The Art of Guessing in Combined Side-Channel Collision Attacks

Changhai Ou, Siew-Kei Lam and Guiyuan Jiang

School of Computer Science and Engineering, Nanyang Technological University,
chou@ntu.edu.sg,assklam@ntu.edu.sg,gyjiang@ntu.edu.sg

Abstract. Recent combined collision attacks have shown promising results for exploiting side-channel leakage information from both divide-and-conquer and analytical distinguishers. However, divide-and-conquer distinguishers used such as Correlation Power Analysis (CPA) cannot directly provide the success probability of attack which impedes effective threshold setting for determining the candidate space. In particular, they uniformly demarcate the thresholds for all sub-keys, which restricts the candidate space that is able to be analyzed and increases the attack difficulty. Moreover, the existing works mainly focus on improving collision detection algorithms, and lacks theoretical basis. Finally, the inadequate use of collision information and backward fault-tolerant mechanism of existing schemes lead to low attack efficiency. To overcome these problems, this work first introduces guessing theory into Template Attack (TA) to facilitate the estimation of success probability and the corresponding complexity of key recovery. We also extend Multiple-Differential Collision Attack (MDCA) to a new combined collision attack named Multiple-Differential Combined Collision Filter (MDCCF), which achieves the multiple-differential voting mechanism via two levels: Distinguisher Voting (DV) and Collision Voting (CV). DV exploits the information from CPA, TA and Correlation enhanced Collision Attack (CCA) to filter the candidates of TA that fall within a threshold. CV further applies differential voting on the selected sub-keys with the smallest number of candidates to vote other sub-keys. The experimental results show that the proposed MDCCF significantly improves key ranking, reduces the candidate space and lowers the complexity of collision detection, without compromising on the success probability of attacks.

Keywords: MDCCF · distinguisher voting · collision voting · combined collision attack · candidate space · collision attack · side-channel attack

1 Introduction

Implementations of cryptographic algorithms on devices produce unintentional leakages such as power consumption [18], electromagnetic radiation [2] and cache patterns [30], which pose security vulnerabilities to Side-Channel Attacks (SCA). SCA has been demonstrated successfully on various chips and devices, such as SIM cards [19], PDAs [14], desktop computers [15] and even cloud servers [17]. Existing methods for SCA can be divided into two general approaches: divide-and-conquer and analytical. These approaches can exploit two types of information: direct leakages and collision leakages.

Existing divide-and-conquer distinguishers, such as Differential Power Analysis (DPA) [18], Correlation Power Analysis (CPA) [9] and Template Attack (TA) [10, 25], divide the huge candidate key space into several small blocks and conquer them one by one. They are often combined with key enumeration [24, 12] to avoid unnecessary guesses, but this is only feasible if the keys fall within the enumerable space. In practice, keys often locate in spaces that cannot be enumerated directly. On the other hand, analytical attacks such as algebraic attack [26] and collision attack [27, 3], exploit more leakage information than divide-and-conquer distinguisher, but are harder to mount.

In order to leverage the benefits of the two approaches, combined collision attacks have been proposed. These attacks combine a divide-and-conquer distinguisher and an analytical collision distinguisher to exploit more leakage information and construct the relationship between different sub-keys. They rely on the principle wherein several weak distinguishers can construct a stronger distinguisher. Here we take voting, one of the simplest differential methods, to illustrate this principle. For a set of testing samples, each weak distinguisher obtains a set of classification results. Majority voting is then used to determine their final classes, which is more accurate than that of a single distinguisher. This paper takes one step further by extending the Multiple-Differential Collision Attack (MDCA) to a combined collision attack and introducing guessing theory to enhance it. Before introducing our contributions, we will introduce the related works in the next sub-section.

1.1 Related Works

Existing combined collision attacks combine a divide-and-conquer distinguisher (e.g. CPA) and an analytical collision distinguisher (e.g. Correlation enhanced Collision Attack (CCA) [20]) to construct the relationship between different sub-keys and exploit more leakage information. Each distinguisher can delimit the range (i.e. threshold) of candidates to be considered based on the guessing conditions. For example, the thresholds τ_k and τ_d of CPA and CCA indicate that only the first τ_k and τ_d candidates of these two distinguishers are considered. The undesirable candidates that do not satisfy the given collision conditions are discarded. In other words, only desirable candidates satisfying the given collision conditions are remained. In this way, the original candidate space can be significantly reduced to lower the attack difficulty.

Combined collision attack was first proposed in [5] where a scheme called Test of Chain (TC) was presented. TC tries to find a long chain from the first sub-key to the last sub-key within the threshold τ_k. However, how to implement TC was not discussed. The first practical combined collision attack was proposed in [29] considering the combination of CPA and CCA, and $\tau_d = 1$. This method, called Fault Tolerant Chain (FTC), aims to find the collisions between the first sub-key and the other 15 sub-keys, and exhaust the first sub-key. Due to the insufficient usage of collision information, FTC needs to enumerate a large number of candidates.

MDCA [4] uses multiple thresholds to conduct differential voting when matching the power traces of two intermediate values (e.g., binary voting and ternary voting based on Euclidean distance between them). Obviously, MDCA here is not a combined collision attack and hence, it is not the focus of this paper. The work in [22] considers $\tau_d > 1$ using a combined collision attack called Group Verification based Multiple-Differential Collision Attack (GV-MDCA). Unlike MDCA, the differential voting mechanism of GV-MDCA is performed on the number of collisions, not on the matching degree of power traces of two intermediate values like Euclidean distance. It votes a sub-key using the collisions between its candidates and the candidates of the remaining 15 sub-keys (i.e., the groups). The candidates are then ranked in descending order according to the number of votes. GV-MDCA makes more sub-keys rank first, thus improving CPA's efficiency. Although GV-MDCA can be regarded as a kind of combined collision attack, it does not alter the rank of the key due to the unchanged probabilities or scores of the candidates.

A flaw of FTC caused by insufficiently utilized collision information was also shown in [21] wherein if the i-th sub-key k_i and the collision value were both wrongly guessed, and the two candidates from k_1 and k_i still constituted a collision, then FTC would regard them as desirable candidates. The larger τ_k and τ_d are, the more frequently this situation occurs. To solve this problem in FTC, Group Collision Attack (GCA) [21] was further proposed. The huge candidate space was divided into several big groups (e.g., 16 sub-keys of AES-128 are divided into 4 non-overlapping big groups). Intra-group collision is used to perform the first round of undesirable candidates deletion within groups, and inter-group verification is used to perform another round of deletion among groups. GCA exploits collisions about three times more than FTC.

Both FTC and GCA combine CPA with CCA. CPA is exploited to determine the ranks of sub-keys, and CCA further filters possible collisions from them. This contributes to a significant reduction of the candidate space, thus decreasing the difficulty of attack. In this way, GCA exploits more collision information and further extends FTC's conquerable space. However, GCA still encounters great difficulty when dealing with larger space (see Section 7 for details). Moreover, the correlation coefficients produced by CPA do not provide a reasonable reference for the delimitation of threshold τ_k. To find a good threshold, we need to constantly test and adjust it. Finally, FTC and GCA set a unified threshold τ_k for all sub-keys in CPA. In order to ensure that all sub-keys are within the threshold, τ_k should be greater than or equal to the one with the deepest position. Obviously, a large number of additional undesirable candidates need to be considered, which increases the complexity of key recovery.

1.2 Our Contributions

The main contributions of this paper are as follows:

- We introduce guessing theory into TA to optimize the setting of threshold τ_k, which aids in rapid estimation of the success probability and the corresponding guessing space. This helps to optimize the choice of τ_k based on the available computing power, hence increasing the efficiency of attacks and evaluations.

- We extend the traditional analytical distinguisher MDCA to a combined collision attack, and propose a two-level Multiple-Differential Combined Collision Filter (MDCCF). In the first level, the information from CPA, TA and CCA is extracted to build a multiple-differential voting mechanism named Distinguisher Voting (DV). DV can be further optimized by voting on candidate values of TA using the ones from CPA and the utilization of collision information in CCA is left behind MDCCF.

- In the second level of MDCCF, we select several sub-keys with the smallest number of candidates to vote other sub-keys, thus building a second differential voting mechanism on collisions with fault tolerance named Collision Voting (CV). CV further eliminates some undesirable candidates and narrows the search space, which significantly lowers down the complexity of attacks.

A good combined collision attack should reduce candidate space as much as possible without significantly reducing the probability of successful key recovery. The proposed two-level MDCCF with fault-tolerant mechanism obviously can achieve this. Its success probability is close to the theoretical one under a very limited number of measurements if the collision conditions are set reasonably to make most collisions fall within thresholds and consider fault tolerance on others. This is significantly higher than the success probability of both FTC and GCA.

