Full Collision Attack: Pushing the Limits of Exhaustible Key Spaces

Changhai Ou and Siew-Kei Lam

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

Abstract. Recovering keys efficiently from far beyond exhaustible candidate spaces is a meaningful but very challenging topic in Side-Channel Attacks (SCA). Recent methods often utilize collision optimizations to reduce the key candidate space so that exhaustive search methods can be feasibly applied for key recovery. However, the current collision optimization methods can only utilize information of a small number of collisions, which limits the number of wrong key candidates that can be removed. In addition, their application is restricted to situations where only small thresholds can be applied. As such, the existing methods are not feasible for recovering the full key if sub-keys and collision values are located in much deeper spaces as we will discuss in this paper. To overcome these problems, we propose Full Collision Attack (FCA). Compared to the existing methods, FCA makes use of all possible collisions between any two sub-keys and removes a larger number of wrong key candidates, thus enabling key recovery in much deeper spaces. Moreover, we find that the collision values that fall beyond the threshold usually occurs only for a few sub-keys. Based on this finding, we propose the Rotational Error Tolerant FCA (RET-FCA) to significantly reduce the candidate space of collisions. Our results show that RET-FCA performs favourably when the collision values fall in the intractable space of FCA.

Keywords: FCA, full collision, group collision attack, divide and conquer, brute-force, side-channel attack.

1 Introduction

Implementations of cryptographic algorithms on devices produce unintentional leakages from various channels such as time [13, 32], power consumption [14, 31, 2], electromagnetic [10, 9], cache patterns [18, 7], acoustic [11], etc. These side-channel information can be statistically analyzed for key recovery, which pose a serious threat to the security of cryptographic devices. Power consumption is one of the most widely used channels in Side-Channel Attack (SCA), and a large number of distinguishers have been constructed based on it. Divide and conquer attacks, such as Differential Power Analysis (DPA) [14], Correlation Power Analysis (CPA) [4], Template Attack (TA) [5] and Mutual Information Analysis (MIA) [12, 29], divide the full key into several sub-keys and recover

them one at a time. Based on the premise that the attacker captures enough leaky information, the correct sub-keys can be distinguished from the wrong ones by distinguishers. However, if the attacker does not obtain sufficient power traces, it's possible that one or several sub-keys are not ranked on but somewhere close to the most possible one of their corresponding guessing sequences. In such scenarios, the attacker needs to optimize the exhaustive search schemes in order to recover keys.

Key enumeration [24, 6, 16, 28] is a very hot topic in recent years. It computes the probabilities of full key candidates from the combination of sub-keys, then enumerates them from the most possible one to the least possible one. By exploiting leaky information, each correct sub-key is ranked as one of the most likely candidates of a distinguisher. Therefore, key enumeration is essentially an optimized method for exhaustive search. Another optimization method is to use collision attacks [26, 15] for constructing efficient distinguishers and deleting wrong key candidates that fall within a threshold. Since only a few candidates that satisfy the given collision conditions are retained, the key can be recovered more quickly from the smaller candidate space. Ou et al. in [22] proved that collision optimizations could recover the key quickly from huge search spaces (e.g. 2^{80}) if the collision satisfied given conditions. This paper enhances collision optimization to recover the key from a much larger key space (e.g. larger than 2^{96}), which is infeasible for the existing works. Related works will be given in the next subsection before introducing our contributions.

1.1 Related Works

The idea of using collisions to optimize exhaustive key search originated from the Test of Chain (TC) introduced in in [3]. Let k_i denote the i-th sub-key and k denote the threshold of each sub-key, which means only the k most likely key guesses are considered. TC sets a reasonable threshold for the outputs of a distinguisher, and then uses collision attack to find long chains from the first sub-key to the last sub-key, thus eliminating key candidates outside these chains. The key recovery fails if at least one collision of the correct sub-keys exceeds the threshold. This requires the attacker to either capture more leakages or set a larger threshold. However, no practical implementation scheme was given in [3]. Wang et al. constructed the first feasible scheme called Fault Tolerant Chain (FTC) in [30]. Instead of finding a long chain containing all the sub-keys, FTC finds the collisions between the first sub-key and the remaining sub-keys. Specifically, FTC uses Correlation enhanced Collision Attack (CCA) [20] to rank and guess the collision values between sub-keys. These values are then introduced into CPA to construct possible sub-key combinations (i.e. collisions). If a sub-key does not collide with the first sub-key, the attacker enumerates it from the most possible one to the least possible one. It is worth mentioning that these sub-keys are enumerated independently, rather than using key enumeration to enumerate them simultaneously. If multiple sub-keys are error and deleted, it is difficult for FTC to recover the full key since there are too many possible candidates remaining (see Section 2.3).

It is noteworthy that TC and FTC only consider the case where the threshold d of collision attack equals 1, which means only the most possible collision value is considered. In this case, CCA should be more powerful than CPA, thus most of collision values among sub-keys are the most possible ones. This enlarges the probability that collisions occur and makes the collision optimized exhaustion meaningful. However, if the performance of two distinguishers is notably different, the full key can be recovered more efficiently with the one of higher performance. In fact, CCA's performance is worse than CPA's. This is not surprising since the original intention of CCA is to attack flawed masks (e.g. DPA contest $v4.1$ [1]). Group Verification Chain (GVC) [23] considers $\mathsf{k} > 1$ and $\mathsf{d} > 1$, and uses the collisions between the current sub-key and other sub-keys to verify and rank its possible values within the threshold. In this case, a sub-key is verified by other sub-keys. The more collisions, the higher probability the guessing value to be the correct sub-key. Therefore, Ou et al. in [22] classified GVC as key re-ranking, which could not be used to reduce the key space.

Combining the characteristics of collision and verification, GCA in [22] considers two distinguishers with similar performance for the first time, which makes GCA more realistic and meaningful. Specifically, GCA divides the full key space into several groups containing the same number of sub-keys and makes full use of collision information to remove wrong candidates within each group. These sub-keys do not overlap, i.e., the same sub-key appears only once in all groups. It then uses collisions to construct verification chains, which establish the relationship between groups and avoid exponential growth in the number of recombined possible full keys. GCA uses the thresholds of two distinguishers (e.g. k of CPA and d of CCA) to evaluate its search ability. It can utilize collision information several times more than TC and FTC, and has much higher search ability than TC and FTC. However, there are still a lot of unused collisions among groups under GCA (see Section 3 for details). Moreover, GCA builds larger groups through constant verifications, which requires a lot of computation that significantly increases its runtime.

Collision optimized exhaustions use collision information to establish the relationship between different sub-keys. Each pair of collision results in the removal of a large number of wrong candidates. Taking AES-128 as an example, if a pair of sub-key values of the first and second S-boxes are deleted since no collision occurs, then all k^{14} possible full keys of this guessing pair and the subsequent 14 sub-keys within threshold k are deleted. An attacker can quickly recover the key from the remaining space which is much smaller than the original one after collision optimization. Unlike key enumeration, collision optimized exhaustions search all remaining keys with the same probability. In this case, collision values under CCA provide equal amount of collision information for sub-keys. As long as the correct full key and the corresponding collision values are within reasonable thresholds, the collision optimized exhaustions ensure successful key recovery. It is noteworthy that both key enumeration and collision optimized exhaustions are algorithm implementations. As such, their search performance depends on the efficiency of the algorithms. A more efficient algorithm will en-

able the key to be recovered from a deeper space (larger thresholds k and d) in a shorter time than its slower counterparts.

1.2 Our Contributions

Due to the insufficient use of collision information, FTC and GCA leave the attacker a large number of possible full keys when performing search in large intractable spaces. Moreover, although GCA uses collisions several times more than FTC and has much stronger search ability than FTC, the efficiency of its implementation is very low, leading to very time-consuming searches in large candidate spaces. In this paper, we propose Full Collision Attack (FCA), which efficiently eliminates the wrong candidates by using collision information between any two sub-keys, thus having much stronger search capability than FTC and GCA. In addition, by observing the distribution of collisions in the intractable space of FCA, we obtain an interesting finding that optimization can be achieved by performing error tolerance only on several sub-keys. Based on this finding, we propose Rotational Error Tolerant FCA (RET-FCA), which further improves the performance of FCA. Experiments on the public power trace set DPA *contest* $v4.1$[1] demonstrate the superiority of our scheme.

