Less Effort, More Success: Efficient Genetic Algorithm-Based Framework for Side-channel Collision Attacks

Jiawei Zhang[1], Jiangshan Long[1,*], Changhai Ou[1,**], Kexin Qiao[2], Fan Zhang[3], and Shi Yan[1]

[1] Wuhan University, Wuhan Hubei Province, China
[2] School of Cyberspace Science and Technology, Beijing Insitute of Technology
[3] College of Computer Science and Technology, Zhejiang University, Hangzhou, 310027

Abstract. By introducing collision information, the existing side-channel Correlation-Enhanced Collision Attacks (CECAs) performed collision-chain detection, and reduced a given candidate space to a significantly smaller collision-chain space, leading to more efficient key recovery. However, they are still limited by low collision detection speed and low success rate of key recovery. To address these issues, we first give a Collision Detection framework with Genetic Algorithm (CDGA), which exploits Genetic Algorithm to detect the collision chains and has a strong capability of global searching. Secondly, we theoretically analyze the performance of CECA, and bound the searching depth of its output candidate vectors with a confidence level using a rigorous hypothesis test, which is suitable both for Gaussian and non-Gaussian leakages. This facilitates the initialization of the population. Thirdly, we design an innovative goal-directed mutation method to randomly select new gene values for replacement, thus improving efficiency and adaptability of the CDGA. Finally, to optimize the evolutionary of CDGA, we introduce roulette selection strategy to employ a probability assignment based on individual fitness values to guarantee the preferential selection of superior genes. A single-point crossover strategy is also used to introduce novel gene segments into the chromosomes, thus enhancing the genetic diversity of the population. Experiments verify the superiority of our CDGA.

Keywords: CDGA · Genetic Algorithm · collision attack · collision chain · key recovery · side-channel analysis.

1 Introduction

Cryptographic systems, when subjected to the execution of their algorithms on physical devices, are vulnerable to inadvertent leakage of secret information

* Corresponding author:longjiangshan@whu.edu.cn
** Corresponding author: keanut@126.com

through side channels such as execution time [14], power consumption [28], electromagnetic radiation [8], and even cache patterns [18]. Side-channel attacks (SCAs) can be classified into two categories: divide-and-conquer attacks and analytical attacks [3]. The former, like Correlation Power Analysis (CPA) [4] and Template Attacks (TA) [6], compartmentalize the full key of an encryption/decryption algorithm into smaller blocks (e.g., sub-keys), and conquer them independently. The latter, like collision attacks [29], simultaneously processes multiple blocks by solving a system of equations. The analytical attacks, in particular, capitalize on more leaky information, rendering them more potent. Correlation-Enhanced Collision Attacks (CECA) is a very classic distinguisher, and it obtains the rank vector of collision candidates by calculating the correlation between the averaged power consumption of two blocks of plaintexts in block ciphers. This allows adversaries to attack the key without needing to know the specific details of cryptographic implementation, making it highly suitable for scenarios like flawed maskings, and attracting wide attention.

It is noteworthy that the sub-keys or their collisions often does not rank as the optimal ones in side-channel attacks when the sample size is insufficient. In this case, the divide-and-conquer attacks achieve key search by combining a key enumeration method [22, 25], while collision attacks perform collision detection to seek collision chains containing information of sub-keys, and exploit a strategy to recombine them to achieve key recovery. However, the existing key recovery schemes of CECA still face issues such as slow collision detection speed and low success rate in key recovery. In the next section, we will describe these key recovery schemes before going on to explain our contributions.

1.1 Related Works

The purpose of collision-chain detection is to break the independence among sub-keys and establish their connections by introducing collision information, thereby avoiding the random combinations of candidates and the huge key candidate space facing with. Collision attacks can target serially implemented cryptographic algorithms with S-box operations such as AES, DES, and PRESENT [10]. These attacks prioritize the collision values from the most probable to the least probable in collision analysis. A threshold is established for it, which means the adversary only considers a part of best candidates for each collision. The inaugural approach for collision-chain detection was given in [3]. Since this introduction, various collision-chain detection schemes have been developed, aiming to enhance collision exploitation, fault tolerance and storage optimization of collision chains.

Regarding *collision exploitation*, the pioneering collision-chain detection scheme, termed Test-Of-Chain (TOC), was given in [3]. For a block cipher with a total of N sub-keys, TOC utilizes collisions between each of two adjacent sub-keys, thereby exploiting only $N-1$ collisions out of $N(N-1)/2$ pairs while neglecting the majority. The another scheme named Fault-Tolerant Chain (FTC) [27] exploits collisions between the first sub-key and the remaining sub-keys, and incorporates *fault-tolerant strategies* to increase the probability of the correct key

meeting the given collision condition. TOC and FTC are the same if we extend them to Full-Collision Chain (FCC) [20], in which all $N(N-1)/2$ collisions are exploited during collision-chain detection. Obviously, this is very easy to achieve. Specifically, we only need to detect the collisions between the candidates of the next collision and the candidates of each collision chain, and retain new chains that meet the collision conditions [20, 21]. Unlike the above schemes considering the number of collisions satisfying the given collision conditions, some works like Wiemers et al. [30], adopts cumulative correlation coefficients to select partially optimal collision chains, and also works very well.

For *storage optimization* of collision chains, issues arise when multiple chains share the same sub-key candidates. Similar to key enumeration schemes, to alleviate the repetitive collision detection, collision chains are stored during their detection. However, this consumes large storage space, and happens in TOC [3], FTC [27], FCC [20] and all the existing schemes. To optimize this, the Lightweight Collision Detection (LCD) algorithm was given in [21]. Taking advantage of tree storage structure, LCD adopts a top-down candidates insertion mechanism to insert the collision candidates satisfying the collision conditions into the tree. It further adopts a bottom-up branch removal mechanism to delete the branches unsatisfying the collision conditions. However, this makes collision detection significantly more time-consuming compared to TOC, FTC and FCC in the scenarios where all collisions are considered.

The above strategies signify a shift towards more effective cryptographic analysis methods, which reveal key information with lower computational demand by identifying and exploiting patterns of collisions. It is noteworthy that a reasonable threshold τ_d, which means only τ_d optimal candidates for each collision is considered, plays an important role in key recovery of CECAs. If there is without such a threshold, these schemes mentioned above will return to the worst case (i.e., exhaustive case), because when considering all collision values, the key space to be faced is the same as the original key space. However, the theoretical security of CECA and the depth estimation on the ranking of collision value have never been supported by theory. All the above schemes set a unified threshold for all collision values, which significantly increases computational power.

1.2 Our Contributions

In this paper, rather than enhancing the performance of CECA like these in [2] and [7], we aim to introduce Genetic Algorithm for key recovery of CECA, establish theoretical confidence intervals for resolving population initialization issues. Our main contributions are as follows:

(1) We give a Collision Detection framework with Genetic Algorithm (CDGA), which exploits Genetic Algorithm to detect the collision chain, and has a strong capability of global searching.
(2) We theoretically analyze the performance of CECA, and bound the searching depth of the output candidate vectors with a confidence level using a rigorous hypothesis test which is suitable both for Gaussian leakages and

non-Gaussian leakages. This allows us to efficiently initialize the population of Genetic Algorithm in CDGA.

(3) We propose an innovative goal-directed mutation method for CDGA. This method enhances the efficiency of mutation and the quality of solutions by randomly selecting novel gene values from the candidates vector of a collision value for replacement. The strategy takes the advantages of random mutation and goal-directed selection, thereby enhancing the efficiency and adaptability of CDGA.

(4) We introduces refined roulette wheel selection, which employs a probability assignment process based on individual fitness values, thereby ensuring the preferential selection of superior genes. We also employ single-point crossover strategy and introduce novel gene segments into the chromosomes to enhance the genetic diversity of the population.

Experiments on the benchmark DPA *contest* $v4.1$ dataset and AT89S52 micro-controller implementing AES-256 an AES-128 algorithms verify the superiority of our CDGA.

1.3 Organization

The rest of this paper is organized as follows: Section 2 introduces side-channel attacks, CECA and its several existing key recovery schemes. Our CDGA framework is presented in detail in Section 3, and its optimizations are presented in Section 4. Section 5 presents experiments on the DPA *contest* $v4.1$ dataset and an AT89S52 micro-controller implementing AES-256 and AES-128 algorithms to illustrate the superiority of our CDGA. Finally, Section 6 concludes this paper.