1.3 Organization

The rest of the paper is organized as follows. The experimental setup, principles of TA and collision attack, and the existing combined collision attacks are introduced in Section 2. Mathematical metrics of guessing theory, including guessing model under TA, marginal guesswork and partial guessing metrics, are given in Section 3. DV and CV, the two levels of our MDCCF, are introduced in detail in Sections 4 and 5. Trade-off considerations to balance success probability and complexity is given in Section 6. Experiments are performed on an *AT89S52* micro-controller in Section 7 to demonstrate the practicability and efficiency of MDCCF. Finally, Section 8 concludes this paper.

2 Preliminaries

2.1 Experimental Setup

Our experiments are performed on the power traces leaked from an AT89S52 micro-controller. The operating frequency of the AT89S52 system is 12 MHz, the shortest instructions take 12 clock cycles to execute. We use assembly language to implement the AES-128 algorithm, and the instruction "**MOVC A, @A+DPTR**" is utilized to complete S-box operation, which requires 24 clock cycles. The DPTR register saves the starting address of the S-box, register A saves the offset, and the output of the look-up table operation is saved back in register A. The sampling rate of our Picoscope 3000 is set to 125 MS/s. We finally acquire 51200 power traces and use CPA to select a Point-of-interest (POI) [13] with the highest correlation coefficient to perform the subsequent experiments.

2.2 Template Attack

Template Attack (TA) [10, 25, 16] is one of the most powerful side-channel distinguishers. It can be divided into two stages: template construction and template classification. Let $|\mathcal{N}|$ denote the number of sub-keys and $|\mathcal{K}_j|$ denote the candidate space of the j-th sub-key, $\mathbf{t}_j^i\left[1\cdots n\right]$ $\left(1 \leq j \leq |\mathcal{N}|, 1 \leq i \leq |\mathcal{K}_j|\right)$ denote a total of n power traces used to profile the i-th intermediate value of the j-th sub-key $\mathbf{I}_j^i$ (e.g. the S-box output in the first round of AES-128), and $\mathbf{m}_j^i$, $\mathbf{C}_j^i$ denote the corresponding mean power consumption vector and the noise covariance matrix, which constitute a template $\left(\mathbf{m}_j^i, \mathbf{C}_j^i\right)$. Here

$$\mathbf{m}_j^i = \frac{1}{n}\sum_{\kappa=1}^{n} \mathbf{t}_j^i\left[\kappa\right] \tag{1}$$

and

$$\mathbf{C}_j^i = \frac{1}{n}\sum_{\kappa=1}^{n} \left(\mathbf{t}_j^i\left[\kappa\right] - \mathbf{m}_j^i\right)\left(\mathbf{t}_j^i\left[\kappa\right] - \mathbf{m}_j^i\right)^T, \tag{2}$$

and symbol "T" denotes matrix transposition. For a power trace $\mathbf{t}$ used for attack, the probability of it corresponding to template $\left(\mathbf{m}_j^i, \mathbf{C}_j^i\right)$ is:

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

where $\left|m_j^i\right|$ represents the length of $\mathbf{m}_j^i$, i.e., the number of POIs used to profile the templates.

2.3 Collision Attacks

We consider AES-128 in this paper, which performs 16 parallel S-box operations in its first round. Let $x_{j_1} \in \mathbb{F}_{2^8}$ and $k_{j_1} \in \mathbb{F}_{2^8}$ denote the j_1-th byte of a plaintext to be encrypted and the j_1-th sub-key, a linear collision happens if two S-boxes generate the same intermediate values:

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

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

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

if the plaintexts are known (in fact, plaintexts are usually assumed to be known in SCA). For simplicity, we use $k_{j_1} \leftrightarrow k_{j_2}$ to represent this pair of collision.

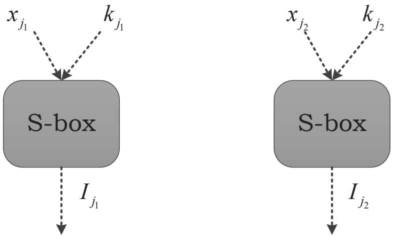

Figure 1: A pair of collision between two S-boxes in AES-128.

A specific implementation of CCA was given in Algorithm 2 in [29]. Taking the collision between the first and second S-boxes of AES-128 as an example, CCA divides the measurements of the first and second bytes into 256 classes according to their plaintexts, calculates the mean power consumption vector of each class, and computes the correlation coefficient

$$\rho\left\{\left(\mathbf{m}_1^{x_1}, \mathbf{m}_2^{x_1 \oplus \delta_{1,2}}\right) \middle| x_1 = 0, 1, 2, \cdots, 255\right\} \tag{6}$$

under a guess $\delta_{1,2}$ (see Eq. 5).

2.4 Combined Collision Attacks

A combined collision attack named Test of Chain (TC) in [5] tried to find a long chain from the first sub-key to the 16-th sub-key including 15 pairs of collisions. TC opened up a new direction of combined attacks, but unfortunately there was no feasible solution given. Wang et al. gave the first practical scheme named Fault Tolerant Chain (FTC) in [29], which combined CPA and CCA, and tried to find the collisions between the first sub-key and the other 15 sub-keys (as shown in Fig. 2). Here two thresholds τ_k and τ_d are for CPA and CCA respectively, ξ_{j_1} denotes the candidates of the j_1-th sub-key sorted in descending order according to their probabilities or scores, and $\xi_{j_1}^i$ denotes its i-th candidate. If $\tau_k = 10(\tau_d = 10)$, only first 10 candidates for each sub-key (collision δ_{j_1,j_2}) are considered. However, guessing all the sub-keys together is not conducive to making full use of collision information. Moreover, the candidate space is too large, which dramatically weakens the advantage of divide-and-conquer.

Group Collision Attack [21] improves FTC by dividing the 16 sub-keys of AES-128 into 4 big non-overlapping groups. The sub-key combinations within each group that do not meet the given collision conditions are discarded, and the candidate space is further reduced by inter-group verification. There are $3 \sim 4$ pairs of collisions on each sub-key. Verification is essentially a kind of collision. In order to reduce collision detection, GCA continuously verifies long chains from short chains (see Example 1).

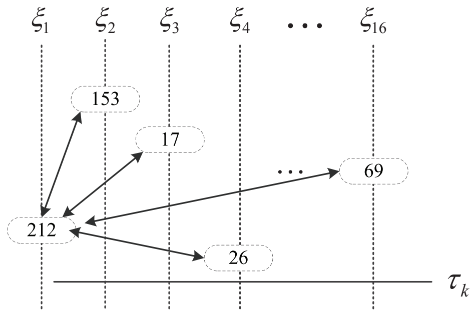

Figure 2: Fault Tolerant Chain (FTC).

Example 1. Take the three C_4 chains (see Section III-A in [21]) including 3 pairs of collisions among 4 sub-keys from the 1-st ~ 4-th, 3-rd~6-th and 5-th ~8-th sub-keys of an experimental output in Fig. 3 as an example, 17 ↔ 242 and 192 ↔ 24 in 17 ↔ 242 ↔ 192 ↔ 24 are also in 212 ↔ 153 ↔ 17 ↔ 242 and 192 ↔ 24 ↔ 229 ↔ 126. In this way, 17 ↔ 242 ↔ 192 ↔ 24 is verified and regarded as a desirable candidate.

It is noteworthy that the verified chain 17 ↔ 242 ↔ 192 ↔ 24 in Example 1 does not mean that the 4 candidates on it are correct. In fact, the 4-th sub-key k_4 is 9. Inter-group verification greatly alleviates the growth of guessing space after the candidate recombination from the 4 groups.

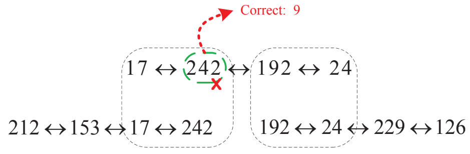

Figure 3: Key verification in GCA.

3 Mathematical Metrics of Guessing Difficulty

3.1 Guessing Model

Considering a sub-key k_j, we wish to evaluate the efficiency of an attacker trying to recover it given access to an oracle for queries "is $\xi_j^i = k_j$?" Actually, Eq. 3 gives the probability of a new measurement $\mathbf{t}$ corresponding to template $\left(\mathbf{m}_j^i, \mathbf{C}_j^i\right)$ (i.e. the probability that the sub-key $k_j = \xi_j^i$). We normalize and rank the probabilities in descending order, and obtain $p_j = \left\{p_j^1, p_j^2, \ldots, p_j^{|\mathcal{K}_j|}\right\}$ satisfying

$$\sum_{i=1}^{|\mathcal{K}_j|} p_j^i = 1 \tag{7}$$

and

$$p_j^1 > p_j^2 > \ldots > p_j^{|\mathcal{K}_j|} \tag{8}$$

as introduced in [11], where p_j^i is the i-th largest probability. The optimal brute-force attack [11] can be performed if the candidates are enumerated in this order. It represents the minimum number of candidates that an attacker needs to guess in order to recover the sub-key.