1.3 Organization

This paper is organized as follows: Measurement setups, Collision Attack (CA), Test of Chain (TC), Fault Tolerant Chain (FTC) and Group Collision Attack (GCA) are introduced in Section 2. Our Full Collision Attack (FCA) and the corresponding algorithms are detailed in Section 3 with the help of an example. Our finding that collision values beyond the search ability of FCA can be obtained by applying error tolerance on only a few sub-keys and the proposed rotational error tolerance strategy RET-FCA are described in Section 4. Rotational error tolerance for different sub-keys needs to repeatedly compute a large number of collisions. We use the prefix chains to optimize this in Section 5. Experiments on measurements from DPA *contest* $v4.1$ [1] are presented in Section 6 to compare our FCA and RET-FCA, with GCA and FTC. Finally, we conclude this paper in Section 7.

2 Preliminaries

2.1 Measurement Setups

Our experiments are performed on the public power trace set provided by DPA *contest* $v4.1$ [1] implementing Rotated S-boxes Masking (RSM) [21] protected AES-256 algorithm on the Side-Channel Attack Standard Evaluation Board (SASEBO). We then implement our experiments on MATLAB $R2016b$ on a HP desktop computer with 6 Intel(R) Xeon(R) E5-1650 v2 CPUs, 16 GB RAM and a Windows 10 operating system. Based on the suggestion in [25] that there

should be at least a clock cycle between two Points-of-Interest (POIs) [8], Ou et al. in [22] used CCA to find the 4 POIs with maximum correlation coefficients from the 100000-th and 110000-th time samples and performed GCA on them. The other 15 S-boxes were aligned to the first one. Since the performance of CPA is higher than that of CCA, CPA outputs the correct sub-keys before CCA can produce correct collision values. We use the optimal leakage model provided in [19] to perform first-order CPA to find and intercept 700 time samples of the first S-box (from the 100301-th to the 101000-th). As there are five regions with high correlation coefficients in these samples, we extract 5 best POIs with highest correlation coefficients from them. CPA and CCA in FTC, GCA, FCA, and RET-FCA are performed on these samples. This reduces the gap between the performance of CCA and CPA, and makes the key recovery process more challenging.

2.2 Collision Attacks

AES-256 performs the SubBytes operation (16 parallel S-box applications) in its first round. Let k_i and k_j denote the i-th and j-th sub-keys, and p_i and p_j denote the corresponding encrypted plaintext bytes. A generalized internal AES-256 linear collision 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:

$$\mathsf{Sbox}(p_i \oplus k_i) = \mathsf{Sbox}(p_j \oplus k_j), \tag{1}$$

which is equivalent to

$$p_i \oplus k_i = p_j \oplus k_j. \tag{2}$$

Since the 16 S-boxes of AES-256 are exactly the same in each round, the following collision value is obtained:

$$\Delta_{i \leftrightarrow j} = k_i \oplus k_j = p_i \oplus p_j. \tag{3}$$

For simplicity, this paper uses $k_i \leftrightarrow k_j$ to represent this collision. Although the specific values of these two sub-keys are unknown, they have a fixed XOR value that can be distinguished from other possible guessed collision values by a distinguisher. In this case, collision attacks establish relationships among multiple sub-keys.

Correlation enhanced Collision Attack (CCA) [20] is a typical collision attack. It divides the power traces of each S-box into 256 categories according to the input plaintext byte values of AES-256, then computes the correlation between them under different guessed collision values. The detailed algorithm implementation is explained in [30]. It is noteworthy that the efficiency of CCA is much lower than that of CPA. However, this is not surprising since CCA is mainly used for attacking flawed masking implementations such as DPA contest $v4.1$ in [19]. Some contributions like [33] also aim to secure masked implementations under CCA. If CPA and CCA are directly performed on the 700 samples and 5 POIs

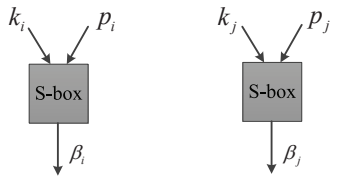

Fig. 1. A linear collision between the i-th and the j-th S-boxes happens if $\beta_i = \beta_j$.

described in Section 2.1 respectively, their performance will still exhibit a big gap. In this paper, similar to [22], we also introduce Amplified Template Attack (ATA) [34] into CCA to reduce its performance gap with CPA. Specifically, we use the five POIs to construct 256 templates for plaintext byte values. Measurements of each plaintext byte value p_i is matched with the template of plaintext p_j satisfying Eq. 3 and an Euclidean distance is calculated in the attack phase. The matching performance under a guessed collision value is expressed by the cumulative value of Euclidean distances, and the correct collision value corresponds to the minimum cumulative value. It is worth mentioning that although both FTC and GCA use CCA and CPA to find collisions, Ou et al. defined them as post-processing tools of distinguishers' outputs, which are independent of the specific distinguishers (see [22]).

2.3 Fault Tolerant Chain

The attacker obtains $k_i \leftrightarrow k_j$ if the i-th and the j-th S-boxes collide. Bogdanov et al. proposed Test of Chain (TC) in [3], which attempts 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 no specific schemes were given. Wang et al. provided the first practical scheme named Fault Tolerant Chain (FTC) in [30]. FTC tries to find collisions between the first sub-key and other 15 sub-keys: $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$, ..., $k_1 \leftrightarrow k_{16}$. Its performance is notably influenced by k_1. TC requires all sub-keys and collision values to be within the thresholds k and d. Evidently, this either requires many power traces or the thresholds need to be set at very large values. Unlike TC, FTC allows several sub-keys and collision values to exceed the thresholds, and searches them independently according to the ranks of CPA outputs. In other words, FTC enumerates them from the most possible one to the least possible one independently.

If there is no collision between k_1 and k_i, which means that all possible guessing values of k_1, k_i and $k_1 \leftrightarrow k_i$ are not within the thresholds simultaneously, FTC assumes that an error has occurred. However, if the threshold k is large, even if the correct sub-key exceeds the threshold, some false collision values can result in false sub-key values colliding with k_1. FTC will regard these false sub-key values as correct ones. Another shortcoming of FTC is that it enumerates

the sub-keys identified as errors. If too many errors occur, FTC has to enumerate a very large number of possible keys, which is infeasible. This usually happens when the threshold k is increasing and a large number of wrong key values are not successfully deleted. The search complexity of FTC is

$$2^8 \cdot \left(2^8\right)^n \binom{15}{n} \tag{4}$$

if there are n wrong sub-keys (see Section 3.2 in [30]). If errors happen on two sub-keys, a maximum of about 2^{31} searches are required. From the perspective of collision information utilization, FTC and TC only use 15 pairs of collisions. Since there is a pair of collision between any two correct sub-keys and 120 pairs of collisions in total, most of collision information is unused. Although FTC is not perfect, it triggers new opportunities for collision attacks.

2.4 Group Collision Attack

Group Collision Attack (GCA) proposed in [22] is ingeniously designed. Taking AES-256 as an example, GCA divides the 16 sub-keys into 4 non-overlapping groups of equal size, and performs a round of wrong candidates deletion within them. Then, the verification chains are used to further delete the wrongly guessed keys to prevent explosive growth in the search space. The construction of verification chains is the same as that of groups. It is difficult to make full use of collision information within or among groups. GCA achieves this goal through continuous reconstructions and verifications (as shown in Fig. 2). For chain based GCA (see Fig. 2(a)), the attacker finds a collision between the first and the second sub-keys, and another collision between the second and third sub-keys. If there is a guessed value of k_2 that leads to $k_1 \leftrightarrow k_2$ and $k_2 \leftrightarrow k_3$ being established simultaneously, then the attacker obtains a longer chain that includes three sub-keys $k_1 \leftrightarrow k_2 \leftrightarrow k_3$. If $k_1 \leftrightarrow k_3$ is also established, then these three sub-key values form a ring. To avoid complex computation, the verification between two rings consisting of k_1, k_2, k_3 and k_1, k_2, k_4 is similar to the construction of long chains (see Fig. 2(b)), wherein the guessed values of k_1 and k_2 should be the same. Since chains (rings) are stored in separate rows in implementation, it is very convenient to construct long chains (rings) from short ones. This collision optimized exhaustion is very simple and has been demonstrated to be very efficient.