2 Preliminaries

2.1 Side-channel Leakage

Let $k^* = (k_1^*, k_2^*, \ldots, k_N^*)$ denote the secret key with a total of N sub-keys (e.g., $N = 16$ for AES-128) and $k = (k_1, k_2, \ldots, k_N)$ denote a guessing candidate. Side-channel attacks usually assume a known plaintext scenario where a number of plaintexts together with the leakages unintentionally emitted during their encryptions are available to the adversary. Let $x^i = (x_1^i, x_2^i, \cdots, x_N^i)$ denote the i-th encrypted plaintext and $L^i = (L_1^i, L_2^i, \cdots, L_N^i)$ denote the corresponding power side-channel leakage measurement (i.e., power traces) related to the cryptographic computation f currently under interest (e.g., the nonlinear S-box operation of AES). Here an identical leakage model can be expressed as:

$$L_j^i = g \circ f\left(x_j^i, k_j^*\right) + \mathbb{N}_j^i = \varphi\left(x_j^i, k_j^*\right) + \mathbb{N}_j^i, \tag{1}$$

where g is the leakage function depending on the physical circuits (e.g., the Hamming weight operator), φ is the composition of g and f for simplicity, and $\mathbb{N}$ is the independent (but not necessarily Gaussian) noise. It captures both the

irrelevant leakages emitted from other computations simultaneously from the monitored device and the electronic disturbance introduced because of possibly poor measurement set-ups. Without loss of generality, we assume that the noise follows a distribution with mean $\mathbf{E}\{\mathbb{N}_j^i\} = \mu_N$ and variance $\mathbf{D}\{\mathbb{N}_j^i\} = \sigma_N^2$.

In a side-channel attack, the adversary first collects two $Q \times N$ matrices: the plaintext byte matrix $\boldsymbol{X} = (x^1, x^2, \ldots, x^Q)^\mathsf{T}$ and the side-channel measurement matrix $\boldsymbol{L} = (L^1, L^2, \ldots, L^Q)^\mathsf{T}$. The symbol "$\mathsf{T}$" here denotes matrix transpose. Then in the offline phase, he turns to some efficient side-channel analysis tool to recover k^* from the two matrices illegally.

2.2 Correlation-Enhanced Collision Attack

Correlation-Enhanced Collision Attack (abbreviated as CECA here) [29] is a potent side-channel attack to efficiently identify collisions in cryptographic computations. This attack is particularly effective against devices implementing block ciphers like AES-128. In an AES-128 algorithm, a linear collision occurs when two S-boxes within the same AES encryption or across different encryptions receive identical byte values as their inputs. Formally, a collision can be represented as:

$$\mathbf{Sbox}(x_{j_1}^{i_1} \oplus k_{j_1}^*) = \mathbf{Sbox}(x_{j_2}^{i_2} \oplus k_{j_2}^*). \tag{2}$$

Here the symbol "**Sbox**" denotes the look-up table operation. In this context, we define δ_{j_1,j_2} as the resulting XOR differential between the j_1-th and j_2-th sub-keys implicated in the collision. Consequently, we can rewrite Equ. (2) as follow:

$$\delta_{j_1,j_2} = k_{j_1}^* \oplus k_{j_2}^* = x_{j_1}^{i_1} \oplus x_{j_2}^{i_2}. \tag{3}$$

Obviously, the connection between these two sub-keys is established by δ_{j_1,j_2}. CECA categorises the leakage samples of j_1-th and j_2-th S-boxes into 256 distinct groups based on the byte values of their plaintexts, respectively. It then averages the power consumption of each group, and executes the correlation analysis using these averaged power consumption for each collision δ_{j_1,j_2}. The correlation is quantified as:

$$\rho(\overline{L}_{j_1}^{\beta \in \mathbb{F}_2^n}, \overline{L}_{j_2}^{\beta \oplus \delta_{j_1,j_2}}) \tag{4}$$

under a guessing δ_{j_1,j_2}. Here $\rho(\cdot)$ denotes the correlation coefficient computation, and $\overline{L}_j^\beta$ denotes the averaged power consumption corresponding to the j-th plaintext byte with a value of β. Obviously, if the number of power traces is enough for attack and the guessing value of δ_{j_1,j_2} is correct, ρ should be maximized. In other words, CECA ditinguisher satisfies:

$$\mathcal{D}_{\mathsf{CECA}} = \underset{\delta_{j_1,j_2} \in \mathbb{F}_2^n}{\arg\max}\, \rho(\overline{L}_{j_1}^{\beta \in \mathbb{F}_2^n}, \overline{L}_{j_2}^{\beta \oplus \delta_{j_1,j_2}}). \tag{5}$$

2.3 Collision Chain and Test-of-Chain

Let τ_d denote the threshold for CECA, i.e., only the optimal τ_d candidate of each collision δ_{j_1,j_2} are taken into consideration. When a collision occurs between two

key bytes k_{j_1} and k_{j_2}, it is captured by the Equ. (4). If a total number of n collisions are detected, the adversary obtains a system of m linear equations as follows:

$$\begin{cases} k_{j_1} \oplus k_{j_2} & = \delta_{j_1,j_2}, \\ k_{j_3} \oplus k_{j_4} & = \delta_{j_3,j_4}, \\ & \vdots \\ k_{j_{2n-1}} \oplus k_{j_{2n}} & = \delta_{j_{2n-1},j_{2n}}. \end{cases} \tag{6}$$

This collision system may only contain partial information of the full key, rather than cover its entire information. In this case, this collision system contains several independent sub-systems, with each sub-system containing several sub-keys, and each sub-key only belonging to a sub-system. To achieve key recovery, we only need to guess a sub-key for a sub-system, thus considerably constricting the candidate key space.

Cryptographic systems, such as AES-128 with its 2^{128} key candidate space, present a formidable challenge for exhaustive key search. Leveraging collision information, current key recovery schemes of CECAs transform the candidate space confined by τ_d into a reduced collision domain, thus markedly decreasing the key recovery expenses. Specifically, collision chains serve as an efficient mechanism to reduce this complexity. Collision chain comprises sub-systems, each associated with independent variables and containing collision information for a subset of sub-keys. Test-of-Chain(TOC) [3] involves examining sequential collisions between adjacent sub-keys, i.e., it detects collisions $\delta_{1,2}$, $\delta_{2,3}$, ..., $\delta_{N-2,N-1}$, $\delta_{N-1,N}$ for a cryptographic algorithm with a total of N sub-keys in the full key (e.g., $N = 16$ for AES-128).

2.4 Collision Scheme of Wiemers et al.

Rather than detecting whether a collision $\delta_{i,j}$ is within the given threshold τ_d, Wiemers et al. provided a candidate key selection method based on cumulative correlation coefficients as:

$$B = \sum_{j_1<j_2} \rho(\overline{L}_{j_1}^{\beta\in\mathbb{F}_2^n}, \overline{L}_{j_2}^{\beta\oplus\delta_{j_1,j_2}}), \tag{7}$$

in [30]. This equation is employed to refine the selection of collision candidates by maximizing observed correlations B, thereby enhancing the probability of identifying the correct AES key or significantly narrowing down the candidate space. By iteratively updating the score B, this method effectively ranks key candidates and screens out the key with a high probability.

3 Collision-chain Detection Framework with Genetic Algorithm

The Genetic Algorithm [26] is a probabilistic optimization method inspired by natural evolution. In this paper, we introduce our Collision-chain Detection

framework with Genetic Algorithm (CDGA) for CECA, and employ selection, crossover, and mutation operators to systematically search and optimize the collision chain, thus achieving more efficient key recovery. Table 1 provides an explanation of the Genetic Algorithm terms.

3.1 CDGA Framework

The corresponding framework of our CDGA is shown in Algorithm 1. We perform CECA on a set of plaintexts $\boldsymbol{X}$ and the corresponding power traces $\boldsymbol{L}$, and obtain the ranked candidate vectors Δ for each of the collisions (Step 1). In other words, Δ stores $N(N-1)/2$ collision candidate vectors for a block cipher with N subkeys, with each of them ranked according to their correlation coefficients in descending order. We further employ rigorous hypothesis testing to extract a part of optimal candidates in each collision (see Section 4). The corresponding results are output in C_{list} (Step 1), in which we mark collision values as 1 and non-collision values as 0 (see Section 3.2 for more details). The hypothesis testing to bound a valid search depth of the candidates vector for each collision will be detailed in Section 4.

Table 1: Genetic Algorithm terms.