3.2 Marginal Guessworks

The widely used Guessing Entropy (GE) [28] in SCA provides the expected position of a sub-key. The expected number of guesses required by an attacker to find k_j in a guess was defined as guesswork or guessing entropy by Pliam in [23]:

$$\mathcal{G}\left(\mathcal{K}_j\right) = \sum_{i=1}^{|\mathcal{K}_j|} i \cdot p_j^i, \tag{9}$$

which is the optimal brute-force search. Choudary et al. named this guesswork as Massey's guessing entropy (GM) in [11], and proved its superiority compared to GE in evaluation. The theoretical boundaries of the average number of guesses have also been deeply discussed in [6, 8]. These techniques can be extended to TA to evaluate the difficulty of recovering a sub-key.

3.3 Partial Guessing Metrics

As stated in [6], guesswork and entropy metrics failed to model the tendency of real-world attackers to cease guessing against the most difficult cases. For example, they may choose a small part of candidates with high probabilities to guess and discard most candidates with low probabilities in TA (e.g. the first 2^{40} full-key candidates with the greatest probabilities from the total space 2^{128} of AES-128). This often happens when the computing power is limited. A typical example is key enumeration.

Suppose that an attacker tries to recover the sub-key k_j only from its first β candidates, the probability of success in an attack is

$$\lambda_\beta\left(\mathcal{K}_j\right) = \sum_{i=1}^{\beta} p_j^i, \tag{10}$$

which was defined as *β-success-rate* in [7]. The "success rate" here is actually the theoretical expectation of success. It is different from the one defined by Standaert et al. in [28], which is the proportion of the number of successful attacks to the total number of experiments in reality. Another metric considers only the candidates that satisfy the given probability value $\alpha \in (0, 1)$:

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

which is named as α-$work$-$factor$ in [23]. These two typical partial guessing metrics can be used in SCA, and their superiority will be fully demonstrated in the next sections.

4 Distinguisher Voting

Several weak distinguishers can be combined to construct a stronger distinguisher to exploit more leakage information. Here we combine CPA, TA and CCA to complete Distinguisher Voting (DV), the first level of our MDCCF. To comply with rules of combined collision attacks, we use CPA+CCA and TA+CCA to select candidates of TA, with the aim recovering the key from the collision space of TA+CCA (i.e. the remaining candidate space after combined collision attacks). Specifically, our target is simply to divided the 16 sub-keys of AES-128 into 8 big blocks, on which a multi-differential collision voting mechanism is further established to reduce the key search space. This can be undertaken after completing DV and CV, since voting on sub-keys is more effective than voting on collisions.

4.1 Distinguisher Selection

As stated in [21], the combination of several distinguishers with close performance is more meaningful. If one has much better performance than others, SCA can be performed on it directly without considering others. We use Amplified Template Attack in [31] to improve the efficiency of TA. In this case, the probability of a candidate ξ_j^i to be the correct one, is the product of the probabilities that all n power traces are classified into their corresponding templates according to it, namely $p_j^i = \prod_{\kappa=1}^{n} p_j^i\left[\kappa\right]$.

Since all probabilities are smaller than 1, p_j^i decreases with the growth of the number of traces. If there are too many power traces, the probability products may be too small to be expressed. Here we use logarithmic function to solve this problem:

$$\ln p_j^i = \ln \prod_{\kappa=1}^{n} p_j^i\left[\kappa\right] = \sum_{\kappa=1}^{n} \ln p_j^i\left[\kappa\right]. \tag{12}$$

In this case, $p_j^i = \exp\left(\sum_{\kappa=1}^{n} \ln p_j^i\left[\kappa\right]\right)$. All probabilities then minus the maximum value of $\ln p_j$ denoted by $\ln p_j^{max} = \max\left(\ln p_j^i \middle| i = 1,\ldots,|\mathcal{K}_j|\right)$, since

$$\frac{p_j^i}{p_j^{max}} = \frac{\exp\left(\ln p_j^i\right)}{\exp\left(\ln p_j^{max}\right)} = \exp\left(\ln p_j^i - \ln p_j^{max}\right). \tag{13}$$

We then normalize them. With the increase number of measurements, the success probability by guessing the first several candidates of each sub-key increases. We randomly encrypt 25600 plaintexts for templates construction. The average success probability of first sub-key under different numbers of guesses when 160 $\sim$ 360 power traces are considered in each of 200 repetitions is shown in Fig. 4.

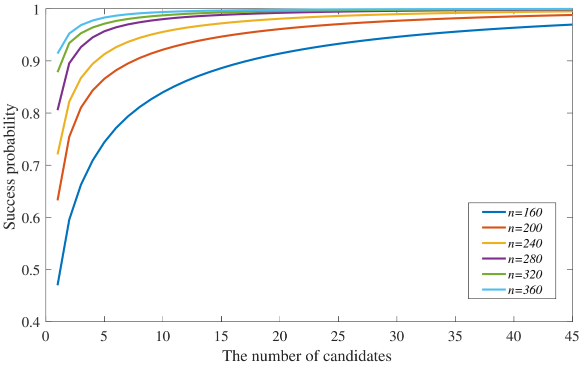

Figure 4: The average success probability of first sub-key under different numbers of guesses.

The performance of CPA and TA is much better than that of CCA. It's not surprising since CCA's classic use is to attack flawed masking schemes (e.g., DPA *contest* v4.1 [1]).

We use 4^{th}-order Minkowski distance instead of correlation coefficient in Eq. 6 when classifying each power trace, which achieves good results in our experiments.

4.2 Voting

As different distinguishers have different classification performance, this makes the rank of candidates different when dealing with the same measurements. Let $\xi_{j_1}^{i_1}$ and $\xi_{j_2}^{i_2}$ denote the i_1-th and i_2-th candidates of the j_1-th and the j_2-th sub-keys, $\delta_{j_1,j_2}^{\mathsf{CPA}}$ and $\delta_{j_1,j_2}^{\mathsf{TA}}$ denote the candidates of collision between them in CPA+CCA and TA+CCA. Here we define a simple Distinguisher Voting (DV) mechanism:

$$\Psi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right) = \begin{cases} 1, & \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \left(\delta_{j_1,j_2}^{\mathsf{CPA}} \bigcap \delta_{j_1,j_2}^{\mathsf{TA}}\right) \\ 0, & otherwise. \end{cases} \tag{14}$$

This means, if a collision $\xi_{j_1}^{i_1} \leftrightarrow \xi_{j_2}^{i_2}$ satisfies $\xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \delta_{j_1,j_2}^{\mathsf{TA}}$ and $\xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \delta_{j_1,j_2}^{\mathsf{CPA}}$, we keep it, otherwise discard it. In this case, we filter a part of collisions in TA+CCA using the collisions from CPA+CCA, which further optimizes the guessing space.

Example 2. Taking an experimental result under 300 measurements, $\alpha = 0.95$ and $\tau_d = 70$ as an example, TA+ CCA and CPA+CCA have 13 and 15 pairs of collisions between the first two sub-keys respectively (as shown in Fig. 5). The threshold of CPA is the same as TA. The long red arrows in Fig. 5 represent the common collisions between CPA+CCA and TA+CCA. Since $\tau_d > 1$, each candidate of a sub-key can have at most τ_d pairs of collisions. For example, the candidate 166 has 4 pairs of collisions $166 \leftrightarrow 153$, $166 \leftrightarrow 113$, $166 \leftrightarrow 116$ and $166 \leftrightarrow 83$ on TA+CCA, and only has two pairs of collisions $166 \leftrightarrow 153$ and $166 \leftrightarrow 28$ on CPA+CCA. Only a common collision $166 \leftrightarrow 153$ satisfies the collision conditions given in Eq. 14. Therefore, 3 other collisions are discarded, 166 and 153 are the remaining candidate of the first and second sub-keys respectively.

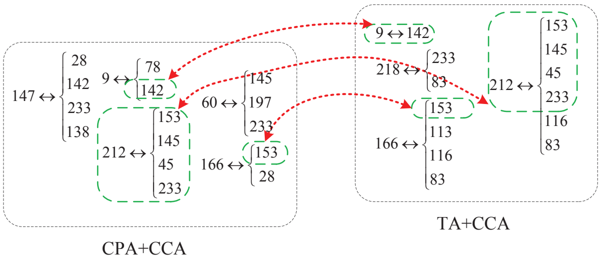

Figure 5: Distinguisher voting between the first two S-boxes of AES-128 considering CPA+CCA and TA+CCA.