GCA considers the collision situation where the threshold $\mathsf{d} > 1$, which is more complicated than only considering $\mathsf{d} = 1$ in TC and FTC. GCA uses much more collision information than FTC to eliminate the wrong sub-key candidates, which makes its search ability stronger than FTC. Taking the most efficient ring based GCA as an example, it uses 40 pairs of collisions to remove wrong key candidates within 8 groups (4 for verification, see [22]). Verification chains are further used to delete wrong sub-key candidates among groups, but this second round deletion does not take advantage of collision information from additional sub-keys except the 8 constructed verification chains or rings. In summary, GCA

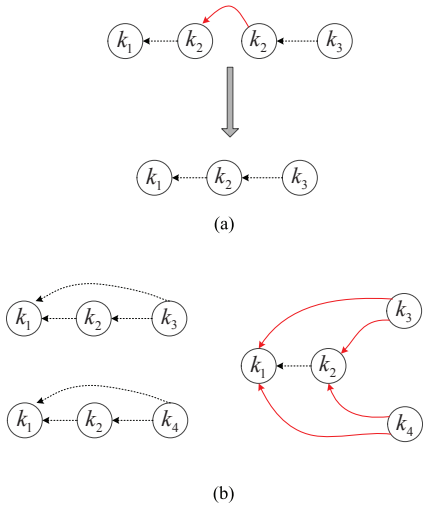

Fig. 2. Chain based GCA (a) and ring based GCA (b).

has its limitations: (1) The algorithm runtime is too slow since GCA needs constant combinations and verifications, and (2) there are too many remaining keys after two rounds of deletions, since the remaining combinations in 4 groups can only be recombined by multiplication. In other words, GCA can not prevent the explosive growth in the number of recombined key candidates under very large thresholds `k` and `d` (see Section 6).

3 Full Collision

We first highlight the limitation of GCA before describing the core idea of the proposed FCA in the subsequent sub-section. Taking the 16-byte key in the first round of AES-256 as an example, ring based GCA given in Fig. 2 uses information from 40 pairs of collisions for its first round of wrong key candidates deletion, and employs verification chains for an additional round of deletion as explained in Section 2.4. This enables GCA to outperform TC and FTC. However, the efficiency of GCA's implementation has a lot of room for improvement, and there are still 80 pairs of collisions that have not been utilized. The latter accounts for two-thirds of the collision information. When the sub-keys and collision values are located in much deeper spaces, GCA quickly fails. In view of the above limitations of GCA, we design a more efficient collision optimization called Full Collision Attack (FCA), which can utilize information from all 120 pairs of collisions to quickly delete the wrong key candidates.

3.1 Core Idea of FCA

The proposed FCA is capable of exploiting all the collision information between any two sub-keys. We will explain how this is achieved with the help of the example in Fig. 3. The white circles represent a set of guessed sub-keys constituting a Full Collision Chain (FCC), while the gray circles represent the sub-keys currently waiting to be added. FCA sequentially searches for full collision information of each sub-key. It first finds all possible collisions of the first sub-key k_1 and the second sub-key k_2 (the corresponding guessing ranks are denoted as Φ_1 and Φ_2, and guessing ranks of all sub-keys are represented by Φ), and stores each of them as a row in table Θ. To consider the new sub-key k_3, FCA traverses each row in table Θ to determine whether the guessed k_1 and k_2 collide with k_3 simultaneously. If so, FCA obtains a FCC of these 3 sub-keys from the FCC of the guessed k_1 and k_2, which is equivalent to R^3 in GCA (see Section III-B in [22]). FCA stores the new FCC in table Θ^1 and replaces it with Θ after traversing all possible collisions in Θ consisting of Φ_1, Φ_2 and Φ_3. It then considers k_4 and proceeds to the subsequent iterations (see Fig. 3(b) and Fig. 3(c)). The iteration terminates after Φ_{16} of the last subkey k_{16}. Since every iteration in FCA ensures that the current Θ preserves all possible FCCs of the current sub-key, it guarantees the full utilization of collision information. In summary, the advantage of FCA lies in its traversal algorithm and efficient storage that enable it to rapidly perform the collision detection and verifications among 16 sub-keys.

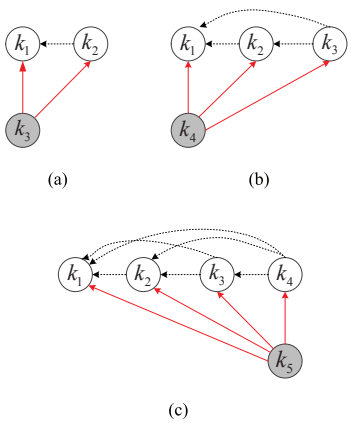

Fig. 3. The procedure of FCA.

The number of possible FCCs increases rapidly in FCA because of the limited information of collisions within the first several sub-keys. It is necessary to

search for $i-1$ pairs of collisions for each FCC in Θ to construct a new FCC when the new sub-key k_i is added. The more collision information that needs to be searched, the more time-consuming the algorithm is. Fortunately, as larger number of wrong key candidates are also deleted, the runtime of FCA also decline rapidly after a rapid initial surge. Thus, the main computation is focused on the collision construction of the first several sub-keys. This phenomenon will be described in detail in the example given in Section 3.4. FCA represents the upper bound of the wrong key candidates that an attacker or evaluator can remove under given thresholds k and d.

3.2 Collision Detection Optimization

The collision detection may be repeated when constructing the FCCs, which seriously lowers down the efficiency of the algorithm. For FTC and GCA, in order to find collisions between two sub-keys k_i and k_j, it is necessary to traverse all possible sub-key values in Φ_i and Φ_j, and collision values $\Delta_{i\leftrightarrow j}$ (the corresponding guessing ranks are denoted as $\Psi_{i\leftrightarrow j}$) within thresholds k and d, with the complexity of $\mathsf{d}\cdot\mathsf{k}^2$. FCA successively computes collisions $k_1 \leftrightarrow k_2$, $k_1 \leftrightarrow k_3$, ..., $k_{14} \leftrightarrow k_{15}$ and $k_{15} \leftrightarrow k_{16}$, and converts all collision values within the threshold d into flags. The collisions between k_i and k_j are saved in the loc-th column of the flag matrix F where loc satisfies

$$loc = 15 \cdot i - (i-1)(i-2)/2 + (j-i). \tag{5}$$

If a collision value $\Delta_{i\leftrightarrow j}$ is within threshold d, FCA sets $F(loc, \Delta_{i\leftrightarrow j})$ to 1. Thus, FCA obtains the flag matrix F with $120\cdot 256$ flags. In this way, to check whether a combination of k_i and k_j collides within the thresholds, FCA only needs to check whether the flag $F(loc, \Delta_{i\leftrightarrow j})$ is 1, and the complexity can be reduced to k^2, which significantly improves the collision detection efficiency.

3.3 Algorithm Description

Unlike GCA, FCA is simple, efficient and easy to implement. It consists of two simple sub-algorithms (see Algorithms 1 and 2). Algorithm 1 is the main function, which implements the whole FCA where each sub-key is added. Its inputs include all guessing collision ranks Ψ output by CCA and all ranks Φ of the 16 sub-keys output by CPA, as well as thresholds k and d. All possible FCCs and corresponding number n are the outputs. Θ^1 represents the remaining FCCs after each sub-key being added, and also the initialization of the next iteration. FCA assigns the guessing values Φ_1 of the first sub-key to Θ and computes the flag matrix F to initialize the algorithm (Steps $1 \sim 2$). It finds all FCCs between the current i-th sub-key and each FCC constituted by $k_1 \sim k_{i-1}$ in Θ and returns all new FCCs to Θ^1 (Step 4). If Θ^1 is null, there is no collision between the current guessing sub-key and $k_1 \sim k_{i-1}$ and FCA exits the current iteration (Steps $5 \sim 7$).

Algorithm 1: Full Collision Attack.

Input: guessing sub-keys ranks Φ, guessing collision ranks Ψ, thresholds k and d.

Output: n and the remaining FCCs in Θ.

1 $\Theta \leftarrow \Phi_1$;
2 $F \leftarrow \mathsf{BuildFlag}\ (\mathsf{d}, \mathsf{k}, \Phi, \Psi)$;
3 **for** i *from* 2 *to* 16 **do**
4 $\quad (\Theta^1, n) \leftarrow \mathsf{Collisions}\ (F, \mathsf{k}, \Theta, \Phi_i)$;
5 $\quad$ **if** $\Theta^1 = \emptyset$ **then**
6 $\quad\quad$ return i;
7 $\quad\quad$ break;
8 $\quad$ **end**
9 $\quad \Theta = \Theta^1$;
10 **end**

The function Collisions used in FCA in Algorithm 1, which constructs all possible FCCs considering the new ranks Φ_i of sub-key k_i, is shown in Algorithm 2. The inputs of the algorithm are the flag matrix F, threshold k, Θ including all FCCs of the previous $i-1$ sub-keys and Φ_i. The algorithm consists of three loops. For each possible value in Φ_i, FCA traverses each row in Θ to see if it fully collides with FCCs of the previous $i-1$ sub-keys. Specifically, the position loc of the corresponding collision values in F is calculated according to iteration r and i, and the possible sub-key value in Θ_q^r is XORed with Φ_i^p (Steps $5 \sim 6$). Here Φ_i^p represents the p-th element of Φ_i and Θ_q^r denote the r-th guessed sub-key of the q-th FCC in Θ. If the XOR value is in Ψ_{loc}, then Φ_i^p and Θ_q^r collide, the collision counter cn of Φ_i^p increases by 1 (Steps $7 \sim 9$). FCA detects the collision counter to see if Φ_i^p collides with all the previous $i-1$ guessed sub-key values in Θ_q after traversing a FCC. If so, Θ_q and Φ_i^p will be added to Θ^1 as a new FCC, and the current number of FCCs n^1 increases by 1 (Steps $11 \sim 14$). n and Θ are updated with n^1 and Θ^1 when considering the $(i+1)$-th sub-key in the next iteration (Steps $17 \sim 18$).