Terms	Explanation
Individual	Collision chain
Gene	Part of Collision Chain (i.e., $\delta_{i,j}$)
Locus	Position of Gene
Alleles	Values of Gene
Phenotype	Decoded Collision Chain
Genotype	Encoded Collision Chain

First, we initialize the population using the results from CECA, output the decimal initial population POP with a size of n_{pop}. By combining with C_{list}, the candidates of each collision in Δ are reordered. Candidates that are ranked high in Δ but identified as non-collisions in C_{list} are moved back, while candidates that are ranked low in Δ but identified as collisions in C_{list} are moved forward, resulting in the reordered vectors as $\Delta^{'}$ (Step 2). Then we compute the initial population's fitness in Step 3. Before the population evolves, the current evolution generation is set to 0 and the key recovery flag is set to false (Step 4). If the correct key is not successfully recovered, or if the evolution generation has not reached the threshold τ_h, the population evolves (Step 5). The population is first encoded into a binary style $BiPOP$ (Step 6). Then, selection, crossover and mutation operations of Genetic Algorithm are performed, and the population is updated by $BiPOP^{'}$ (Steps 7 $\sim$ 9). A target-directed mutation method is designed to guide the population towards higher fitness in this work (see Section 3.5). Then new population $BiPOP^{'}$ is decoded into a decimal style (Step 10). After applying the genetic operators, the new population's fitness F_{list} is

computed (Step 11). Using the new population and its fitness, the optimal individual (collision chain) is selected and returned as k for key recovery (Step 12). Finally, we verify the guessing key k using the known plaintexts x and their corresponding ciphertexts c. If the encryption algorithm can use the guessing key k to encrypt the plaintexts x and generate the ciphertexts c, then we achieve a successful key recovery (Steps 13 $\sim$ 16).

Algorithm 1 Efficient CDGA framework for CECA.

Input: threshold of generation τ_h, size of population n_{pop}, probability of crossover p_c, probability of mutation p_m, a set of power traces $\boldsymbol{L}$, a set of plaintexts $\boldsymbol{X}$ corresponding to the power traces, mutation threshold τ_d.

Output: the key k^*.

1: $[\Delta, C_{list}]$:= **CECA**$(\boldsymbol{X}, \boldsymbol{L})$;
2: $[POP, \Delta^{'}]$:= **InitPop**$(\Delta, C_{list}, n_{pop})$;
3: F_{list} := **FITNESS**(n_{pop}, POP, C_{list});
4: $flag := false$; $generation := 0$;
5: **while** !$flag$ and $generation < \tau_h$ **do**
6: $BiPOP$:= **EncodePOP**(POP);
7: $BiPOP^{'}$:= **SELECTION**$(n_{pop}, BiPOP, F_{list})$;
8: $BiPOP^{'}$:= **CROSSOVER**$(n_{pop}, BiPOP^{'}, p_c)$;
9: $BiPOP^{'}$:= **MUTATION**$(n_{pop}, BiPOP^{'}, p_m, \tau_d.\Delta^{'})$;
10: $POP^{'}$:= **DecodePOP**$(BiPOP^{'})$;
11: F_{list} := **FITNESS**$(POP^{'})$;
12: k := **MaxFitness**$(POP^{'}, F_{list})$;
13: $flag$:= **KeyVerify**(k,x,c);
14: **if** $flag == true$ **then**
15: break;
16: **end if**
17: $POP := POP^{'}$; $generation := generation + 1$;
18: **end while**
19: **return** $k^* = k$.

3.2 Population Initialization

The ranked candidates vectors of CECA are given as a two-dimensional array Δ in Algorithm 1. In other words, we exploit Δ to sequentially store candidates of the collisions $\delta_{1,2}, \delta_{1,3}, \ldots, \delta_{1,N}, \delta_{2,3}, \delta_{2,4}, \ldots, \delta_{2,N}, \ldots, \delta_{N-1,N}$. $\Delta[i,j]$ denotes the i-th optimal candidate (with the i-th largest correlation coefficient in CECA) of the j-th collision. We can extract the optimal n_{pop} rows from array Δ to form the initial population, and an example is given in Fig. 1. We further employ rigorous hypothesis testing to extract a part of optimal candidates in each collision (see Section 4). The corresponding results are output in C_{list} (Step 1). The population initialization is further detailed in Algorithm 2. Here C_{list} is defined as a two-dimensional array, with $C_{list}[i,j]$ indicating whether the j-th collision

might take the value i ($0 \leq i \leq 255$). $C_{list}[i,j]$ is set to 1 if the j-th collision is possible, and 0 otherwise.

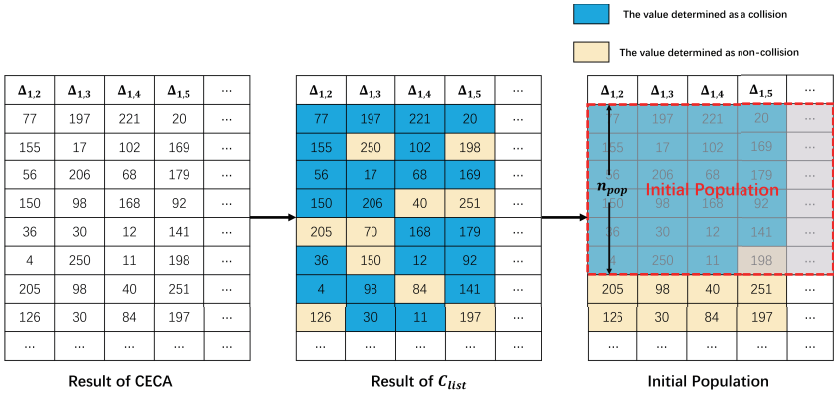


Fig. 1: An example of population initialization in our CDGA.

Algorithm 2 Population Initialization function: **InitPop**($\cdot$).

Input: the result Δ of CECA, the population POP, the collision values C_{list} for all δ-s, size of population n_{pop}, the number of key bytes N.
Output: the initial population POP, new collision rank $\Delta^{'}$.

1: $[row, col] :=$ **size**(Δ);
2: **for** $i := 1$ to row **do**
3: $\quad List_1 := []$;
4: $\quad List_0 := []$;
5: $\quad$ **for** $j := 1$ to col **do**
6: $\quad\quad$ **if** $C_{list}[\Delta[i,j],j] == 1$ **then**
7: $\quad\quad\quad List_1 :=$ **append**$(List_1, \Delta[i,j])$;
8: $\quad\quad$ **else**
9: $\quad\quad\quad List_0 :=$ **append**$(List_0, \Delta[i,j])$;
10: $\quad\quad$ **end if**
11: $\quad$ **end for**
12: $\quad \Delta^{'}[:,j] :=$ **concatenate**$(List_1, List_0)$;
13: **end for**
14: $POP := \Delta^{'}[1 : n_{pop}, 1 : N-1]$;
15: **return** POP, $\Delta^{'}$;

In the context of our CDGA framework, individuals are represented by collision chains with $N-1$ collisions $(\delta_{1,2}, \delta_{1,3}, \ldots, \delta_{1,N})$ to represent individuals in the Genetic Algorithm. The initialization process commences with an iteration through the elements in Δ. If the corresponding value in C_{list} for $\Delta[i,j]$ is 1, the related collision value is added to the $List_1$ (Step 7); otherwise, it is added

to the $List_0$ (Step 9). Subsequently, the concatenate function combines the elements of the $List_1$ and the $List_0$ in a sequential manner, thereby re-arranging the elements of the Δ list and generating a new list, designated as $\Delta^{'}$ (Step 12). Finally, the optimal n_{pop} rows of $\delta_{1,2}$, $\delta_{1,3}$, ...,$\delta_{1,N-1}$ and $\delta_{1,N}$ in array Δ are extracted in initialize the population (Step 14). Here N denotes the number of sub-keys in a block cipher as mentioned before.

3.3 Selection and Fitness Computation

Fitness function plays a pivotal role in Genetic Algorithm, serving as the primary metric for evaluating the quality of each individual. It directly influences the algorithm's performance and outcomes. In this study, the fitness function is defined in terms of the number of collisions for each individual. Algorithm 3 provides a comprehensive overview of the fitness function. Specifically, individuals are initially represented as collision chains. In order to ensure the accuracy of collision counting, it is necessary to expand each chain with partial collision information into a chain with all collisions δ-s using the **ChainExtension** in Step 4. This expansion process is based on the relationships between collision values:

$$\delta_{i,k} = \delta_{i,j} \oplus \delta_{j,k}. \tag{8}$$

Algorithm 3 Fitness computation function: **FITNESS**$(\cdot)$.

Input: size of population n_{pop}, the population POP, the collision values C_{list}.
Output: the fitness list F_{list} of POP.

```
1: [row, col] := size(POP);
2: for i := 1 to n_pop do
3:     for j := 1 to col do
4:         Chain[i,:] := ChainExtension(POP[i,:]);
5:     end for
6: end for
7: [row, col] := size(Chain);
8: for i := 1 to n_pop do
9:     F_list[i] := 0;
10:    for j := 1 to col do
11:        flag := isMember(Chain[i,j], C_list[i,:]);
12:        if flag then
13:            F_list[i] := F_list[i] + 1;
14:        end if
15:    end for
16: end for
17: return F_list;
```

For AES-128, a chain with 15 δ-s is extended into a chain with all 120 δ-s through pairwise **XOR** operations (Step 4). We iterate through these collision

chains with all collisions information, and for each collision value in the chain, we use the $flag$ marker to determine whether it belongs to the set of candidate values marked as collisions in C_{list} (Step 11). If the collision marker $flag$ is true, it is considered that a collision has occurred in that collision chain (Step 12). Subsequently, we accumulate the number of collisions that occur, which represents the fitness of the corresponding individual (Step 13).