4.3 Relaxation Factor

Since CPA is weaker than TA, the expected ranks of its sub-keys should be deeper than TA. Therefore, we need to set an independent threshold greater than μ_α for it to ensure that the corresponding sub-key falls into the space under consideration. We define this threshold as $\pi \cdot \mu_\alpha$ and the coefficient $\pi\ (\geq 1.0)$ as relaxation factor. For example, if we set $\pi = 1.2$, only 6 pairs of collisions are in both collision sets, and other 7 pairs of collisions of TA+CCA are filtered out (see Fig. 5). The parameter π should be carefully determined.

Larger π leads to looser constraints, and fewer collisions of TA+CCA would be filtered out. The complexity of our DV to filter collisions between two sub-keys k_{j_1} and k_{j_2} is only $\mathcal{O}(\mu_\alpha(\mathcal{K}_{j_1}) \cdot \mu_\alpha(\mathcal{K}_{j_2}))$, which does not bring too much computation to the attacks.

To maintain a candidate of TA, Eq. 14 requires a pair of collision in both CPA+CCA and TA+CCA. This condition is stringent and may delete sub-keys in some cases. For example, when n is large, the number of candidates of a sub-key under TA and fixed μ_α is often very small. If the threshold of CPA (i.e. $\pi \cdot \mu_\alpha$) is not set sufficiently large, it is easy to mistakenly delete the sub-keys. On the other hand, if π is set very large and n is small, both TA and CPA contain a large number of unfiltered candidates, which affects the computation efficiency. The reasonable settings of π will be discussed in detail in Section 7.1.

4.4 Optimization

The collision threshold τ_d does not change in the combined collision attacks TA+CCA and CPA+CCA in DV, the first level of MDCCF. This means that the collision information we get from CCA does not change. Therefore, we can improve MDCCF by optimizing the voting mechanism performed on TA and CPA, and performing collision detection on the remaining candidate space. In this case, let ξ_j^{TA} and ξ_j^{CPA} denote the ranked candidates of k_j output by TA and CPA, Eq. 14 can be further optimized as

$$\left(\xi_j^{\mathsf{TA}}\right)' = \xi_j^{\mathsf{TA}} \bigcap \xi_j^{\mathsf{CPA}}. \tag{15}$$

Obviously, combining the collision information in CCA with the remaining candidates of TA can avoid a lot of unnecessary collision detection.

Example 3. The intersection of CPA+CCA and TA+CCA collisions in Fig. 5 can be further simplified as the collisions between CCA and the remaining candidates $\{9, 212, 166\}$ and $\{142, 233, 153, 45, 145\}$, which is the intersection of CPA and TA. In this way, unnecessary collision detection between 218 and 83, 60 and 145, 147 and 28 etc., can be avoided.

It is noteworthy that fault tolerance can also be decided according to the probabilities of candidates, and the larger the probabilities, the better fault-tolerant mechanisms are required. In other words, the smaller the probability of a candidate, the lower requirements on fault tolerance. In this case, a new optimized DV mechanism is given as:

$$\Psi\left(\xi_j^i\right) = \begin{cases} 1, & p_j^i \geq 0.1 \\ 1, & \xi_j^i \in \xi_j^{\mathsf{CPA}} \ \& \ p_j^i \in \left(10^{-5}, 0.1\right) \\ 0, & otherwise. \end{cases} \tag{16}$$

The above Eq. 16 means that, for each candidate ξ_j^i ($i \in [1, \mu_\alpha(\mathcal{K}_j)]$) of a sub-key k_j whose probability p_j^i in TA is greater than 0.1, it can be maintained directly. If p_j^i is between 10^{-5} and 0.1, and ξ_j^i is a candidate in CPA, it will be maintained. These strategies are designed to reduce the relaxation factor π and the loss caused by mistakenly deleting the sub-keys, while not considerably reducing the success probability of key recovery.

It is difficult to guarantee the key recovery if it is in a very deep space. Due to the limitation of computing power, we can only selectively guess a small part of key candidates using key enumeration. Collision detection is performed on all candidates within the threshold. For the case that when the probability threshold α is set to a large value which leads to a large volume of candidates, we can discard a lot of candidates if their probabilities are too small. This is because they will not significantly improve the success probability of attacks, but instead drastically increase the workload of collision detection. This optimization is embodied in Eq. 16, i.e., a candidate is discarded if its probability is smaller than 10^{-5}.

5 Collision Voting

DV, the first level of MDCCF, can be optimized into a simple collision-free voting mechanism (see Eq. 15 and Eq. 16). However, if 16 sub-keys are divided into 8 big group before Collision Voting (CV), it will inevitably vote a large number of collisions. Actually, this can also be done after CV. CV, the second level of our MDCCF, is mainly multiple-differential combined collision attack to enable fault tolerance, built on the lists with smallest number of candidates that we selected in Section 5.1. Its voting mechanism is also a fault-tolerant mechanism, and the candidates of each sub-key to be voted that satisfy the given collision conditions are retained.

5.1 Lists Selection

CV further filters the candidates by making full use of collisions information, and theoretically provide relaxed filtering conditions to reduce the possibility of mistakenly deleting sub-keys. Detecting all collisions among $|\mathcal{N}|$ sub-keys has a complexity of $\mathcal{O}\left(\prod_{j=1}^{|\mathcal{N}|}\mu_\alpha\left(\mathcal{K}_j\right)\right)$. This would be very time-consuming if there are too many sub-keys and a large number of collisions among them. Therefore, there is a need to optimize CV.

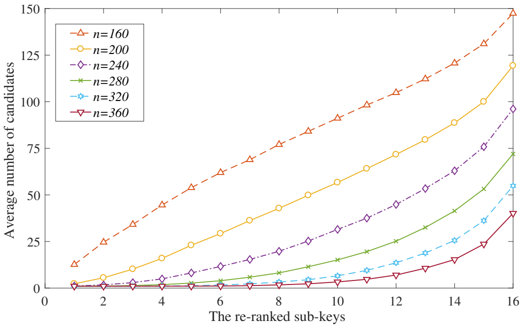

Figure 6: The average number of candidates of 16 re-ranked sub-keys.

We rank all the 16 lists of candidates of AES-128 in ascending order according to their α-$work$-$factor$, and evaluate their corresponding average number of guesses by repeating each evaluation 200 times and using 160 to 360 measurements in each repetition. In order to be consistent with the experiments in Section 7, we set α to 0.996, and obtain the results shown in Fig. 6. The average number of candidates and probability distribution of the re-ranked sub-keys vary greatly in TA. Moreover, with the increase number of measurements n, the average number of candidates of the 8 deepest sub-keys decreases significantly. However, the average number of candidates of the 5 shallowest sub-keys is less than 25 when $n \geq 200$. Obviously, it will be very efficient if we select these sub-keys with fewest candidates to establish a differential voting mechanism to vote the remaining sub-keys.

5.2 Fault Tolerance on Sub-keys

The sub-keys beyond threshold τ_k often happen on the ones with the lowest Signal-to-Noise Ratio (SNR) of their power traces, which illustrates the necessity of fault tolerance on them. Suppose that the n_α sub-keys with minimum number of candidates are selected to vote the remaining $16 - n_\alpha$ sub-keys. Since the probability distribution of each sub-key in TA is independent, the probability P_r that X sub-keys from these n_α sub-keys fall into the candidate space μ_α follows the Bernoulli distribution $X \sim \mathcal{B}\left(n_\alpha, \alpha\right)$:

$$P_r\left(X=\kappa\right)=\binom{n_\alpha}{\kappa}\alpha^\kappa\left(1-\alpha\right)^{n_\alpha-\kappa}. \tag{17}$$

Here $\kappa = 0, 1, 2, \ldots, n_\alpha$ and

$$\binom{n_\alpha}{\kappa}=\frac{n_\alpha!}{\kappa!\left(n_\alpha-\kappa\right)!}. \tag{18}$$

If we set n_α and α to 6 and 0.996, and a sub-key out of its threshold $\tau_k = \mu_\alpha$ is allowed, the success probability will reach $P_r\left(X = 0 \; or \; 1\right) = 0.996^6 + 6 \cdot \left(0.996^5\right) \cdot 0.004^1 = 0.9998$, compared to $P_r\left(X = 0\right) = 0.996^6 = 0.9762$. If $\alpha = 0.9$, $P_r\left(X = 0 \; or \; 1\right) = 0.8857$ and $P_r\left(X = 0\right) = 0.5314$, which shows significant improvements in success probability and illustrates the effectiveness of our fault-tolerant mechanism on CV.

Example 4. A simplified example is shown in Fig. 7, wherein a collision 153 ↔ 26 between k_2 and k_4 is used to vote the candidate 192 of k_5. That is, two sub-keys k_1 and k_3 are fault-tolerated, and only other two sub-keys k_2 and k_4 are used for voting. This requires a total of three pairs of collisions: one between these two sub-keys, and two between them and the sub-key k_5 to be voted.