3.4 An Example

We describe the case where the correct sub-keys and collision values are ranked in very deep spaces in Fig. 4, 60 power traces are randomly selected to perform CPA and CCA. The rank of the deepest sub-key k_{12} is 81, and the ranks of the second and eighth sub-keys is about 80. The rank of the deepest collision value is the one between k_3 and k_6, which ranks at 84. Therefore, in order to ensure that FCA can recover the key, the thresholds k and d should be set at least to 81 and 84 respectively. In this case, the key candidate space and collision detection space reach $2^{101.4376}$ and $2^{102.2771}$, which are about $2^{21.4376}$ and $2^{22.2771}$ times larger than the maximum thresholds $\mathsf{k} = 32$ and $\mathsf{d} = 32$ discussed in [22]. Fig. 4(b) shows that 7 of the 15 pairs of collisions between k_1 and $k_2 \sim k_{16}$ exceed

Algorithm 2: The function Collisions used in FCA.

Input: flag matrix F, threshold k, Θ and Φ_i.
Output: n and the remaining FCCs in Θ.

for *p from 1 to* k **do**
  for *q from 1 to n* **do**
    cn=0;
    for *r from 1 to* $i-1$ **do**
      $loc \leftarrow \mathsf{GetLocation}\,(r, i)$;
      $temp = \mathsf{Xor}\left(\Phi_i^p, \Theta_q^r\right)$;
      if $F(loc, temp) = 1$ **then**
        $cn++$;
      end
    end
    if $cn = i - 1$ **then**
      n^1++;
      $\Theta^1 \leftarrow \Theta^1 \cup [\Theta_q, \Phi_i^p]$;
    end
  end
end
$n \leftarrow n^1$;
$\Theta \leftarrow \Theta^1$;

threshold $\mathsf{d} = 1$, which pose a problem for FTC as there are too many possible keys to be exhaustively searched (see Eq. 4).

This paper considers the FTC under threshold $\mathsf{d} > 1$ for the first time. It can also be seen from the experimental results given in Fig. 13(a) that FTC leaves intractable spaces (about $2^{65.6} \sim 2^{66.5}$) for the attacker under $\mathsf{k} = 64$ and $\mathsf{d} = 64$ when 60 power traces are randomly selected in each repetition. These spaces will be much larger when $\mathsf{k} = 81$ and $\mathsf{d} = 84$. Moreover, the key itself is located in a space that can not be enumerated, and the current key enumeration methods will fail. However, using the collision optimization discussed in Section 3.2, FCA can rapidly reduce a total of $2^{103.8696}$ possible key combinations from the huge thresholds $\mathsf{k} = 90$ and $\mathsf{d} = 84$ to only 58 in 326.5 seconds (as shown in Fig. 5). Verifying the key from these 58 candidates becomes a trivial task. Therefore, FCA exhibits a powerful search capability under large thresholds.

The attacker does not know the optimal thresholds k and d in real attack scenarios. In other words, he does not know the exact ranks of sub-keys and the collision values, and can only constantly test them. The number of remaining FCCs when 15 sub-keys are successively added under the thresholds near $\mathsf{k} = 84$ and $\mathsf{d} = 84$ is shown in Fig. 5. Although only one pair of collision between k_3 and k_6 falls outside thresholds $\mathsf{k} = 84$ and $\mathsf{d} = 80$, only a FCC remains when the 11-th sub-key is added, and no FCC when considering the 12-th sub-key. Similar phenomenon occurs at k_{12} when $\mathsf{k} = 84$ and $\mathsf{d} = 82$. We also consider the case of $\mathsf{k} = 76$ and $\mathsf{d} = 84$, and there is no FCC in the first 12 sub-keys. Some curves are not given since they are too dense in Fig. 5. However, there are

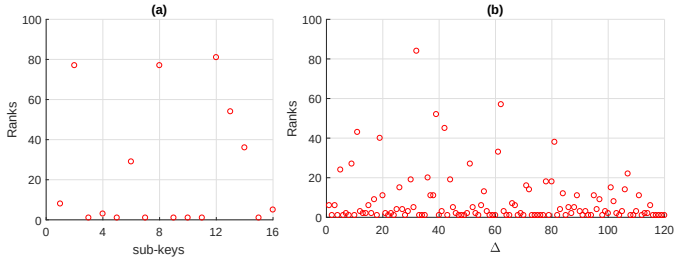

Fig. 4. The ranks of sub-keys (a) and correct Δ-s (b).

36 (47) FCCs left in Θ when $\mathsf{k} = 80$ (82) and $\mathsf{d} = 84$. If we continue to enlarge k and d, these erroneous FCCs will always be in Θ and more erroneous ones will satisfy full collision. This phenomenon is more obvious in FTC and GCA since they only use a small part of collisions. It is worth noting that d can not be enlarged indefinitely. It is equivalent to exhausting all possible keys in threshold k when $\mathsf{d} = 256$. Fortunately, the number of FCCs under FCA does not increase significantly. For example, when $\mathsf{k} = 84$ and $\mathsf{d} = 80$, there are 0 combinations left, and becomes 65 combinations when d reaches 90. There are 6 combinations remaining when $\mathsf{k} = 78$ and $\mathsf{d} = 84$, and 58 combinations when k reaches 90. This also indicates that the number of FCCs under several close thresholds is almost the same.

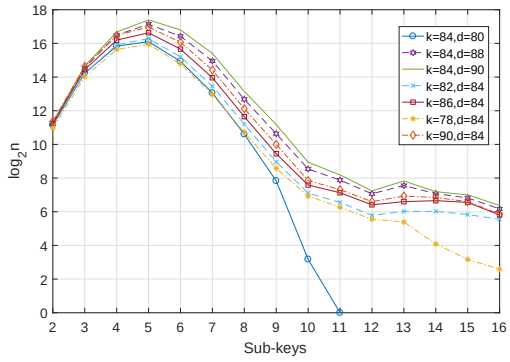

Fig. 5. The number of FCCs under each sub-key.

The number of remaining FCCs in FCA is related to the distributions of sub-keys and collision values, as well as the size of k and d. The complexity of constructing the FCCs of sub-key k_i in Algorithm 2 is $(i-1)\cdot\mathsf{k}\cdot n$. n varies greatly in different thresholds, and is similar to R^3 in GCA under three sub-keys (see

Section III-B in [22]). This is due to the fact that few pairs of collisions are used to verify the initial several sub-keys, resulting in a very limited number of error combinations being deleted. This is demonstrated in Fig. 5, where the number of FCCs increases rapidly when the second to sixth sub-keys are added. It even reaches from 60,000 to 130,000 at the fifth sub-key. This results in a large amount of collision computation when the sub-keys from third to eighth are added. If larger thresholds are considered, the FCA performed on the first several sub-keys becomes slower. The number of remaining FCCs under different thresholds in Fig. 5 is less than 2^9 after the 10th sub-key. Obviously, the construction of subsequent FCCs becomes very fast, and just accounts for only a small part of the algorithm overhead.

4 Error Tolerance

4.1 Distribution of Collision Values Beyond Threshold

FCA can efficiently delete wrong key candidates by maximizing the use of collision information and quickly reduce candidate space. Therefore, the thresholds k and d can be set very large under FCA, and the number of remaining FCCs n will not be so inexhaustible as FTC and GCA. If the key is within the threshold, it can be recovered easily. However, FCA requires all 120 correct collision values to fall within threshold d. There may be a few collision values in much deeper space than we have discussed in Section 3.4 previously, which either makes the execution time of FCA unbearable or difficult to reduce the candidates space to exhaustible one. Obviously, this requires error tolerance for FCA that allows a small number of collisions to be outside the threshold d. In other words, we consider the chains that are not so "full". A feasible error tolerance scheme is to record the number of collisions and set a reasonable threshold when building each FCC. The attacker checks the collision state between the new sub-key and the previous sub-keys. If the number of collisions in a chain is smaller than the threshold (e.g. two-thirds of collisions are required, but only one-third are actually established), then delete it. However, the implementation of this algorithm is complicated.