Next, the selection operation serves as the central aspect of the Genetic Algorithm, aiming to select the better adapted individuals from the current population to generate a new generation. This process emulates the Darwinian principle of "*survival of the fittest*" from natural selection, implemented through a roulette wheel selection strategy [15] to ensure that individuals with higher fitness have a greater chance to be selected, as shown in Fig. 2.

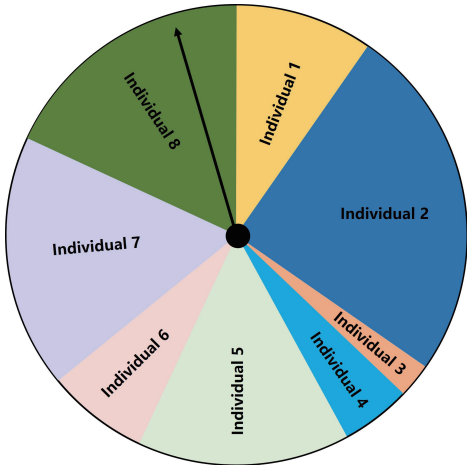


Fig. 2: Selection operation in Genetic Algorithm.

The roulette wheel selection strategy begins with calculating the sum of the fitness values $TotalFitness$ of all individuals in the population (Step 1). Then, each individual's fitness value is divided by the $TotalFitness$ to obtain the selection probability $Probabilities$ for each individual (Step 2). Next, the cumulative probability distribution for each individual is calculated (Step 3). Subsequently, a total of n_{pop} random numbers between 0 and 1 are generated and then sorted (Step 4). Individuals are selected through the random numbers (Steps 6∼7). We select the first individual whose cumulative probability value exceeds the random number to form the new population (Step 6). Algorithm 4 provides a comprehensive account of this selection process, elucidating the manner in which fitness evaluations are transformed into tangible individual selection actions. The roulette wheel selection strategy ensures that individuals with higher fitness have a greater chance of reproduction through proportional selection, while also main-

taining population diversity to some extent, thus preventing the algorithm from prematurely converging to local optima.

Algorithm 4 Selection function: **SELECTION**$(\cdot)$.

Input: size of population n_{pop}, binary population $BiPOP$, fitness values of the population F_{list}.
Output: new binary population $BiPOP^{'}$ after selection.

$TotalFitness :=$ **sum**(F_{list});
$Probabilities := F_{list}$ / $TotalFitness$;
$CumulativeProb :=$ **cumSum**$(Probabilities)$;
$RandomNum :=$ **sort**(**randomArray**(n_{pop}));
for $i := 1$ to n_{pop} **do**
    $SelectedIdx :=$ **find**$(CumulativeProb \geq RandomNum[i])$;
    $BiPOP^{'}[i,:] := BiPOP[SelectedIdx,:]$;
end for
return $BiPOP^{'}$;

3.4 Crossover

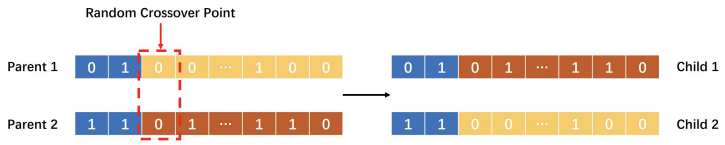


Fig. 3: Crossover operation in CDGA.

The crossover operation represents a fundamental aspect of the genetic search process in Genetic Algorithm. It simulates the chromosomal crossover observed in biological genetics, introducing new genetic material into the offspring of a population. The primary target of crossover is to enhance genetic diversity, thereby expanding the search space and increasing the probability of identifying the global optimum. We employ a single-point crossover strategy [12], as illustrated in Fig. 3. In single-point crossover, a randomly selected position on the chromosomes of two parental individuals serves as the crossover point, thereby dividing the chromosome into two parts. At this crossover point, the chromosome segments of the parents are exchanged, thereby creating two new offspring individuals. In particular, the first offspring receives the front half of the chromosome from the first parent and the rear half from the second parent, while the second offspring receives the reverse inheritance order.

Algorithm 5 provides a comprehensive procedure for implementing single-point crossover, where the input parameter p_c represents the crossover probability. The algorithm initially evaluates the population size in Step 1, then iterates

through each pair of individuals in the population to perform crossover operations. The algorithm compares a randomly generated number with the crossover probability, p_c, to determine whether to proceed with crossover (Steps 3∼4). If a decision is made to execute the crossover, the crossover point *cpoint* is determined by multiplying a random number by the length of the individual and then rounds the result to the nearest integer (Step 5). The individuals exchange gene segments at the crossover point, thereby creating new individuals (Steps 6∼9). This process helps to preserve beneficial genetic trait combinations and to avoid disrupting them.

Algorithm 5 Crossover function: **CROSSOVER**$(\cdot)$.

Input: size of population n_{pop}, binary population $BiPOP$, probability of crossover p_c.
Output: new binary population after crossover $BiPOP^{'}$.

1: $[row, col] :=$ **size**$(BiPOP)$;
2: **for** $i := 1$ to $n_{pop} - 1$ step by 2 **do**
3: $r :=$ **Random**(0,1);
4: **if** $r < p_c$ **then**
5: $cpoint :=$ **round**$(r \times col)$;
6: $BiPOP^{'}[i, 1 : cpoint] := BiPOP[i, 1 : cpoint]$;
7: $BiPOP^{'}[i, cpoint + 1 : col] := BiPOP[i + 1, cpoint + 1 : col]$;
8: $BiPOP^{'}[i + 1, 1 : cpoint] := BiPOP[i + 1, 1 : cpoint]$;
9: $BiPOP^{'}[i + 1, cpoint + 1 : col] := BiPOP[i, cpoint + 1 : col]$;
10: **else**
11: $BiPOP^{'}[i, :] := BiPOP[i, :]$;
12: $BiPOP^{'}[i + 1, :] := BiPOP[i + 1, :]$;
13: **end if**
14: **end for**
15: **return** $BiPOP^{'}$.

3.5 A Goal-directed Mutation Strategy

A well-designed mutation strategy in Genetic Algorithm can significantly enhance its efficiency and the quality of solutions. We introduce a goal-directed mutation method in this section. The method is characterized by the specification of a threshold value, τ_d, for variation. Only new values from the optimal τ_d collision candidates of a collision in $\Delta^{'}$ are randomly selected for replacement. The outline of our mutation function is presented by Algorithm 6. First, we decode the current binary population into a decimal population (Step 1). Then, a gene locus (*mpoint*) is first randomly determined by choosing a random integer between 1 and 15 (Step 5). Then, we select the collision candidate values corresponding to the gene locus from $\Delta^{'}$ (Step 6), and randomly choose one item from the optimal τ_d entries of these candidate values for replacement (Steps 8∼9). After traversing all individuals in the population, we encode the new decimal population back into a binary population (Step 14).

A significant advantage of our mutation strategy is its ability to optimize the search for collision chains by directing the population to explore collision chains with a higher number of collisions while maintaining genetic diversity. By purposefully selecting the optimal τ_d candidates of each collision in $\Delta^{'}$ for random replacement, it helps to increase the probability of finding a better collision chain. Concurrently, the high flexibility and directed nature of this strategy facilitate the reduction of the vast collision space, thereby reducing the searching complexity on collision chains. Furthermore, by enabling the algorithm to explore novel solution spaces not encompassed by the current population, basic bit mutation facilitates the algorithm's escape from local optima traps, thereby enhancing the probability of identifying global optima.

Algorithm 6 Mutation function: **MUTATION**$(\cdot)$.

Input: size of population n_{pop},the binary population $BiPOP$, the probability of mutation p_m, mutation threshold τ_d, new rank of collisions $\Delta^{'}$.
Output: new binary population after mutation $BiPOP^{'}$.

$POP :=$ **Decode**$(BiPOP)$;
for $i = 1$ to n_{pop} **do**
  $r :=$ **random**(0,1);
  if $r < p_m$ **then**
    $mpoint :=$ **randomInt**$([1, 15])$;
    $choose := \Delta^{'}[1 : \tau_d, mpoint]$;
    $POP^{'}[i, :] := POP[i, :]$;
    $rankIndex :=$ **randomInt**$([1, \tau_d])$;
    $POP^{'}[i, mpoint] := choose[rankIndex]$;
  else
    $POP^{'}[i, :] := POP[i, :]$;
  end if
end for
$BiPOP^{'} :=$ **Encode**$(POP^{'})$;
return $BiPOP^{'}$;

4 Parameters Selection and Optimization

4.1 A Bottleneck

Taking advantages of the characteristics of Genetic Algorithm, our CDGA scheme achieves a strong capability of global searching. However, its efficiency is largely subject to performance of the side-channel collision attack exploited, especially when the Signal-to-Noise Ratio (SNR) is low and the collision values in overall are sunk to much deeper positions in the output candidate vectors. Therefore, the proper choice of threshold τ_d for collision values becomes a serious bottleneck for this kind of attacks, further optimizations for this issue is still expected and motivating.