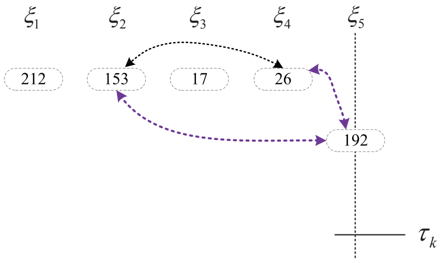

Figure 7: A collision 153 ↔ 26 between k_2 and k_4 is used to vote the candidate 192 of k_5.

If all the sub-keys and collision values are within the thresholds τ_k and τ_d, there will be a total of $\binom{n_\alpha+1}{2}$ pairs of collisions between each two of the n_α sub-keys used for voting and the sub-key waiting for voting. A large number of candidates will not satisfy this strict collision condition, which is also verified by our experiments. It is noteworthy that fault-tolerant mechanism DV voting on sub-keys is not simply reducing the number of collisions from $\binom{n_\alpha+1}{2}$, since they are not necessarily related to the fault-tolerated sub-keys. If σ sub-keys are out of threshold, all collisions associated with them should be fault-tolerated. In this case, a total of

$$\eta_{n_\alpha,\sigma}=\binom{n_\alpha+1}{2}-\binom{n_\alpha+1-\sigma}{2} \tag{19}$$

pairs of collisions are allowed to be out of thresholds τ_k and τ_d, and the differential threshold is set to $\tau_\sigma = \binom{n_\alpha+1-\sigma}{2}$.

Example 5. Taking $\sigma = 1$ and $\sigma = 2$ under $n_\alpha = 5$ for example, the threshold τ_σ is only 10 and 6 respectively. In other words, 10 (or 6) pairs of collisions between other 4 (or 3) sub-keys are required to collide with the sub-key to be voted.

It is worth mentioning that it would be better if the above fault-tolerant mechanism is only performed on the n_α sub-keys used for voting. If it is applied on the sub-keys to be voted, the exhaustion of a fault-tolerated sub-key will be very time-consuming and space-consuming. This also explains why Eq. 19 adopts $\binom{n_\alpha + 1 - \sigma}{2}$ as the differential threshold for the fault tolerance on the sub-keys used for voting, without considering the exhaustion of the fault-tolerated ones.

5.3 Fault Tolerance on Collisions

Whether a candidate is retained depends on the number of collisions required. The more pairs are required, the larger probability that it will be deleted. The fault tolerance on collisions reduces the number of required collisions, increases the probability of sub-keys falling into the remaining candidate space, but also enlarges it. GCA requires $3 \sim 4$ pairs of collisions occurring on each candidate without any fault-tolerant mechanism. If we use $n_\alpha = 4$ sub-keys to vote the remaining 12 sub-keys, and 2 pairs of collisions $212 \leftrightarrow 17$ and $153 \leftrightarrow 192$ are fault-tolerated, each candidate to be voted requires $2 \sim 5$ pairs of collisions, since we allow the fault-tolerated collisions to occur anywhere (see Fig. 8).

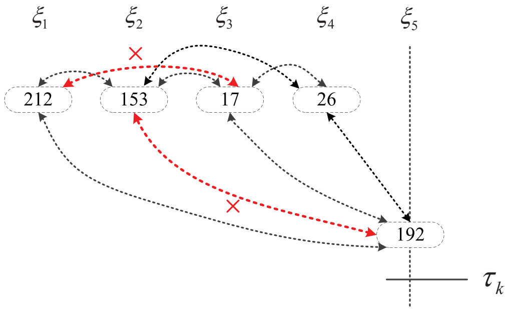

Figure 8: Two collisions out of threshold are tolerated in CV.

Here we use our CV, a new multiple-differential combined collision attack, to further establish a fault-tolerant mechanism. Specifically, let $\psi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right)$ denote the collision flag of $\xi_{j_1}^{i_1}$ and $\xi_{j_2}^{i_2}$, if the collision establishes, i.e., $\xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \delta_{j_1,j_2}^{\mathsf{TA}}$, this flag is set to 1. It satisfies

$$\psi\left(\xi_{j_1}^{i_1}, \xi_{j_2}^{i_2}\right) = \begin{cases} 1, & \xi_{j_1}^{i_1} \oplus \xi_{j_2}^{i_2} \in \delta_{j_1,j_2}^{\mathsf{TA}} \\ 0, & otherwise. \end{cases} \tag{20}$$

In order to reduce the repetitive detection of collisions, not only all possible combinations of the n_α sub-keys used for voting but also the total number of collisions corresponding to each combination should be calculated. Suppose that $\left(\xi_1^{i_1}, \xi_2^{i_2}, \ldots, \xi_{n_\alpha}^{i_{n_\alpha}}\right)$ is a combination, its total number of collisions is

$$\eta\left(\xi_1^{i_1}, \xi_2^{i_2}, \ldots, \xi_{n_\alpha}^{i_{n_\alpha}}\right) = \sum_{j_1=1}^{n_\alpha-1} \sum_{j_2=j_1+1}^{n_\alpha} \psi\left(\xi_{j_1}^{i_{j_1}}, \xi_{j_2}^{i_{j_2}}\right). \tag{21}$$

If σ pairs of collisions are allowed to be out of thresholds τ_k and τ_d, the multiple-differential threshold can be

$$\tau_\sigma = \binom{n_\alpha + 1}{2} - \sigma. \tag{22}$$

From the number of collisions point of view, this collision condition is much higher than fault tolerance on sub-keys (see Eq. 19). In fact, all combinations with σ collisions out of threshold can be discarded directly during collision detection, thus reducing the detection burden when considering the subsequent sub-keys.

It is not difficult to see from Figs. 7 and 8 that fault tolerance on sub-keys is different from that on collisions. For the former, the collisions between the fault-tolerated sub-keys and other sub-keys (including the one to be voted) are tolerated. However, for the latter, it's fault-tolerance is only for collisions, not sub-keys. Two collisions ($\sigma = 2$ and $\tau_\sigma = 10 - 2 = 8$) are fault-tolerated in Fig. 8, so these five candidates of $k_1 \sim k_5$ can still satisfy the collision conditions. If $\sigma = 0$ or $\sigma = 1$, this combination will be discarded. The selected sub-keys have the smallest number of candidates, we can appropriately increase α, which will not considerably increase the load of collision detection in CV but can significantly improve the reliability of voting. This fully illustrates the advantages of CV.

5.4 Collision Detection Algorithm

Fault tolerance on sub-keys and collisions requires collision detection. In principle, the performance of the algorithms wherein all collisions are not repeatedly detected would be better. However, it is difficult to achieve this goal in fault tolerance. We vote the remaining 10 sub-keys by using $n_\alpha = 6$ sub-keys with the fewest candidates, where one sub-key and an additional pair of collision are allowed to be beyond thresholds τ_k and τ_d. As introduced in Section 5.3, we first find collisions between these sub-keys used for voting and obtain all combinations with the number of collisions larger than the threshold τ_σ. We then detect the collisions between them and the remaining sub-keys. These two steps are given in Algorithms 1 and 2.

The inputs of Algorithm 1 include the number of sub-keys n_α used for voting, the re-ranked sub-keys $\xi = \left(\xi_1, \xi_2, \ldots, \xi_{|\mathcal{N}|}\right)$, and their corresponding number of candidates within the threshold $\mu_\alpha(\mathcal{K})$ $\left(\mathcal{K} = \left\{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_{|\mathcal{N}|}\right\}\right)$. The outputs include all combinations Θ of these n_α sub-keys satisfying the collision condition and the corresponding pairs of collisions η. We use $Q = \xi \setminus \xi_{j_2}$ and $\mathcal{K}' = \mathcal{K} \setminus \mathcal{K}_{j_2}$ to denote the removal of k_{j_2}, i.e. the fault tolerance on k_{j_2}, and use Ω to record the intermediate results of each round of fault-tolerance (Step 2). Ω is initialized by Q_1 (Step 3).

ϕ is used to record the number of collisions on each chain in Ω. For each sub-key added subsequently, we initialize ϕ' as a flag array with the same number of flags as the chains in Ω (expressed by $\mathsf{Row}(\Omega)$). We also initialize the fault-tolerant threshold $\tau_\sigma = \binom{j_1}{2} - 1$, since only a pair of collision is allowed to be out of τ_d except for the fault-tolerated sub-key k_{j_2} (Step 4). To achieve this goal, we need to detect collisions between the $j_1 - 1$ candidates in each row of Ω and candidates $Q_{j_1}^{i_1}$ within threshold $\mu_\alpha\left(\mathcal{K}'_{j_1}\right)$ (Step 7). For example, the s-th candidate of the q-th chain Ω_q^s collides with $Q_{j_1}^{i_1}$, the flag cn of the new chain $\left[\Omega_q, Q_{j_1}^{i_1}\right]$ including j_1 candidates will increase by 1. Each combination of the selected $n_\alpha = 6$ sub-keys should contain at least 9 pairs of collisions. If the new chain satisfies this collision condition, it will be added to Ω'_{cn} (Steps 8 $\sim$ 11). ϕ and Ω are updated by ϕ' and Ω' after traversing all $\mu_\alpha\left(\mathcal{K}'_{j_1}\right)$ candidates in Q_{j_1} (Step 14). The fault-tolerant results of ξ_{j_2} will also be updated accordingly, η_{j_2} is updated by ϕ and Ω is saved in $\Theta_{(j_2-1)\cdot n_\alpha+1:j_2\cdot n_\alpha}$, i.e., the columns from $(j_2 - 1) \cdot n_\alpha + 1$ to $j_2 \cdot n_\alpha$ of Θ.