An interesting phenomenon is the collision values outside the threshold d do not occur randomly, but are related to one or several sub-keys. In fact, this can be analyzed from the selected power traces. If the Signal-to-Noise Ratio (SNR) [17] of power traces of a S-box is relatively low, the collision values in CCA between it and other sub-keys are also ranked relatively backward. Moreover, it is difficult to align the power traces of all S-boxes strictly. The correct collisions values outside $\mathsf{d} = 40$ are $k_3 \leftrightarrow k_6$, $k_3 \leftrightarrow k_{13}$, $k_3 \leftrightarrow k_{16}$ and $k_5 \leftrightarrow k_{13}$ in Fig. 4. A close observation reveals that these collisions are related to k_6, k_{13} and k_{16} (or k_3 and k_5). If considering error-tolerance on these three (or two) sub-keys, d can be reduced from 84 to 40 (or 44). If only one sub-key is considered, we only need to take k_3 or k_6 into account, and the threshold can be set to 58 in Fig. 4.

Another experimental result from CPA and CCA performed on new randomly selected 60 power traces is shown in Fig. 6. The deepest sub-key k_{16} is ranked

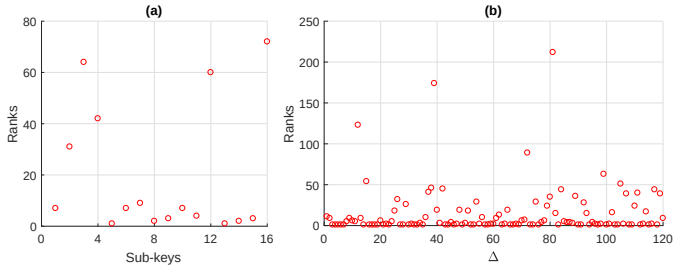

Fig. 6. Error tolerance is required if sub-keys (a) and correct Δ-s (b) are ranked at a depth that FCA can't conquer.

at 72. Unlike Fig. 4, the collision values in Fig. 6 are much deeper. The deepest collision value that happens between k_7 and k_{13} even reaches 212. However, only collisions $k_1 \leftrightarrow k_{13}$, $k_1 \leftrightarrow k_{16}$, $k_3 \leftrightarrow k_{13}$, $k_6 \leftrightarrow k_{13}$, $k_7 \leftrightarrow k_{13}$, $k_9 \leftrightarrow k_{16}$ and $k_{10} \leftrightarrow k_{16}$ are out of threshold $\mathsf{d} = 50$. These collisions are related to k_1, k_3, k_6, k_7, k_9 and k_{10}, or only k_{13} and k_{16}. If the error-tolerant two sub-keys k_{13} and k_{16} are required to satisfy 7 pairs of collisions, the collision threshold d can be easily reduced from 212 to 50, so that the unrecoverable key under FCA can be quickly conquered under error tolerance. If considering only one sub-key in error-tolerance, then k_{13} is considered and d can also be reduced to 64.

4.2 RET-FCA

The instances of collision values falling outside the threshold d given in Sections 3.4 and 4.1 are prevalent. This conclusion can also be drawn from the experimental results given in Section 6. In this case, Rotational Error Tolerant FCA (RET-FCA) will reduce the threshold d. Error tolerance here also means that the chains we obtain are Partial Full Collision Chains (PFCCs). Specifically, most of the sub-keys can be randomly selected to construct FCCs, and consider error-tolerance on other sub-keys. For simplicity, here we only take 7 sub-keys shown in Fig. 7 for example, wherein the sub-keys are divided into full collision part (the first 5 sub-keys) and error-tolerant part (k_6 and k_7). The construction of full collision part is exactly the same as that of FCA, and only a few pairs of collisions are required between these two parts. Thus, if the collision values outside the threshold d are all related to k_6 or k_7, the collision threshold can be lowered to d under RET-FCA. The smaller the number of collisions between the two parts is required, the more significantly the threshold d decreases, and the more PFCCs remain. For AES-256 or AES-128 with 16 sub-keys, we can allow up to 5 pairs of collision values for each sub-key in the error-tolerant part to be outside the threshold d.

The specific distribution of collision values under a certain attack is unknown. Moreover, we find that the collision values which need error tolerance does not always happen on the sub-key leading to Θ an empty set in many experiments.

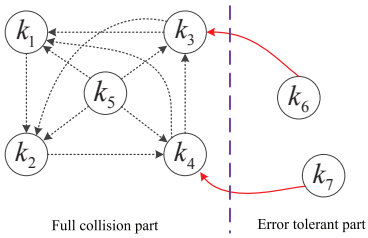

Fig. 7. Rotational Error Tolerant FCA (RET-FCA).

For example, only $k_3 \leftrightarrow k_6$ is out of thresholds $\mathsf{k} = 84$ and $\mathsf{d} = 80$ in Fig. 4, but empty set Θ happens on k_{11}. Therefore, it is difficult to achieve pertinent error-tolerance. We can only take a rotational error-tolerance, assuming that collisions on m different sub-keys are outside threshold d in each round. If the correct full key is not found in the current error tolerant round, m other sub-keys are randomly selected for a new round of error tolerance. This guarantees that all possible PFCCs are in Θ. For n sub-keys, if considering error-tolerance on m sub-keys, a total of $\binom{n}{m}$ rounds are required. For example, if considering error-tolerance on two sub-keys in AES-128 or AES-256, the upper bound of error-tolerance is 120 rounds. Obviously, an advantage of rotational error tolerance for m sub-keys is that as long as all collision values outside the threshold d are related to any one of these m sub-keys, the attacker can conquer the key if he sets a collision threshold to be greater than or equal to d.

It can be seen from Section 3.4 that the main computational load of FCA comes from the first several sub-keys, and the subsequently considered sub-keys result in the significant decrease in the number of possible keys. For example, only 115 possible FCCs of the first 14 sub-keys remain when the thresholds are $\mathsf{k} = 90$ and $\mathsf{d} = 84$ in Fig. 5. The addition of the last two sub-keys only results in a very small amount of computation, which is almost negligible compared with the previous calculation. For RET-FCA, this is equivalent to performing FCA for 120 repetitions under $\mathsf{k} = 90$ and $\mathsf{d} = 44$, which may incur a longer runtime for error tolerance compared to FCA. However, since FCA itself can efficiently recover the key from very large thresholds, RET-FCA serves only to recover the key from much larger spaces where FCA encounters difficulty, such as the typical example given in Fig. 6.

4.3 Exhaustion Avoidance

All combinations of k_6 and k_7 shown in Fig. 7 are exhaustively searched if no collisions between them are required as a precondition of RET-FCA. This is very time-consuming when the thresholds k and d are very large. Although error tolerance is considered on the two sub-keys k_{13} and k_{16} in Fig. 6, there is a

collision between them. Furthermore, FCA selects two new sub-keys for error tolerance in each repetition, and it will find other combinations satisfying the error tolerant conditions. This pair of sub-keys are used to construct FCCs in the next round of error tolerance. Therefore, the collision between these two sub-keys can be required as a precondition of RET-FCA to avoid exhaustively searching all possible combinations. This error-tolerant scheme can also be extended to the case of 3 or 4 sub-keys and requires a small number of collisions among them, thereby reducing error-tolerant rounds. If the correct key is not found under the current thresholds $\mathtt{k}$ and $\mathtt{d}$, then they are increased in the next round of error tolerance. FCA and RET-FCA can quickly recover keys from deeper candidate spaces than FTC and GCA, but if the sub-keys and their collision values are located in the space that they can't exhaustively search, more advanced FCA schemes should be taken into consideration.

5 Collision Optimization

The purpose of RET-FCA is to reduce $\mathtt{d}$, so that sub-key values can still be recovered even if collision values fall in the space where FCA cannot cope with. However, error tolerance for multiple sub-keys introduced in Section 4.2 requires repeated computations for the same collision, which is time-consuming. For example, 16, 120, 560 rounds of computation are required if considering error tolerance on 1, 2 and 3 sub-keys. Therefore, this needs to be optimized. In this paper, prefix chain, a new concept is proposed to optimize 120 rounds of error tolerance of two sub-keys to reduce the repeated computation of collisions.