Table 2: Experimental ranks of collisions among sub-keys of AES-256 on DPA *contest* $v4.1$ dataset.

$\hat{\rho}(\delta_{1,2})$	$\hat{\rho}(\delta_{1,3})$	$\hat{\rho}(\delta_{1,4})$	$\hat{\rho}(\delta_{1,5})$	$\hat{\rho}(\delta_{1,6})$	$\hat{\rho}(\delta_{1,7})$	...
0.747 (77)	0.646 (197)	0.731 (221)	0.747 (20)	0.707 (204)	0.753 (49)	...
0.186 (155)	0.224 (250)	0.207 (102)	0.186 (198)	0.212 (133)	0.198 (88)	...
0.177 (56)	0.198 (17)	0.189 (68)	0.164 (169)	0.212 (185)	0.173 (14)	...
0.166 (150)	0.164 (206)	0.167 (40)	0.163 (251)	0.207 (240)	0.163 (224)	...
0.164 (205)	0.162 (70)	0.158 (168)	0.159 (179)	0.191 (165)	0.162 (138)	...
0.157 (36)	0.153 (150)	0.155 (12)	0.158 (92)	0.185 (132)	0.158 (39)	...
0.156 (4)	0.151 (98)	0.154 (84)	0.158 (141)	0.181 (107)	0.156 (15)	...
0.151 (126)	0.151 (30)	0.149 (11)	0.157 (197)	0.169 (78)	0.154 (68)	...
0.150 (15)	0.150 (19)	0.147 (231)	0.151 (186)	0.156 (243)	0.148 (179)	...
0.146 (132)	0.148 (3)	0.147 (159)	0.147 (207)	0.155 (134)	0.139 (161)	...
0.145 (207)	0.146 (56)	0.141 (149)	0.143 (150)	0.153 (76)	0.136 (133)	...
0.144 (113)	0.142 (144)	0.140 (105)	0.143 (175)	0.152 (80)	0.136 (176)	...
0.138 (184)	0.141 (216)	0.139 (6)	0.140 (52)	0.151 (242)	0.136 (124)	...
0.138 (240)	0.140 (72)	0.137 (73)	0.137 (225)	0.150 (73)	0.132 (222)	...
0.137 (159)	0.139 (249)	0.136 (226)	0.136 (70)	0.141 (129)	0.131 (95)	...
...	...	...	...	...	...	...

To overcome the huge potential searching space left to our CDGA and achieve a better initialization of population, a natural and direct way comes to bound the searching depth of the output candidate vectors with a confidence level. That is, for each output vector of the side-channel collision distinguisher, we wish to identify a boundary (or criteria) so that we can discard those candidates which in theory seem less worthy to consider. An experiment example of collisions among 16 sub-keys in the first round of AES-128 returned by CECA on the open dataset DPA *contest* $v4.1$ [1] is displayed in Table. 2. As shown, CECA sorts the collision candidates according to the descending values of correlation coefficient, indicating that those placed in front are most likely to be the correct ones. Yet, the exact possibilities remain unclear so it raises a quantification question that from which collision candidate (to the last one in the output vector) their small possibilities are allowed to be omitted by without considering them in the following CDGA. In essence, this is a hypothesis test problem where the candidate space can be squeezed according to a settled confidence level. Once successfully determining a valid searching depth, it will significantly reduce the burdens of our CDGA (corresponding to the discarded candidates covered by shadow in the Table 2) and improve its efficiency.

To bound a valid searching depth, we have to deal with the distribution of correlation coefficients in CECA, which is a hard problem in statistic. Some mathematical tools have been invented to approximate the distribution of correlation coefficients asymptotically (e.g., the Fisher's z-transformation [17]). However, they usually come with the cost of low accuracy and (or) requirement of large samples which may be far from satisfactory. Therefore, in this paper, we take an in-depth look on the behaviours of correlation coefficients in collision

scenario and introduce an equivalent transformation which results in a much easier derivation of the confidence interval.

4.2 Correlation Coefficient in Collision Scenario

In this subsection, we simplify the expression of correlation coefficient taking the case of collision between the j_1-th and j_2-th sub-keys. Specifically, we strip off those terms which are independent of the guessing collision δ_{j_1,j_2} in the concerned scenario of this paper. Expanding the expression of CECA, we have:

$$\begin{aligned}\mathcal{D}_{\mathsf{CECA}} &= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\max}\ \hat{\rho}(\overline{L}_{j_1}^{m\in\mathbb{F}_2^n}, \overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}) \\ &= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\max}\ \frac{\hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_1}^{m}\times\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\} - \hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_1}^{m}\}\times\hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\}}{\sqrt{\hat{\mathbf{D}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_1}^{m}\}}\times\sqrt{\hat{\mathbf{D}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\}}}.\end{aligned} \tag{9}$$

CECA classifies leakages according to the plaintext byte value and calculates sample means $\overline{L}_{j_1}^{m} = \hat{\mathbf{E}}_{i\in[1,Q]}\{L_{j_1}^{i}|x_{j_1}^{i}=m\}$ and $\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}} = \hat{\mathbf{E}}_{i\in[1,Q]}\{L_{j_2}^{i}|x_{j_2}^{i}=m\oplus\delta_{j_1,j_2}\}$ before estimating the correlation coefficient, in order to overcome some effects of noise and random masking. As we have stated in Section 4.1, we pursue the main goal of addressing the full secret key k^* especially for the cases in which correct candidates are buried deep in the output vector so the time requirement for key exhaustion turns to be unaffordable. Typically, SNR of this scenario is low and the corresponding leakage measurements should be sufficient. As a result, there will appear all possible values of the randomly encrypted plaintext bytes x_{j_1} and x_{j_2}. This ensures that $\forall m, \delta_{j_1,j_2}\in\mathbb{F}_2^n$, we can always find leakages belonging to plaintext byte pair $(x_{j_1}^{i_1}=m, x_{j_2}^{i_2}=m\oplus\delta_{j_1,j_2})$ (the exact number of measurements needed to fulfill this condition can be found in [16]). Guessing different candidates of the collision δ_{j_1,j_2}^{*} boils down to holding the order of sample means from the j_1-th sub-key while reordering those from the j_2-th. Based on this observation, the expression of CECA may be simplified as:

$$\begin{aligned}\mathcal{D}_{\mathsf{CECA}} &= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\max}\ \hat{\rho}(\overline{L}_{j_1}^{m\in\mathbb{F}_2^n}, \overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}) \\ &= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\max}\ \hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_1}^{m}\times\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\},\end{aligned} \tag{10}$$

due to the closure property of "$\oplus$" operation in cryptographic algorithm, which is the same as that of balanced measurement set (see [5]).

4.3 Gaussian Leakages

In this subsection, we first focus on the most common side-channel collision case where leakages follow Gaussian distribution so strict confidence interval is available. Contrary to what we did in the simplification of correlation coefficient

(i.e., stripping off collision independent terms), here we add additional terms to construct a statistic that is easy to tackle. In this case, Equ. (10) for CECA can be rewritten as:

$$\begin{aligned}
\mathcal{D}_{\mathsf{CECA}} &= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\max}\ \hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{\overline{L}_{j_1}^{m}\times\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\}\\
&= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\min}\ \hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{-2\times\overline{L}_{j_1}^{m}\times\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\}\\
&= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\min}\ \hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{-2\times\overline{L}_{j_1}^{m}\times\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}\}+\hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{(\overline{L}_{j_1}^{m})^2\}\\
&+\hat{\mathbf{E}}_{m\in\mathbb{F}_2^n}\{(\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}})^2\}\\
&= \underset{\delta_{j_1,j_2}\in\mathbb{F}_2^n}{\arg\min}\sum_{m\in\mathbb{F}_2^n}(\frac{\overline{L}_{j_1}^{m}-\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}}-0}{\sigma_N/\sqrt{Q/2^{n+1}}})^2.
\end{aligned}\tag{11}$$

Traversing the output candidate vector, once we encounter the correct collision value, then we have a series of standard normally distributed variables $Z_{\mathsf{ceca}}^{(m,\delta^*_{j_1,j_2})}=\sqrt{\frac{Q}{2^{n+1}\sigma_N^2}}(\overline{L}_{j_1}^{m}-\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}})\sim\mathcal{N}(0,1)$, $m=0,1,...,2^n-1$. They are independent and identically distributed (i.e., i.i.d). Otherwise for any incorrect collision candidate, the variables become $Z_{\mathsf{ceca}}^{(m,\delta_{j_1,j_2})}=\sqrt{\frac{Q}{2^{n+1}\sigma_N^2}}(\overline{L}_{j_1}^{m}-\overline{L}_{j_2}^{m\oplus\delta_{j_1,j_2}})\sim\mathcal{N}(\mu_m,1)$, $m=0,1,...,2^n-1$. Though independent, they may not be identically distributed: $\exists\, m_1\neq m_2$, it has $\mu_{m_1}\neq\mu_{m_2}\neq 0$. As a consequence, with such intuition, we are able to conduct a hypothesis testing checking the zero population mean, based on the null hypothesis that the current candidate is the correct one.