Based on the output Θ and η of Algorithm 1, the CV of a remaining sub-key k'_r is shown in Algorithm 2. For each sub-key to be voted, it is only necessary to traverse each row of Θ, detect the number of collisions t_c (Step 5) and η_j of that chain, and check if the total number of collisions $t_c + \eta_j^q$ exceeds the threshold τ'_σ (which is set to 14 in our experiments since a sub-key and another pair of collision is fault-tolerated (Step 6)). Due

Algorithm 1: Collision detection performed on the sub-keys selected in voting.

```
Input: $\mu_\alpha(\mathcal{K})$, $n_\alpha$ and $\xi$.
Output: $\Theta$ and $\eta$.
for $j_2$ from 1 to $n_\alpha$ do
    $Q = \xi \setminus \xi_{j_2}$; $\mathcal{K}^{'} = \mathcal{K} \setminus \mathcal{K}_{j_2}$; $\Omega = Q_1$; $\phi = \emptyset$;
    for $j_1$ from 2 to $n_\alpha - 1$ do
        $\phi^{'} = \emptyset$; $\Omega^{'} = \emptyset$; $cn = 0$; $\tau_\sigma = \binom{j_1}{2} - 1$;
        for $i_1$ from 1 to $\mu_\alpha\left(\mathcal{K}^{'}_{j_1}\right)$ do
            for $q$ from 1 to $\mathsf{Row}(\Omega)$ do
                $t_c = \sum_{s=1}^{j_1-1} \psi\left(\Omega_q^s, Q_{j_1}^{i_1}\right)$;
                if $t_c + \phi_q \geq \tau_\sigma$ then
                    $cn++$; $\phi^{'}_{cn} = t_c + \phi_q$;
                    $\Omega^{'}_{cn} = \left[\Omega_q, Q_{j_1}^{i_1}\right]$;
                end
            end
        end
        $\phi = \phi^{'}$; $\Omega \leftarrow \Omega^{'}$;
    end
    $\eta_{j_2} = \phi$; $\Theta_{(j_2-1)\cdot n_\alpha+1:j_2\cdot n_\alpha} \leftarrow \Omega$;
end
```

Algorithm 2: Collision detection on the sub-key to be voted.

```
Input: $\Theta$, $\eta$, $\tau^{'}_\sigma$, $n_\alpha$, $\xi_r$ and $\mu_\alpha(\mathcal{K}_r)$.
Output: $\xi^{'}_r$.
for $i$ from 1 to $\mu_\alpha(\mathcal{K}_r)$ do
    $cflag = 0$; $\xi^{'}_r = \emptyset$;
    for $j$ from 1 to $n_\alpha$ do
        for $q$ from 1 to $|\eta_j|$ do
            $t_c = \sum_{s=(j-1)\cdot n_\alpha+1}^{j\cdot n_\alpha} \psi\left(\Theta_q^s, \xi_r^i\right)$;
            if $t_c + \eta_j^q \geq \tau^{'}_\sigma$ then
                $\xi^{'}_r = \xi^{'}_r \cup \xi_r^i$; $cflag = 1$;
                break ;
            end
        end
        if $cflag = 1$ then
            break;
        end
    end
end
```

to the introduction of fault tolerance, the number of chains used for voting in Θ increases, while ξ_r^i can be retained if it satisfies at least one of them. Therefore, we set a collision flag $cflag$. If the number of collisions exceeds the threshold, the candidate ξ_r^i satisfies the given collision condition. We set the collision flag to 1, save ξ_r^i in $\xi_r^{'}$ and break the current loop (Steps 6 $\sim$ 9). In order to avoid unnecessary collision detection, if $cflag$ is 1, the fault tolerance on other sub-keys used for voting is not required, then Algorithm 2 continues to detect the next candidate (Steps 10 $\sim$ 11).

It is noteworthy that it is too time-consuming for Algorithms 1 and 2 when τ_d and α are very large, and there are too many sub-keys. Since too many collisions need to be detected. However, the number of candidates of the first n_α sub-keys in Fig. 6 will not change significantly in a wide range of number of measurements. Therefore, using them to vote the other $16 - n_\alpha$ sub-keys in Algorithms 1 and 2 is highly efficient. The number of possible chains in Θ is very small, which reduces the probability that very limited number of candidates are deleted in ξ_r. This is because it would be easier to satisfy the given collision conditions under a large number of chains. This also makes the algorithm complete quickly and saves collision detection time. Finally, we divide the 16 lists of remaining candidates into 8 big groups and use key rank estimation to estimate the key ranking (see Section 6.3 for details).

6 Success Probability and Complexity

6.1 Success Probability Estimation

Previous works such as FTC and GCA, set a unified threshold for all sub-keys. The optimal threshold should just be the position of the deepest sub-key, which is difficult to obtain in practice, since the full-key is unknown. The number of candidates of different sub-keys under the same *α-work-factor* defined in Section 3.3 can be very different. Therefore, each sub-key can have an independent threshold. Success probability and the candidate space are two core factors to determine the thresholds. The larger probability of key recovery, the better. However, this may also increase the difficulty of key recovery. The existing combined collision attacks and our MDCCF reflect the balance between them.

The probabilities of 16 sub-keys recovery of AES-128 in TA are independent. Therefore, the probability of 16-byte full-key falling into the threshold $\alpha = \left(\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{N}|}\right)$ is

$$\mathbf{p}_\alpha = \prod_{j=1}^{|\mathcal{N}|} \lambda_{\beta_j}\left(\mathcal{K}_j\right) \geq \prod_{j=1}^{|\mathcal{N}|} \alpha_j. \tag{23}$$

Standaert et al. defined partial success rate and global success rate in [28], the former is like the success probability of a sub-key of AES-128, the latter is like the success probability of the first-round key. However, the success probability here is different from success rate, since the former represents the theoretical expectation of success and the latter represents the success in practice.

Simply raising the threshold of each sub-key will make the candidate space grow rapidly. A good fault-tolerant mechanism can effectively reduce the candidate space, thus reducing the difficulty of collision detection. For example, if we set $\alpha = 0.95$, the probability that all 16 sub-keys fall within the threshold is only $\mathbf{p}_\alpha = 0.95^{16} \approx 0.4401$. If we improve α to 0.99, $\mathbf{p}_\alpha$ only reaches $0.99^{16} \approx 0.8515$). Obviously, this means a much larger key space, which makes it more difficult for us to recover the key (see Fig. 4). The proposed DV and CV significantly improve the success probability of key recovery without significantly enlarging the candidate space, which fully illustrates their superiority and significance.

6.2 Candidate Space Estimation

One advantage of combined distinguishers is that classification information from different distinguishers can be combined and made full use of. The CPA used in FTC and GCA in [29, 21] does not directly reflect the success probability of attackers. They use the same threshold τ_k for all sub-keys. To ensure a successful guess that all the sub-keys are within τ_k, the threshold should be set larger than the deepest sub-key. Obviously, most of the sub-keys do not reach such a depth, and thus a large number of additional undesirable candidates need to be considered. This significantly increases the cost of collision detection.

Unlike CPA, each sub-key can have an independent threshold under probabilities normalized TA. The size of candidate space under given $\alpha = \left(\alpha_1, \alpha_2, \ldots, \alpha_{|\mathcal{N}|}\right)$ of an algorithm with $|\mathcal{N}|$ sub-keys is

$$\Psi = \prod_{j=1}^{|\mathcal{N}|} \mu_{\alpha_j}\left(\mathcal{K}_j\right). \tag{24}$$

It is noteworthy that FTC, GCA and MDCCF vote and discard some candidates from each sub-key to reduce the candidate space. The probability of successful key recovery should be equal to the global success probability of the remaining candidates (i.e. satisfying Eq. 23). This is because combined collision attacks such as FTC, GCA and our MDCCF only skip guessing the discarded candidates, they do not change the original probability distribution of TA.