5.1 Prefix Chains

In order to avoid repeated computation of collisions, one method is to calculate a part of FCCs first, other collisions are then calculated from them. For example, if both $k_1 \sim k_9$ and $k_2 \sim k_9$ are FCCs, then the prefix chain $k_1 \sim k_9$ is used to construct the FCCs of $k_1 \sim k_{10}$. In this case, the computational complexity of constructing the FCCs of newly added sub-keys is the smallest. It is noteworthy that the repeated calculation of collision is unavoidable and can only be minimized. A typical example is shown in Fig. 7, which considers error-tolerance on k_6 and k_7. If considering error tolerance on k_5 and k_7 in the previous round, the FCCs of $k_1, \ldots, k_4$ and k_6 are computed and saved to Θ. When performing error-tolerance on k_6 and k_7, the FCCs of $k_1 \sim k_4$ can not be directly extracted from Θ to construct the FCCs of $k_1 \sim k_5$. Since the addition of k_6 deletes many FCCs of $k_1 \sim k_4$, the FCCs extracted from Θ are only a sub-set of the whole. It also illustrates that although any sub-key combination in the FCCs is also FCCs, the short chains can not be extracted from the long chains for the other rounds of error tolerance. In this case, we can only construct FCCs of $k_1 \sim k_4$ and save them in Θ. If we perform error-tolerance on k_5 and k_7, we can construct FCCs of $k_1 \sim k_4$ and k_6 from Θ. If we perform error-tolerance on k_6 and k_7, we can also construct FCCs of $k_1 \sim k_4$ and k_5 from Θ. This is clearly feasible.

The longer the prefix is, the more subsets exist, and the more repeated computations are required. The error tolerant performance is at the worst case when the length of prefix chains reaches 14, which means that 120 rounds of error tolerance are performed independently. A reasonable division of the subsets optimizes the efficiency of error tolerance. Due to the insufficient utilization of collision information, the sub-keys from k_4 to k_9 have a large number of FCCs, occupying the main calculation of FCCs of all 16 sub-keys (as shown in Fig. 5). There are only less than 2^9 FCCs after the 10-th subkey under the very large thresholds $\mathsf{k} = 90$ and $\mathsf{d} = 84$. According to this, we set the length of the prefix chain to 10, and reduce the repeated construction of FCCs of the first 10 sub-keys to optimize RET-FCA.

5.2 Collision Division

Only up to six sub-keys are allowed in a subset of PFCCs when the length of prefix chain is set to 10. For example, $k_{11} \leftrightarrow k_{12}$, $k_{11} \leftrightarrow k_{13}$, $k_{11} \leftrightarrow k_{14}$, $k_{11} \leftrightarrow k_{15}$ and $k_{11} \leftrightarrow k_{16}$ have the same prefix $k_1 \sim k_{10}$ and the attacker only needs to calculate it once when performing error tolerance on these 6 sub-key pairs. In this case, 120 rounds of error tolerance in RET-FCA only needs to compute about 20 prefixes, which significantly reduces calculation of collisions. Another collision division is to find large subsets first and then small ones, of which a typical example is shown in Fig. 8. Here $\{k_i \sim k_j\}^*$ denotes all possible error tolerant combinations between any two sub-keys from k_i to k_j, and $\{k_i \sim k_j\} \leftrightarrow \{k_m \sim k_n\}$ denotes error-tolerant combinations of the sub-keys between the left set $\{k_i \sim k_j\}$ and right set $\{k_m \sim k_n\}$. Take $S_1 = \{k_1 \sim k_6\}^*$ for example, it represents 15 rounds of error tolerance between any two sub-keys of $k_1 \sim k_6$, which share the longest prefix chain $k_7 \sim k_{16}$. In this case, 120 rounds of error tolerance are divided into 11 non-overlapping sets $S_1 \sim S_{11}$ (the corresponding prefixes are shown in Table 1).

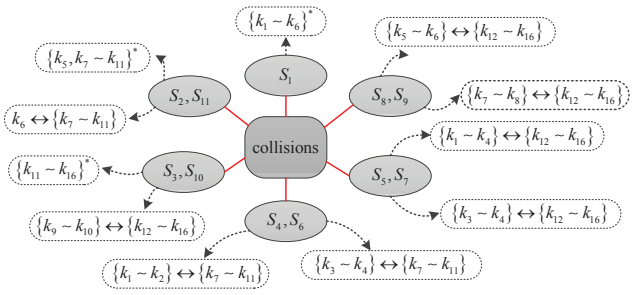

Fig. 8. Collision division in RET-FCA.

Table 1. First round of collision division in RET-FCA.

sub-sets	prefix chains	sub-sets	prefix chains
S_1	$k_7 \sim k_{16}$	S_7	$k_3 \sim k_{11}$
S_2	$k_1 \sim k_4$, k_6, $k_{12} \sim k_{16}$	S_8	$k_1 \sim k_4$, $k_7 \sim k_{11}$
S_3	$k_1 \sim k_{10}$	S_9	$k_1 \sim k_6$, $k_9 \sim k_{11}$
S_4	$k_3 \sim k_6$, $k_{12} \sim k_{16}$	S_{10}	$k_1 \sim k_8$, k_{11}
S_5	$k_1 \sim k_2$, $k_5 \sim k_{11}$	S_{11}	$k_1 \sim k_5$, $k_{12} \sim k_{16}$
S_6	$k_1 \sim k_2$, $k_5 \sim k_6$, $k_{12} \sim k_{16}$	–	–

Table 2. Second round of collision division in RET-FCA.

sub-sets	prefix chains	length
S_1	$k_7 \sim k_{16}$	10
S_2, S_{11}	$k_1 \sim k_4$, $k_{12} \sim k_{16}$	9
S_3, S_{10}	$k_1 \sim k_8$	8
S_4, S_6	$k_5 \sim k_6$, $k_{12} \sim k_{16}$	7
S_5, S_7	$k_5 \sim k_{11}$	7
S_8, S_9	$k_1 \sim k_4$, $k_9 \sim k_{11}$	7

The prefix chains rotate with the error-tolerant and full-collision parts in RET-FCA. We introduce two rounds of prefix chains computation to optimize them. 120 rounds of error tolerance are divided into 11 subsets with prefix chains including 10 sub-keys. These 11 subsets $S_1 \sim S_{11}$ in Fig. 8 are further divided. Each round of division entails several sub-sets having the same prefixes, thus avoiding repeated calculation as much as possible. For example, $S_2 = \{k_5, k_7 \sim k_{11}\}^*$ and $S_{11} = k_6 \leftrightarrow \{k_7 \sim k_{11}\}$ share the same prefix $\{k_1 \sim k_4\} \cup \{k_{12} \sim k_{16}\}$, while $S_3 = \{k_{11} \sim k_{16}\}^*$ and $S_{10} = \{k_9 \sim k_{10}\} \leftrightarrow \{k_{12} \sim k_{16}\}$ share the same prefix $k_1 \sim k_8$. These 11 small subsets shown in Fig. 8 are further divided into 6 subsets shown in Table 2. The FCCs with 10 sub-keys in Table 1 are calculated according to the prefixes in Table 2. Then, all 120 rounds of error tolerance calculation is completed. The above strategy is only a very simple optimization, other strategies to optimize RET-FCA can also be used.

6 Experimental Results

Based on Algorithms 1 and 2, we will continue to use n to denote the number of remaining key candidates after collision optimization. We consider the performances of FTC, GCA, FCA and RET-FCA under different thresholds k and d, and different numbers of measurements N, respectively. Since full keys that fall in Θ in many experiments are difficult to be searched exhaustively, it is unreasonable to use Success Rate [27] to compare the performance of them. Here we use P_r, the probability that the correct full key falls in Θ, as an eval-

uation criterion. Since the evaluations under large thresholds k and d are very time-consuming, we only repeat each experiment 100 times.

6.1 Experiments on Different Thresholds k

Firstly, we set the number of power traces N to 100 and threshold d to 64, and evaluate the performance of the four algorithms FTC, GCA, FCA and RET-FCA under different thresholds k. The choice of N and d is set empirically, and the settings we chose proved to be good thresholds in the remaining experiments. The remaining key candidate space of these four algorithms increases with k (as shown in Fig. 9). For GCA, the attacker can still recover the key from the remaining space Θ by exhaustive search when k is small. However, with the increase of k, GCA leaves a very large candidate space for the attacker, and the exhaustive search quickly becomes intractable. The situation for FTC is even worse than GCA. The remaining candidate space of it grows faster than GCA and reaches about 2^{50} when $k = 32$, and is close to 2^{70} when k reaches 72 (compared with about 2^{60} of GCA). Therefore, GCA is more efficient than FTC.