In order to act directly on the output candidate vector and to check all the variables at once, let $\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}=\sum_{m\in\mathbb{F}_2^n}(Z_{\mathsf{ceca}}^{(m,\delta_{j_1,j_2})})^2$. This statistic should follow central χ^2 distribution (i.e., $\mathbf{E}\{\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}\}=2^n$, $\mathbf{D}\{\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}\}=2^{n+1}$) if and only if $\delta_{j_1,j_2}=\delta^*_{j_1,j_2}$, or else non-central χ^2 distribution with a positive non-centrality parameter $\lambda=\sum_{m\in\mathbb{F}_2^n}\mu_m^2$ and $\mathbf{E}\{\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}\}=2^n+\lambda$, $\mathbf{D}\{\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}\}=2^{n+1}+4\lambda$. It is clear that both right-tailed tests on mean and variance are feasible. In this paper, we discuss the formal case due to its popularity. Fig. 4 exhibits probability density functions (i.e., pdf) of χ^2 distribution with different λ-s. The only central χ^2 distribution corresponding to $\delta_{j_1,j_2}=\delta^*_{j_1,j_2}$ is plotted in blue (i.e., $\lambda=0$). The positive value of λ lies on collision candidate, algebraic property of cryptographic algorithm and physical characteristic of the underlying circuits. We choose $\lambda=10,20,30$ just for illustration which are plotted in red, yellow and purple respectively. They can belong to different incorrect candidates on a certain target device. In the figure, we also set $n=4$ (i.e., $m\in\mathbb{F}_2^4$, degree of freedom $df=16$) for simplicity which well represents some lightweight cryptographic algorithms (e.g., PRESENT) in practice.

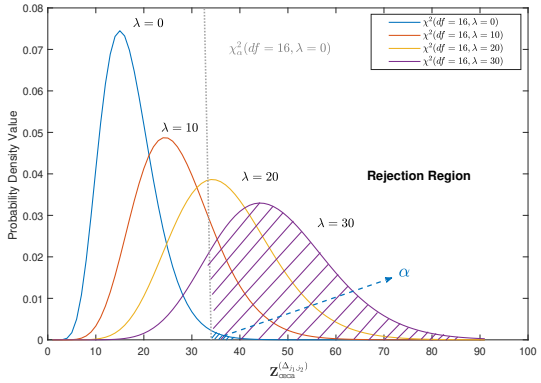


Fig. 4: χ^2 distributions with different non-centrality parameters λ.

In the last step of the hypothesis testing, a significance level will be chosen, says α, to help determining whether we have observed an occurrence of rare event, on the basis of the null hypothesis that the current candidate under checking is correct. The significance level α $(0 < \alpha < 1)$ satisfies:

$$P\{\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})} > \chi^2_\alpha(df = 2^n, \lambda = 0)\} = \alpha, \tag{12}$$

where $\chi^2_\alpha(df = 2^n, \lambda = 0)$ represents the quantile of α on the pdf and it is plotted in grey dotted line. The rejection region is the right part of this line. Taking $\lambda = 30$ as an example, the large purple shadow is the true possibility that $\mathrm{Z}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}$ can fall in the rejection region. However, admitting the null hypothesis, there are hardly any possibilities (i.e., the tiny blue shadow α). As a result, once we have observed an instance $\hat{\mathrm{Z}}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}$ that fell into the rejection region, we can directly reject the null hypothesis with very small probability of making a mistake. Note that a smaller λ indicates a smaller chance to distinguish it from the correct collision value (i.e., a close candidate from the perspective of side-channel collision attack).

To verify the feasibility and effectiveness of above method in practical application, we conduct a simulated experiment in which we control the Gaussian noise. We wish to see whether and to which extent it can bound a reasonable searching depth of the output candidate vector. Experimental results on collision $\delta^*_{1,5}$ of AES-128 are displayed in Fig. 5. Values of instances $\hat{\mathrm{Z}}_{\mathsf{ceca}}^{(\delta_{1,5})}$ in this attack are represented by 256 vertical solid lines. The left border of the unilateral rejection region under $\alpha = 0.01$ is denoted by a grey dotted line. As shown, the only pink line (corresponding to $\hat{\mathrm{Z}}_{\mathsf{ceca}}^{(\delta^*_{1,5})}$) is not placed at the first of the output vector (i.e., the ninth place) and therefore further searching is necessary. In the original scheme, all of the 256 candidates needed to be considered, but only 21 are now worthy of investigation thanks to the hypothesis testing. It reduces the entropy from 256 to 22 by dividing all candidates into two parts and discards those that fell into the rejection region. Additionally, for further validation, we sampled the

statistic $\hat{\mathsf{Z}}_{\mathsf{ceca}}^{(\delta_{1,5}^*)}$ for 10000 times and plot the results in blue histogram whose bin width is chosen according to the Scott's rule [23]. The red dash-dotted line is the theoretical pdf of distribution $\chi^2(df = 256, \lambda = 0)$. Its pretty fitting rate with the histogram implies the soundness of our theory.

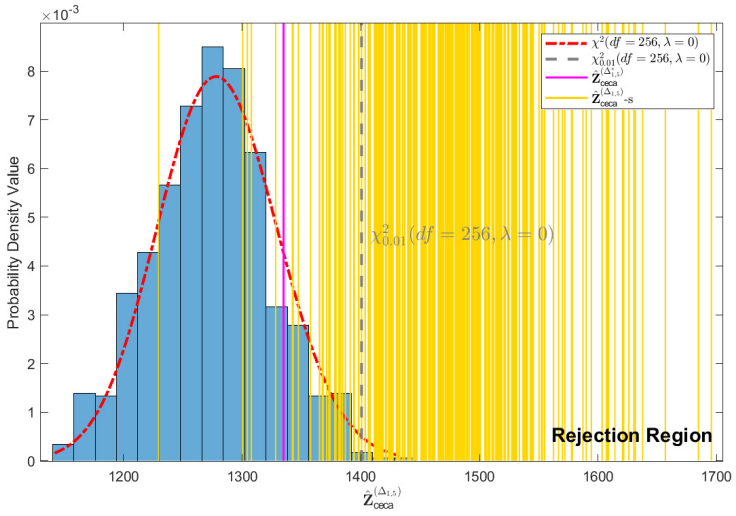


Fig. 5: A simulation experiment on collision $\delta_{1,5}^*$ of AES-128.

4.4 Non-Gaussian Leakages

In practice, side-channel leakages may not strictly follow a Gaussian distribution which restricts the usage of χ^2-based hypothesis testing and the bottleneck problem of attack efficiency can again appear in this case. Hence, in this subsection, we improve the method and put forward a common form of statistic that is suitable for any leakages. It is achieved by introducing a constant fraction:

$$\mathbf{D}_{\mathsf{CECA}} = \underset{\delta_{j_1,j_2} \in \mathbb{F}_2^n}{\arg\min} \frac{1}{2^n} \sum_{m \in \mathbb{F}_2^n} (\overline{L}_{j_1}^m - \overline{L}_{j_2}^{m \oplus \delta_{j_1,j_2}})^2. \tag{13}$$

