only the chains satisfying the collision conditions are considered. 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 ranked at much deeper spaces.

In this paper, we propose a simple yet efficient CODCA named FCC, which is able to exploit all collision information and further extends the reachable space. We further introduce guessing theory to optimize the search order of sub-keys according to their number of candidates within threshold. Benefiting from the high efficiency of its collision detection and stronger exclusion capability of candidates within threshold that do not satisfy collision conditions, FCC can search much larger candidate spaces than FTC and GCA. The collision values beyond the threshold in collision attack can be concentrated on several sub-keys for their fault-tolerance. We introduce fault tolerance strategy to reduce the threshold, thus making FCC more practical. Hence, our FCC provides new ways to further push the limits of reachable key space. The experimental results also demonstrate their superiority. Due to the flexibility of MDCA and fault tolerance mechanisms exploited, FCC is also feasible to be applied to other attack scenarios. This paper only considers fault-tolerance on collision values, and leaves the optimization of FCC and fault-tolerance on sub-keys as open problems. Every scheme has its advantages and disadvantages. MDCA achieves performance close to TA's. However, the candidates within threshold in TA will often have collision values within threshold in MDCA. This will lower the fault tolerance performance of FCC. A better MDCA is also our future work.

References

[BCO04] Eric Brier, Christophe Clavier, and Francis 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.

[BK12] Andrey Bogdanov and Ilya Kizhvatov. Beyond the limits of DPA: combined side-channel collision attacks. *IEEE Trans. Computers*, 61(8):1153–1164, 2012.

[CP17] Marios 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.

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

[dpa] Dpa contest. http://www.dpacontest.org/home/.

[GGP+15] Cezary Glowacz, Vincent Grosso, Romain Poussier, Joachim Schüth, and François-Xavier 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.

[GPPT16] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran 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.

[Gro18] Vincent 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.

[GS13] Benoît Gérard and François-Xavier 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. https://doi.org/10.1007/s13389-013-0051-9.

[GST14] Daniel Genkin, Adi Shamir, and Eran 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.

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

[LMV04] Hervé Ledig, Frédéric Muller, and Frédéric 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.

[LWWW17] Yang Li, Shuang Wang, Zhibin Wang, and Jian 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.

[MME10] Amir Moradi, Oliver Mischke, and Thomas 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.

[OWSZ19] Changhai Ou, Zhu Wang, Degang Sun, and Xinping Zhou. Group collision attack. *IEEE Trans. Information Forensics and Security*, 14(4):939–953, 2019.

[Pli00] John 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.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 61–81, 2016.

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof 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.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti 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.

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier 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.

[WWZ14] Danhui Wang, An Wang, and Xuexin 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.

[WYS+18] Weijia Wang, Yu Yu, François-Xavier Standaert, Junrong Liu, Zheng Guo, and Dawu Gu. Ridge-based DPA: improvement of differential power analysis for nanoscale chips. *IEEE Trans. Information Forensics and Security*, 13(5):1301–1316, 2018.

[YGH16] Yuval Yarom, Daniel Genkin, and Nadia 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.