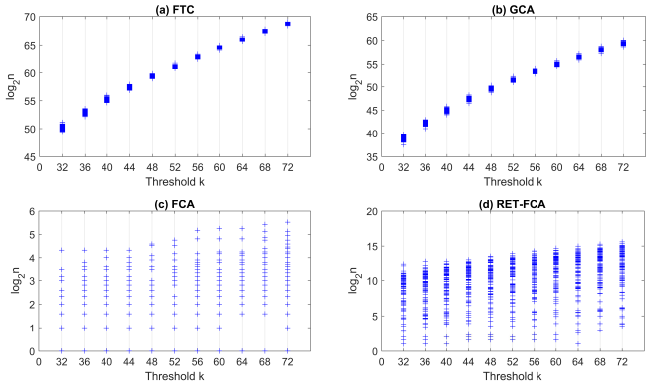

Fig. 9. The remaining key candidates under different thresholds k.

FCA utilizes collisions several times more than those of FTC and GCA, and thus has stronger ability to delete wrong candidates. It can reduce the large candidate space to smaller than 2^6 when k is varied from 32 to 72 (a total of $2^{80} \sim 2^{98.7188}$ key candidates). However, the probability P_r that the full key falling into the remaining candidate space Θ shown in Fig. 10 indicates that the number of wrong candidates deleted by FCA is the largest. Since only 100 measurements are used in each repetition, the correct sub-keys and collision values are ranked deeply, it is impossible for the ones within the thresholds k

and d to form a FCC. RET-FCA solves this problem well, it achieves almost the highest P_r under different thresholds (as shown in Fig. 10(a)). Since RET-FCA utilizes collisions several times more than FTC and GCA, it can obtain P_r similar to them and significantly reduce the number of remaining candidates. When threshold k of RET-FCA is from 32 to 72, there are only less than 2^{15} candidates remaining in Θ, which can be easily exploited by an attacker (also can be seen from Fig. 9 (d)).

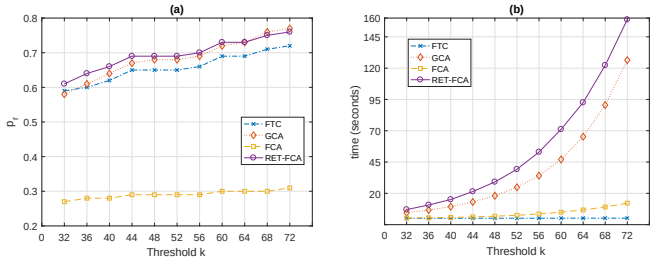

Fig. 10. The probability of correct full key falling within Θ (a) and average time consumption (b) under different thresholds k.

The P_r of RET-FCA increases from 0.61 to 0.76 when k is from 32 to 72, while P_r of FCA increases from 0.26 to 0.31. This also reinforces the phenomenon described in Section 4.1 where the collision values outside threshold d can be restricted to one or several sub-keys for error tolerance, and this situation is common. The "gap" between these two schemes under the same threshold k reflects the probability of this phenomenon. We only tolerate two sub-keys, and at most five collisions per sub-key are allowed to fall outside the threshold. In order to avoid exhausting the error-tolerant sub-keys, we require a pair of collisions between them (as introduced in Section 4.3). The attacker can obtain a higher P_r by allowing more collisions to be outside threshold d, or increasing the number of sub-keys considered in error tolerance. However, this means that more wrong candidates satisfy the current given collision conditions and remain in Θ, which makes it more difficult for attackers to verify their keys.

From the execution time comparison of the algorithms (see Fig. 10(b)), FTC performs the fastest, which uses the smallest amount of collision information and results in the least number of wrong key candidates being deleted. RET-FCA has the longest execution time, followed by GCA. FTC and FCA can recover the key in less than 15 seconds under thresholds $k = 72$ and $d = 64$, but GCA and RET-FCA need about 130 seconds and 160 seconds respectively. This also indicates that the time spent on evaluations is acceptable under the thresholds we choose. In terms of time growth, RET-FCA is much faster than FCA. The former requires twice as much runtime the latter when $k = 32$ and this reaches to about 10 times when $k = 72$. The increase of k causes more sub-keys candidates

to fall within the threshold (as shown in Fig. 10), which increases the probability of collisions and thus becomes more time-consuming.

6.2 Experiments on Different Thresholds d

The distribution of n under different thresholds d when k = 64 and $N = 100$ is shown in Fig. 11. Wang et al. discussed only the case of d = 1 in [30], and left the complex case of d > 1 as an open problem. Ou et al. compared GCA considering d > 1 with FTC in [22]. However, d was always set to 1. If none of the k guesses of sub-key k_i within the threshold collide with the k guesses of k_1, then there is no collision between k_1 and k_i. In this case, the attacker has to perform brute-force search on k_i. However, this probability becomes very small when d is large. Compared with k pairs, a total of k · d pairs of detections make it easier to satisfy collisions. In this case, it is almost impossible for FTC to collide with any guessed values of k_i within the threshold in our experiments. This conclusion can also be drawn from Fig. 9 and Fig. 11. It can be seen from the figures that the range of n of FTC narrows with the growth of k and d, and similar phenomenon occurs in GCA. However, since the two thresholds considered in [22] are much smaller than those considered in this paper, only a very small range of n can be observed, this phenomenon is not obvious, just like FCA in Fig. 11(c).

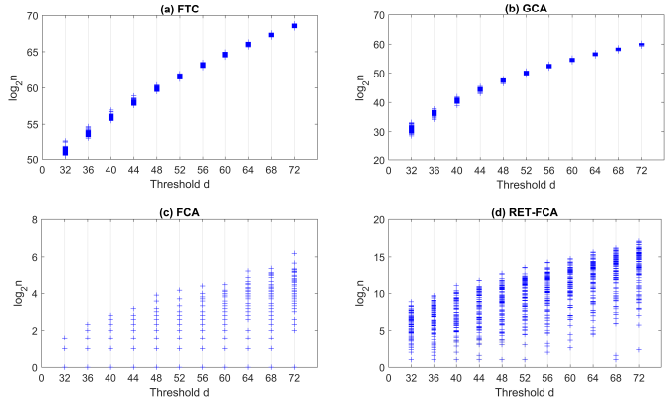

Fig. 11. The remaining key candidates under different thresholds d.

The remaining candidate space of FTC and GCA increases rapidly with d (as shown in Fig. 11). FTC behaves similarly in different k and d, while n of GCA grows faster with d than with k, and the range is more concentrated, although it is much lower at the start. It can also be seen from Fig. 9(b) and Fig. 11(b) that n approaches 2^{38} when d = 64 and k = 32 for the former, and only 2^{32} for the latter. Similar conclusions can be drawn from FCA and RET-FCA. Compared

Fig. 9 with Fig. 11, increasing k makes the growth of the remaining key candidate space of these four schemes faster.

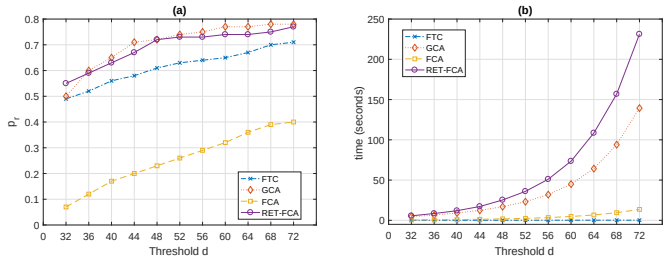

Fig. 12. The probability of correct full key falling within Θ (a) and average time consumption (b) under different thresholds d.

Increasing k also makes the growth of P_r of FTC, GCA, FCA and RET-FCA faster than increasing d (see Fig. 10 (a) and Fig. 12 (a)). This is especially true for FCA, wherein P_r increases from about 0.07 to about 0.40. To some extent, this also reflects that CPA's attack efficiency is higher than CCA's, which makes the correct values of sub-keys rank better than the collision values with higher probability. In this case, it is obvious that increasing k can not make more correct sub-key values fall within it. However, the increase of d can make more collision values fall within it, and the collision conditions are easier to be satisfied and allows for more possible full keys to remain in Θ. k = 64 and d = 72 take more time than k = 72 and d = 64, which indicates that more collisions are established (as shown in Fig. 12). This also proves that CPA has higher efficiency than CCA. It can also be seen that the runtime of FTC and FCA are the shortest, while GCA and RET-FCA are the longest. The time-consumption of GCA is embodied in the design of collision verification, while the time-consumption of RET-FCA is embodied in the rotation of error-tolerance. However, the four schemes can be completed in a very short time even under such a large threshold, and the remaining candidate spaces of FTC and GCA are infeasible for exhaustive search.