In accordance with the central limit theory, for a number of i.i.d random variables $X_1, X_2, ..., X_n$, the sample mean variable $\frac{1}{n}\sum_{i=1}^n X_i$ is approximately Gaussian (i.e., $\frac{1}{n}\sum_{i=1}^n X_i \sim \mathcal{N}(\mathbf{E}(X), \mathbf{D}(X)/n)$) regardless of the actual distribution of X. This allows getting rid of the Gaussian leakage assumption. Again taking CECA and S-box for illustration, let $Y_{\mathsf{ceca}}^{(m,\delta_{j_1,j_2})} = (\overline{L}_{j_1}^m - \overline{L}_{j_2}^{m \oplus \delta_{j_1,j_2}})^2$ and $\mathbf{Y}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})} = \frac{1}{2^n}\sum_{m \in \mathbb{F}_2^n} Y_{\mathsf{ceca}}^{(m,\delta_{j_1,j_2})}$. The hypothesis testing on expectation focuses on:

$$\begin{aligned}
\mathbf{E}\{\mathbf{Y}_{\mathsf{ceca}}^{(\delta_{j_1,j_2})}\} &= \frac{1}{2^n}\sum_{m\in\mathbb{F}_2^n}\mathbf{E}\{(\varphi\left(m\oplus k_{j_1}^*\right)+\overline{\mathbb{N}}_{j_1}-\varphi\left(m\oplus\delta_{j_1,j_2}\oplus k_{j_2}^*\right)-\overline{\mathbb{N}}_{j_2})^2\} \\
&= \frac{1}{2^n}\sum_{m\in\mathbb{F}_2^n}(\varphi\left(m\oplus k_{j_1}^*\right)-\varphi\left(m\oplus\delta_{j_1,j_2}\oplus k_{j_2}^*\right))^2+\mathbf{E}\{(\overline{\mathbb{N}}_{j_1}-\overline{\mathbb{N}}_{j_2})^2\} \\
&+ \frac{1}{2^n}\sum_{m\in\mathbb{F}_2^n}\mathbf{E}\{\left(\varphi\left(m\oplus k_{j_1}^*\right)-\varphi\left(m\oplus\delta_{j_1,j_2}\oplus k_{j_2}^*\right)\right)(\overline{\mathbb{N}}_{j_1}-\overline{\mathbb{N}}_{j_2})\} \\
&= \kappa(k_{j_1}^*, k_{j_2}^*\oplus\delta_{j_1,j_2})+\frac{2^{n+1}\sigma_N^2}{Q},
\end{aligned} \tag{14}$$

where $\overline{\mathbb{N}}_{j_1}$ and $\overline{\mathbb{N}}_{j_2}$ are the averaged independent noise components in $\overline{L}_{j_1}$ and $\overline{L}_{j_2}$, and $\kappa(k_{j_1}^*, k_{j_2}^*\oplus\delta_{j_1,j_2}) = \frac{1}{2^n}\sum_{m\in\mathbb{F}_2^n}(\varphi\left(m\oplus k_{j_1}^*\right)-\varphi\left(m\oplus\delta_{j_1,j_2}\oplus k_{j_2}^*\right))^2$ is the so-called confusion coefficient describing the confusion property of the target device from the perspective of side-channel attacks [9].

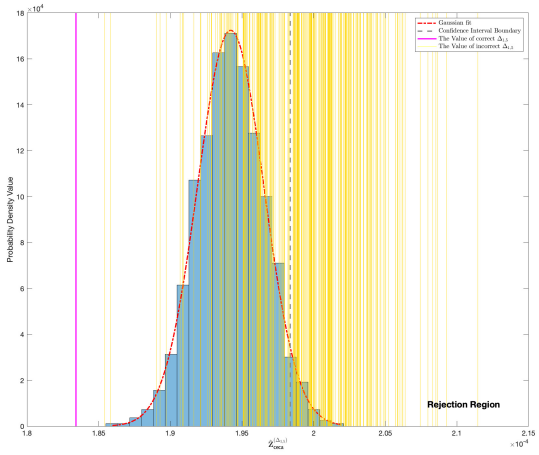


Fig. 6: A real experiment on collision $\delta_{1,5}^*$ of AES-128.

In light of the square operation in confusion coefficient, it is easy to find that if and only if $\delta_{j_1,j_2} = \delta_{j_1,j_2}^*$, then $\kappa(k_{j_1}^*, k_{j_2}^*\oplus\delta_{j_1,j_2}) = \kappa(k_{j_1}^*, k_{j_1}^*\oplus\delta_{j_1,j_2}^*\oplus\delta_{j_1,j_2}) = \kappa(k_{j_1}^*, k_{j_1}^*) = 0$. Otherwise, $\kappa(k_{j_1}^*, k_{j_2}^*\oplus\delta_{j_1,j_2}) > 0$. Right side unilateral hypothesis testing should be viable to detect outliers based on the null hypothesis that the current candidate is the correct one. Remaining steps are the same as the χ^2-based hypothesis testing so we do not detail them here. To validate the feasibility and effectiveness of above method in practical application, we conduct

a real experiment on DPA *contest* $v4.1$ dataset and the results are shown in Fig. 6. Similar to the χ^2-based hypothesis testing, a desirable result has been achieved by suggesting us give up a total of 180 candidates. With it, we can foresee a large improvement of the following CDGA.

4.5 Parameters Used in Genetic Operators

The crossover probability p_c and the mutation probability p_m collectively influence the efficiency of the Genetic Algorithm. Therefore, we examine both of these two parameters concurrently. To evaluate the impact of each value of (p_c, p_m), we performed our CDGA on power traces from DPA *contest* $v4.1$ dataset. Specifically, we set the population size to 20, p_m was between 0 and 0.01 in increments of 0.001, and p_c was between 0.6 and 1 in increments of 0.1. Once the genetic algorithm had converged, the number of collisions recovered under each (p_c, p_m) was counted. Figs. 7 and 8 illustrated the impact of (p_c, p_m) on the number of collisions recovered. The red lines indicated the highest average number of collisions recovered for the corresponding p_c/p_m values. Our CDGA demonstrated optimal performance when $(p_c, p_m) = (0.7, 0.001)$.

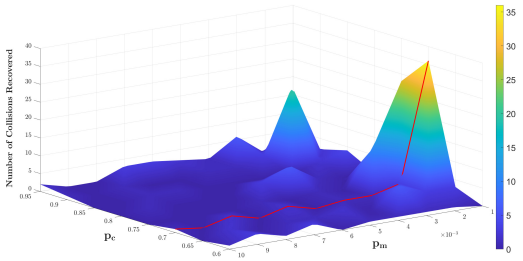


Fig. 7: The relation between the number of collisions recovered and (p_c, p_m).

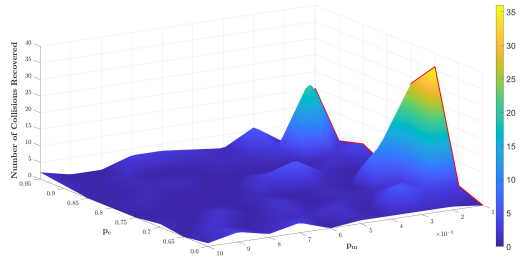
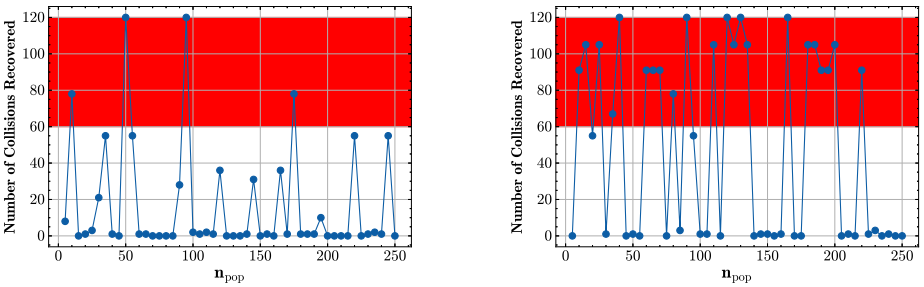


Fig. 8: The relation between the number of collisions recovered and (p_c, p_m).

Based on the (p_c, p_m) optimization, we further selected the population size. In the case of fewer power traces (e.g., 1500), we chose the population size from 2 to 250, with a step size of 5. In the case of more power traces (e.g., 2500), we chose the population size from 2 to 250, with a step size of 10. Once the Genetic Algorithm had converged, the number of recovered collisions was counted and the results were shown in Fig. 9(a) and 9(b). The region where collisions were recovered between 60 and 120 was labelled in the figure. It can be seen that the population size $n_{pop} = 50$ was reached earlier to recover all collisions under both 1500 and 2500 power traces. Therefore, in subsequent experiments, we set $n_{pop} = 50$.



(a) The relation between the number of collisions recovered and n_{pop} under 1500 power traces.

(b) The relation between the number of collisions recovered and n_{pop} under 2500 power traces.

Fig. 9: The relation between the number of collisions recovered and n_{pop}.

5 Experimental Results

To evaluate the efficiency of our CDGA, experiments were conducted on DPA *contest v*4.1 dataset and the dataset captured from the ATS8952 micro-controller, where the AES-256 and AES-128 algorithms were serially implemented. This facilitates linear collision attacks[4]. First, CECA was conducted and its output vectors were optimized and analyzed using the our CDGA, Collision scheme of Wiemers et al., respectively. As mentioned in Section 2.3, TOC only exploits $N-1$ out of $N(N-1)/2$ collisions. Taking AES-128 for an example, the original TOC exploits only 15 collisions, with the remaining 105 collisions being discarded, which is very wasteful. Additionally, it can be observed that TOC and FTC are less effective when the collision space is relatively huge. This is due to the fact that they utilize significantly fewer collisions and are time-consuming when considering the same candidate space. To fully utilize the collision information, we extended them to FCC, thus making the comparison fair.