6.3 Key Rank Estimation

The attacker can selectively set thresholds τ_d and τ_k, and acquire the corresponding guessing space within his computing power. The difficulty of attack depends both on the candidate space and the rank of the key. An attacker has to face huge candidate space when detecting collisions if he sets very large α. We use DV to reduce the space of collision detection, and further use CV, the second level of MDCCF, to carry out collision detection. A smart attacker will not exhaust the remaining candidate space, but uses a key enumeration scheme to optimize his key recovery. GCA divides 16 sub-keys into 4 big ”blocks”. We simply compute collisions between k_1 and k_2, k_3 and k_4, $\cdots$, k_{15} and k_{16}, and launch histogram based key enumeration [24] on these 8 big ”blocks”.

7 Experiments Results

The main purpose of MDCCF is to reduce the candidates, so as to reduce the number of collisions to be detected and improve the key ranking to facilitate key recovery (e.g., key enumeration). Important parameters, such as relaxation factor π, collision threshold τ_d and the number of measurements n, are involved in our MDCCF, and we will discuss them separately in the next 3 subsections. Measurements are randomly selected to launch attacks, and each experiment is repeated 200 times. Histogram based key rank estimation [24] is directly performed on the original space and the remaining space of FTC and GCA, and the 8 big ”blocks” built after MDCCF as introduced in Section 6.3. α is set to 0.996 and the corresponding global success probability $\mathbf{p}_\alpha = 0.996^{16} = 0.9372$.

7.1 The Influence of Relaxation Factor

Firstly, we consider the influence of relaxation factor π on DV, the first level of our MDCCF. Since FTC and GCA are independent of relaxation factor, we defer the introduction of their results to Section 7.3. Specifically, we only show the impacts of π on the remaining candidate space and the rank of full-key in this sub-section. Since the distinguisher voting is actually the intersection of the candidates of 16 sub-keys between TA and CPA, the computational complexity is very small, we no longer compare its time consumption. We set τ_d to 60, and randomly select 240 from 25600 measurements for analysis. The collision makes the probability of the correct key falling into the remaining key space less than the theoretical one, since a part of candidates that do not satisfy the collision condition are discarded. The remaining full-key candidate spaces when π varies from 0.6 to 1.6 are shown in Fig. 9.

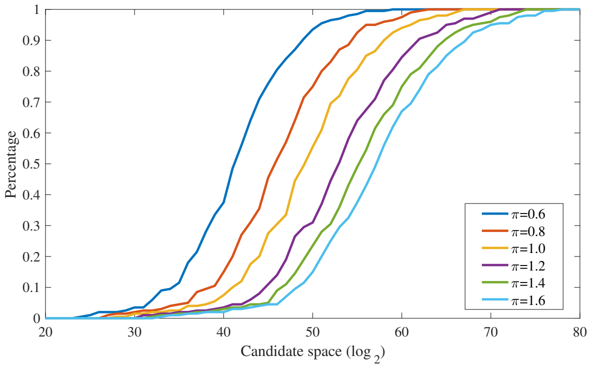

Figure 9: Candidate space (bits) under different thresholds π.

As can be seen from Fig. 9, with the decrease of relaxation factor π, more candidates are discarded by TA, which makes the remaining candidate space smaller. It is worth mentioning that although the change of candidate space on the right side is significantly smaller than that on the left in Fig. 9, this does not mean that the smaller the π is, the faster the candidate space declines. For example, 10 bits drop from 60 to 50 and from 40 to 30 looks similar, but the former is much larger than the latter. The decrease of π also makes the corresponding success probability lower (see the highest success rates under different computing power in Fig. 10). Actually, there are 189 times in 200 attacks that the correct full-key falls into the threshold $\alpha = 0.996$, close to the theoretical probability 0.9372. The success rate changes to 0.665, 0.790, 0.845, 0.880, 0.890 and 0.915 when π ranges from 0.6 to 1.6.

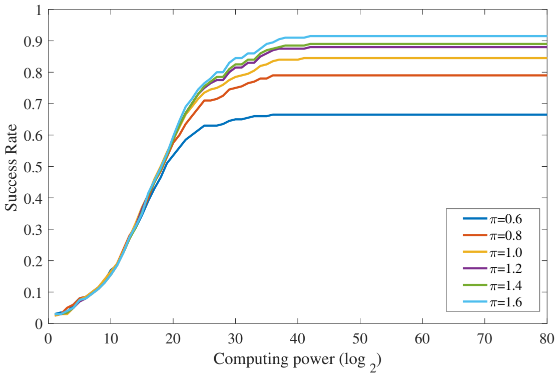

Figure 10: Key rank estimation under different thresholds π.

The key rank estimation under different thresholds π is also shown in Fig. 10. We can see that the success rates under different π are close to each other for the keys that are ranked smaller than 20 bits. This indicates that the looseness or strictness of π does not seriously affect the recovery of the key if its rank is shallow. The reason can also be found from Eq. 16 that we directly retain the candidates with probabilities greater than 0.1. However, some keys locating in deeper spaces are difficult to recover when π is small, which make the difference of their success probabilities greater. Therefore, we need to set a larger π. It can also be seen from Fig. 10 that the change of success probability is relatively large when π is smaller than 1.2, and becomes very small when $\pi > 1.2$. Therefore, it's very reasonable to set π to 1.4 in our experiments.

7.2 The Influence of τ_d

We set relaxation factor $\pi = 1.4$ and the number of power traces used in each repetition to 240, and investigate how τ_d affects the performance of our MDCCF. Actually, recovering the key from the remaining candidate space after the combined collision attack such as FTC and GCA, is a key rank estimation problem. Therefore, τ_d of CCA is independent of the original space, so we only compare the performance of FTC, GCA and our MDCCF in this section, while the introduction of the original candidate space under 240 power traces will be presented in Section 7.3 (see Fig. 12). The success rates under different computing power are shown in Fig. 11. With the increase of τ_d, more candidates satisfying the collision conditions of FTC, GCA and MDCCF are maintained, and the remaining candidate space increases gradually. Significant changes also occur in the rank of the key.

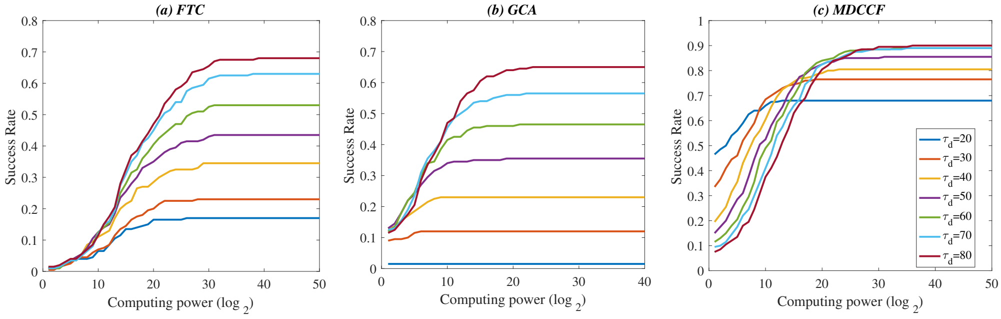

Figure 11: Success rate under different thresholds τ_d.

The smaller the τ_d, the shallower the rank of the key, and the lower the success rate (as shown in Fig. 11). This indicates that fewer candidates (including the correct and wrong ones) satisfy the collision conditions. This also tells us that smaller τ_d requires more relaxed collision conditions and better fault-tolerant mechanism. Unlike Fig. 10, the success rates of FTC, GCA are not entangled under different τ_d, since there is no corresponding fault-tolerant mechanism to preserve candidates with high probabilities. When τ_d is from 20 to 80, the success rates of FTC and GCA increase rapidly, but are still relatively low. They only reach 0.68 and 0.65 at $\tau_d = 80$. MDCCF achieves success rate of 0.68 at $\tau_d = 20$, which also illustrates its high efficiency. When τ_d reaches 60, the success rates of MDCCF are very close, so we think that $\tau_d = 60$ is a good threshold when $n = 240$. There are 190 times in 200 repetitions that the full-key falls into μ_α, so the real success rate is 0.95, close to the theoretical probability 0.9372. It becomes 0.925 after DV, the first level of MDCCF. This shows that DV does not cause a significant change in the success rate. It can also be seen from Fig. 11 that the success rate of MDCCF is about 0.90 when $\tau_d \geq 60$, which shows its superiority.

In terms of algorithm runtime, FTC and MDCCF do not change much under different

Table 1: Time consumption (seconds) under different thresholds τ_d.

τ_d	20	30	40	50	60	70	80
FTC	0.037	0.038	0.046	0.058	0.067	0.093	0.091
GCA	0.21	0.57	1.28	2.60	5.41	13.21	37.50
MDCCF	0.23	0.36	0.49	0.58	0.60	0.58	0.56

thresholds τ_d, while GCA is significantly affected by it (see TABLE 1). In fact, τ_d also has a significant impact on MDCCF. However, we can mitigate its impact by selecting 6 sub-keys with the least number of candidates for CV in the second level of MDCCF, which makes the number of chains used for voting smaller when $n = 240$ (see Fig. 6), and improves the efficiency of voting. On the other hand, Algorithm 2 exits the rotating fault tolerance on sub-keys when there is a chain making a candidate satisfy the fault-tolerant conditions, which notably reduces detection time.