It can be seen from Fig. 10 and Fig. 12 that the P_r of FTC, GCA and RET-FCA are almost the same. P_r of FCA under k = 72 is not even comparable to that of RET-FCA under k = 32 in Fig. 10, and P_r of FCA under d = 72 is also smaller than that of RET-FCA under d = 32 in Fig. 12. This indicates that RET-FCA significantly improves the performance of FCA. In addition to the influence of the distributions of sub-key values and collision values, the number of collisions allowed to be out of threshold d and the number of error-tolerant sub-keys also affect the experimental results. In order to further improve the P_r of RET-FCA, they can be further increased. The former reflects the tradeoff between error-tolerant time and exhaustive time. If it is large, it is more likely that the correct collision values will fall within d. This also makes more erroneous

keys fall within d, which increases the difficulty of the error-tolerant computation and the subsequent key verification.

6.3 Experiments on Different Numbers of Measurements

We also compare the performance of FTC, GCA, FCA and RET-FCA under different numbers of measurements N. Here both k and d are set to 64. The number of remaining key candidates of FTC and GCA under different thresholds is densely distributed (as shown in Fig. 9 and Fig. 11). The "diluted" dense distributions can also be seen in Fig. 13. They increase slightly with the growth of N, but the density does not increase so obviously as the ones under different thresholds k and d. However, the distribution of n corresponding to FCA is relatively scattered, just like the one of GCA under small thresholds discussed in [22]. With the increase of N, the remaining key candidate space of RET-FCA gradually becomes dense in $2^{10} \sim 2^{15}$, and this phenomenon is obvious when $N > 100$. Moreover, the increase of N also makes more correct sub-key values and collision values fall within the thresholds, and the number of symbol "+" grows in Fig. 13. However, due to the limited collision information used by FTC and GCA, a large number of candidate keys satisfy the given collision conditions, and the density of them does not change significantly.

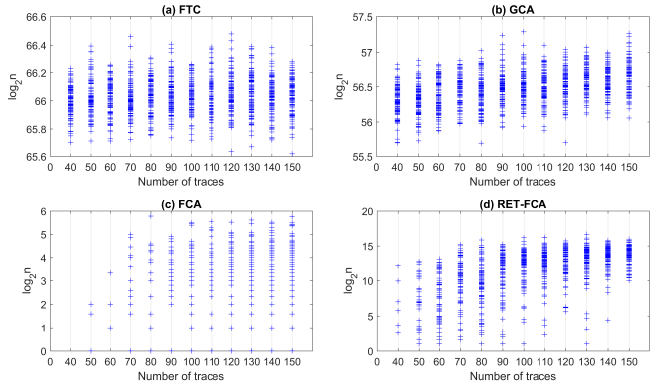

Fig. 13. The remaining key candidates under different numbers of measurements.

Compared with the growth of thresholds k and d, increasing the number of measurements used in each repetition can make the sub-keys and the corresponding collision values fall within the thresholds quickly. When the number of measurements ranges from 40 to 150, the P_r of FTC, GCA and RET-FCA ranges from 0 to about 1, compared to from 0 to about 0.6 of FCA. It can also be seen from Fig. 14 (a) that the "gap" of P_r between RET-FCA and FCA increases

significantly. This indicates that more collisions fall within the thresholds k and d, while empty Θ still occurs under FCA, which illustrates the effectiveness of error tolerance in RET-FCA. There are many long PFCCs in RET-FCA that can still be computed, but under FCA, a null set may appear under the previous several sub-keys. This is also one reason why the runtime of FCA is much smaller than that of RET-FCA in Fig. 10, Fig. 12 and Fig. 14. In principle, more collisions mean that the time consumed by RET-FCA increases, which is a normal phenomenon. The runtime of FTC and FCA is almost unchanged when the thresholds k and d are fixed, while slight changes in GCA, and RET-FCA increase significantly in Fig. 14(b). RET-FCA takes about 40 seconds to 160 seconds under 40 to 150 measurements. With the increase number of measurements, the runtime gap between FCA and RET-FCA increases significantly under the same threshold. A small increase in the number of remaining sub-key combinations in each of 120 rounds of error tolerance in RET-FCA will lead to a rapid growth of n in Θ. The satisfaction of collisions under error tolerant conditions significantly increases the runtime.

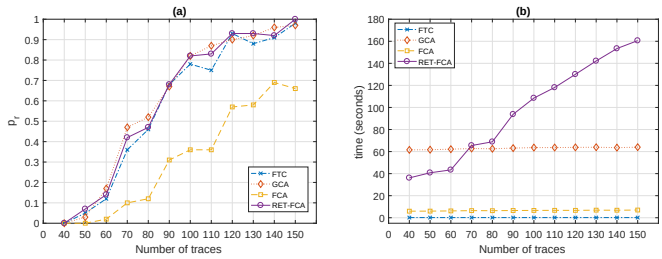

Fig. 14. The probability of correct full key falling within Θ (a) and average time consumption (b) under different numbers of measurements.

FCA deletes a large number of key candidates due to its strict collision conditions under the same k, d and N. Its P_r is much lower than RET-FCA's. However, FCA is much faster than RET-FCA (as shown in Fig. 10, Fig. 12 and Fig. 14). We can also take the outputs of CPA and CCA given in Section 3.4 as an example to illustrate the efficiency of FCA. The thresholds k and d considered can be much larger than 72 as discussed in this section. In fact, when we set both k and d to 96, we are faced with very large key candidate space and collision space of up to $2^{105.3594}$. However, FCA only needs 4207.4 seconds to complete its execution. Only 737.2 seconds and 999.1 seconds are needed when k and d are set to 80 and 96, and 96 and 80. In this case, the remaining key spaces are 87 and 201, respectively. As such, FCA has a very strong search ability and can reduce the same search space to the smallest one compared with FTC, GCA and RET-FCA, which demonstrates the benefits of FCA. This conclusion can also be drawn from Fig. 9 to Fig. 11. Therefore, we can first use FCA to perform a

deep search. If the key is not recovered, we can then deploy RET-FCA. Since FCA is time-consuming, we can use RET-FCA for fault tolerance under lower thresholds than FCA, which is also the original intention of RET-FCA.

7 Conclusion

Several existing methods named TC, FTC and GCA have attempted to recover the key from large search spaces far beyond exhaustive capabilities, and practical schemes are given in FTC and GCA. However, the collision information used by the existing methods are very limited. In particular, when large thresholds are used, these methods result in very large key spaces that makes exhaustive search infeasible. In order to make full use of all collision information, we propose Full Collision Attack (FCA) in this paper, which is very simple and efficient. FCA can search much larger candidate spaces than FTC and GCA. However, FCA requires that collision values fall within the thresholds. We further propose an error-tolerant scheme RET-FCA based on the phenomenon that collisions outside the thresholds are only related to several sub-keys. This significantly reduce the thresholds used in FCA and improve the performance. The intractable key space for FTC and GCA can be easily conquered by FCA and RET-FCA. The experimental results demonstrate the superiority of our scheme. Hence, FCA and RET-FCA provide new directions by pushing the limits of exhaustible key search space. This paper only considers error tolerance for collision values, and leaves the optimization of RET-FCA and error tolerance for sub-keys as open problems.

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. 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.
6. L. David and A. Wool. A bounded-space near-optimal key enumeration algorithm for multi-subkey side-channel attacks. In *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, pages 311–327, 2017.
7. 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.

8. F. Durvaux and F. Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 240–262, 2016.
9. T. Espitau, P. Fouque, B. Gérard, and M. Tibouchi. Side-channel attacks on BLISS 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. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
18. 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.
19. A. Moradi, S. Guilley, and A. Heuser. Detecting hidden leakages. In *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, pages 324–342, 2014.
20. 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.

21. M. Nassar, Y. Souissi, S. Guilley, and J. Danger. RSM: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1173–1178, 2012.
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. 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.
25. C. Rechberger and E. Oswald. Practical template attacks. In *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, pages 440–456, 2004.
26. 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.
27. 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.
28. 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.
29. N. Veyrat-Charvillon and F. Standaert. Mutual information analysis: How, when and why? In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 429–443, 2009.
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. 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.
33. F. Zhang, L. Geng, J. Shen, S. Bhasin, X. Zhao, and S. Guo. Improved low-entropy masking scheme for LED with mitigation against correlation-enhanced collision attacks. In *2017 Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2017, Beijing, China, October 19-20, 2017*, pages 49–54, 2017.

34. H. Zhang. How to effectively decrease the resource requirement in template attack? In *Advances in Information and Computer Security - 9th International Workshop on Security, IWSEC 2014, Hirosaki, Japan, August 27-29, 2014. Proceedings*, pages 119–133, 2014.