7.3 The Influence of the Number of Traces

The number of power traces plays an important role in attacks, more power traces will provide more leakage information and reduce the number of guesses of sub-keys (i.e., smaller $\mu_\alpha(\mathcal{K})$), thus smaller the candidate space. We set the relaxation factor π to 1.4 and the collision threshold τ_d to 60, and the experimental results when the number of traces n ranges from 160 to 320 are shown in Fig. 12. We also show the corresponding original candidate spaces since they are dramatically affected by n. For example, the probability of candidate space larger than 70 bits is almost 1 when $n \leq 180$, and only 50% when $n = 240$, and decreases to nearly 0 when $n = 280$. The estimated ranks of keys from TA in Fig. 12 also illustrate that their locations are shallower.

The keys in FTC rank deeper than those in GCA and MDCCF. The success rates of FTC and GCA are similar and relatively low. They only reach 0.70 and 0.61 respectively when $n = 320$, and are much lower than 0.82 of MDCCF under $n = 160$. This fully demonstrates the high efficiency of DV and CV, which significantly improves the success rate without considerably increasing the candidate space and the difficulty of key recovery. More power traces will make the success rate of MDCCF closer to the theoretical probability of key falling into $\alpha = 0.996$. The difference between them is about 0.13 at $n = 160$ and decreases to 0.04 when $n = 320$. These results demonstrate the superiority of MDCCF over FTC and GCA.

Table 2: Time consumption (seconds) under different numbers of traces.

n	160	180	200	220	240
FTC	0.66	0.39	0.21	0.29	0.068
GCA	707.84	213.92	75.27	17.91	5.95
MDCCF	217.02	30.80	10.69	2.76	0.69
n	260	280	300	320	–
FTC	0.030	0.019	0.012	0.007	–
GCA	2.89	1.35	0.63	0.35	–
MDCCF	0.20	0.066	0.024	0.012	–

The runtime of GCA is most severely affected by the number of power traces (as shown in TABLE 2). We set α to 0.996 in our experiments. The number of candidates of the 6 sub-keys selected by CV are quite large when n is too small (e.g. about 25 for the second and about 60 for the 6-th re-ranked sub-keys in average as shown in Fig. 6). This also makes it time-consuming for MDCCF to tolerate sub-keys and collisions

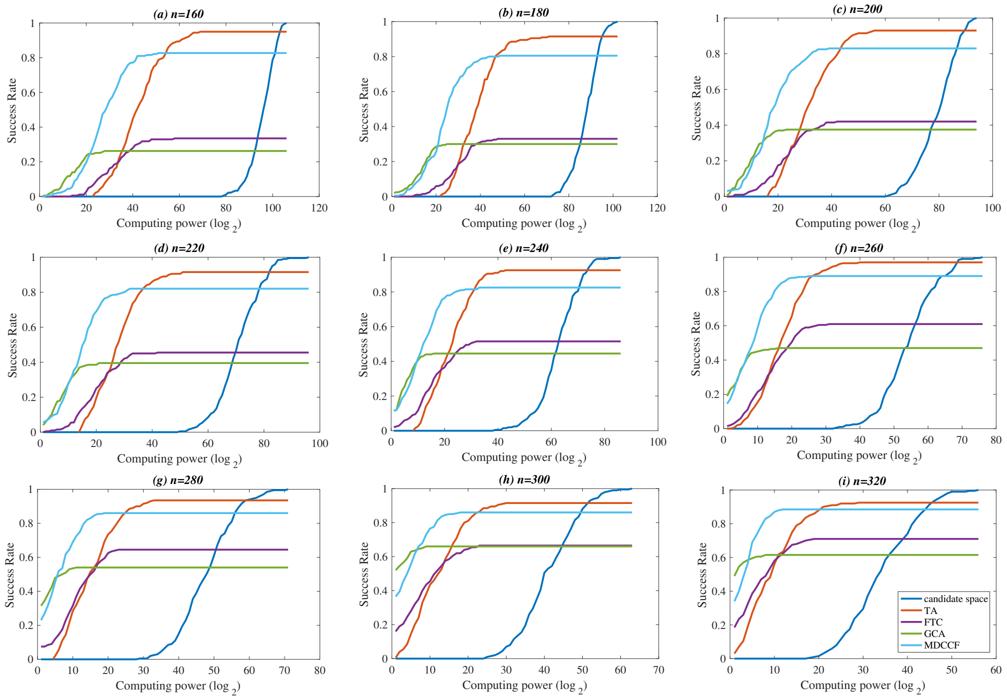

Figure 12: Success rates under different numbers of traces.

outside the thresholds. It is worth noting that the parameter α can be flexibly adjusted according to the computing power. For example, we can set $\alpha = 0.9$ when $n = 160$, and $\alpha = 0.998$ when $n = 320$. MDCCF can easily obtain the success probability of key recovery and determine the candidate space it needs to deal with. It also facilitates to choose a reasonable threshold.

8 Conclusions

This work combines TA with guessing theory to provide an attacker with estimation of the difficulty and success probability of key recovery. Moreover, in order to better filter the candidates, we extend MDCA and propose a two-level multiple-differential combined collision attack named MDCCF, which includes two parts: Distinguisher Voting and Collision Voting (i.e. DV and CV). DV can be further optimized to vote on candidates of TA to reduce the candidate space of collision detection in CV. The utilization of collision information in CCA is left behind MDCCF. CV further chooses several sub-keys with the smallest number of candidates to vote the remaining sub-keys, and retains the candidates satisfying the given collision conditions. Since the sub-keys used for voting have the fewest candidates, CV can flexibly enlarge the thresholds to make more sub-keys fall within them. If the differential condition is set reasonably, the probability of recovering the key from the remaining candidate space is similar to the one from the original space. The proposed MDCCF has made significant improvements compared to the existing combined collision attacks. However, it still has to face the challenge of huge candidate space under very limited measurements. Our future work will focus on optimizing CCA to facilitate the setting of collision threshold, and optimize the collision detection algorithms discussed in Section 5.4 to make them feasible in much deeper guessing space.

References

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

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

[3] A. Bogdanov. Improved side-channel collision attacks on AES. In *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers*, pages 84–95, 2007.

[4] A. Bogdanov. Multiple-differential side-channel collision attacks on AES. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 30–44, 2008.

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

[6] J. Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 538–552, 2012.

[7] S. Boztas. Entropies, guessing, and cryptography. *Department of Mathematics, Royal Melbourne Institute of Technology, Tech. Rep*, 6:2–3, 1999.

[8] S. Boztas. On renyi entropies and their applications to guessing attacks in cryptography. *IEICE Transactions*, 97-A(12):2542–2548, 2014.

[9] 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.

[10] 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.

[11] M. O. Choudary and P. G. Popescu. Back to massey: Impressively fast, scalable and tight security evaluation tools. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 367–386, 2017.

[12] 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.

[13] 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.

[14] C. H. Gebotys, S. Ho, and C. C. Tiu. EM analysis of rijndael and ECC on a wireless java-based PDA. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, pages 250–264, 2005.

[15] D. Genkin, L. Pachmanov, I. Pipman, and E. Tromer. Stealing keys from pcs using a radio: Cheap electromagnetic attacks on windowed exponentiation. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 207–228, 2015.

[16] N. Hanley, M. Tunstall, and W. P. Marnane. Unknown plaintext template attacks. In *Information Security Applications, 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, pages 148–162, 2009.

[17] M. S. Inci, B. Gülmezoglu, G. Irazoqui, T. Eisenbarth, and B. Sunar. Cache attacks enable bulk key recovery on the cloud. In *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, pages 368–388, 2016.

[18] 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.

[19] J. Liu, Y. Yu, F. Standaert, Z. Guo, D. Gu, W. Sun, Y. Ge, and X. Xie. Small tweaks do not help: Differential power analysis of MILENAGE implementations in 3g/4g USIM cards. In *Computer Security - ESORICS 2015 - 20th European Symposium on Research in Computer Security, Vienna, Austria, September 21-25, 2015, Proceedings, Part I*, pages 468–480, 2015.

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

[22] 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.

[23] J. O. Pliam. On the incomparability of entropy and marginal guesswork in brute-force attacks. In *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, pages 67–79, 2000.

[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] M. Renauld, F. Standaert, and N. Veyrat-Charvillon. Algebraic side-channel attacks on the AES: why time also matters in DPA. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 97–111, 2009.

[27] K. Schramm, G. Leander, P. Felke, and C. Paar. A collision-attack on AES: combining side channel- and differential-attack. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 163–175, 2004.

[28] F. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 443–461, 2009.

[29] 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.

[30] 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.

[31] 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.