[4] Linear collision attack is infeasible in parallel implementation as well explained in [11].

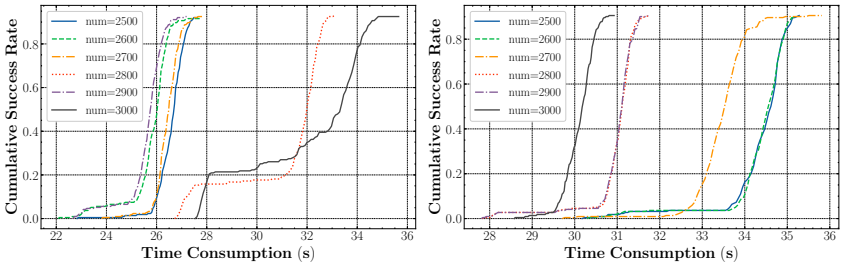
All experiments were conducted on a Lenovo desktop computer equipped with a 10-core Intel Core i5-13400F CPU, 16GB RAM, and the Windows 10 operating system running MATLAB R2023a. Based on the methods outlined in Sections 4.1∼4.4, we utilized a 95% confidence interval and incorporated the results of CECA for population initialization. In the design of the genetic algorithm, informed by the findings in Section 4.5, we set the population size to 50, with a crossover probability p_c of 0.7 and a mutation probability p_m of 0.01. These parameters worked very well in our experiments. To reduce the search space and enhance search efficiency, we conducted experiments with the collision value random replacement range τ_d of the target-oriented mutation strategy, as described in Section 3.5, set to 25, 30, 35 and 50. Due to the fact that these two schemes do not filter collision chains like CDGA, to prevent explosive growth on number of chains and for simplicity, we only consider their 300 optimal collision chain candidates in each iteration in the following experiments.

5.1 Experiments on DPA *Contest v*4.1 Dataset

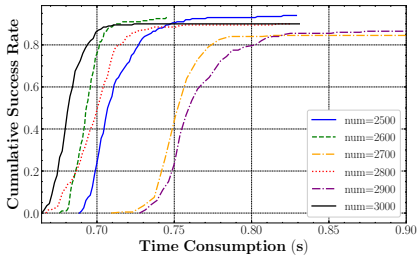
DPA *contest* v4.1 provides a benchmark dataset with an AES-256 algorithm protected by Rotated S-boxes Masking (RSM) [19] and implemented on an Atmel ATMega-163 smart card. The time samples with the highest correlation coefficients in CPA for each S-box were identified, and power traces were randomly extracted for the experiments. A time-cumulative analysis was conducted to evaluate the common evaluation indicator, success rate [24] of key recovery, for three methods: CDGA, collision scheme of Wiemers et al., and FCC (also the extended TOC and FTC). The analysis was performed with varying sample sizes and τ_d parameters. The results in Fig .10(c) demonstrated that the success rate of our CDGA with different sample sizes rapidly increased to nearly 1 after a time consumption of approximately 0.7 seconds, well indicating its efficiency and stability. In contrast, the time consumption of the collision scheme of Wiemers et al. increased significantly with the number of power traces according to Table 3, indicating that it is less efficient than our CDGA. FCC (the extended TOC and FTC) demonstrated a lower success rate compared to the other two schemes when consuming the same running time.

Table 3: Success rate and time consumption under different number of traces on DPA *contest* v4.1 dataset

Number of traces	2500	2600	2700	2800	2900	3000
CDGA	0.8289s 94%	0.7463s 93%	1.4070s 88.5%	0.8256s 90%	1.1027s 90.5%	0.8308s 90%
FCC (the extended TOC and FTC)	35.2778s 90.7%	35.2508s 91.1%	35.8127s 90.8%	31.7694s 90%	31.7315s 90.2%	30.9265s 89.9%
Wiemers et al.	27.6279s 92.07%	27.7746s 91.6%	27.8549s 92.5%	33.0877s 89%	27.1979s 91.14%	35.6522s 89%



(a) Wiemers et al. (b) FCC (the extended TOC and FTC)



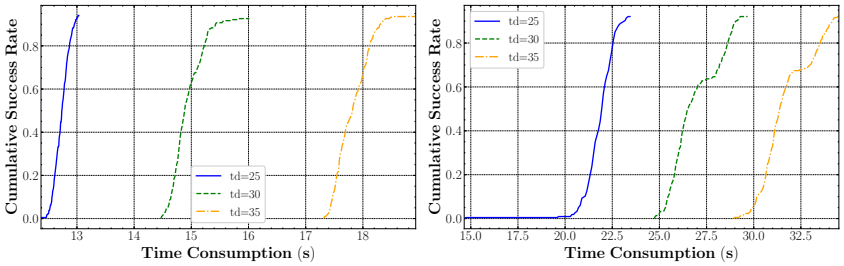
(c) CDGA

Fig. 10: Time consumption of three schemes under different number of traces on DPA *contest* *v*4.1 dataset.

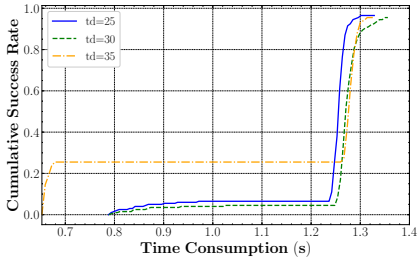
Table 4: Success rate and time consumption under different thresholds τ_d on DPA *contest* *v*4.1 dataset.

Number of traces	25	30	35
CDGA	1.3286s 96.7%	1.3542s 95.4%	1.3228s 95.4%
FCC (the extended TOC and FTC)	23.46s 92%	29.64s 91.8%	34.44s 93.1%
Wiemers et al.	13.019s 94.1%	16.016s 92.6%	18.915s 93.6%

For different τ_d-s, we ran the three methods 200 times under 2500 power traces, and τ_d is set to 25, 30 and 35, respectively. Our CDGA exhibited a consistent running time of less than 1.4 seconds across all τ_d values according to Table 4, with a success rate exceeding 95%. To achieve a success rate slightly lower than that of CDGA, FCC (the extended TOC and FTC) requires approximately 20 times of time consumption according to Fig. 11. Moreover, τ_d exerts a considerable influence on the time consumption of collision scheme of Wiemers et al. and FCC (the extended TOC and FTC) in Fig. 11(a) and Fig. 11(b), whereas it has a relatively minor impact on CDGA.



(a) Wiemers et al. (b) FCC (the extended TOC and FTC)



(c) CDGA

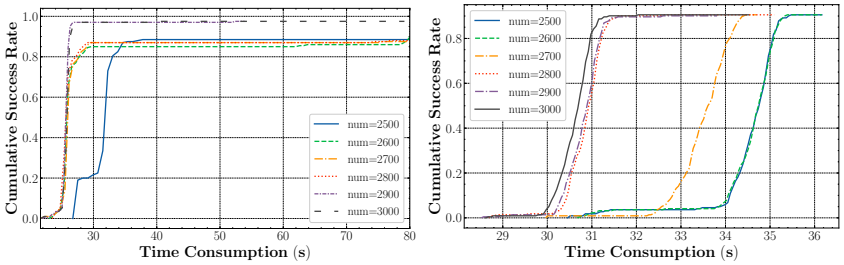
Fig. 11: Time consumption of three schemes under different thresholds τ_d on DPA *Contest* $v4.1$ Dataset.

In conclusion, our CDGA exhibits remarkable computational efficiency across a wide range of sample sizes, with a stable running time of less than 1.4 seconds. This is considerably lower than that of the FCC (also the extended TOC and FTC) and the collision scheme of Wiemers et al. Moreover, the success rate of our CDGA consistently exceeds 95%, indicating excellent stability and reliability. CDGA achieves the optimal performance in both computational speed and success rate compared to FCC (the extended TOC and FTC) and collision scheme of Wiemers et al.

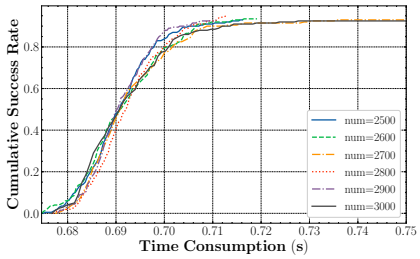
5.2 Experiments on an AT89S52 Micro-controller

Our second experiment was performed on an AES-128 algorithm implemented on an AT89S52 micro-controller designed for side-channel attacks. It ran at 12 MHz and we exploited a Picoscope 3000 to sample power traces under a sampling rate of 125 MS/s.Fig. 12 depicts the time consumption and success rate of CDGA, collision scheme of Wiemers et al., and FCC (the extended TOC and FTC) under different number of traces, respectively. CDGA achieved a success rate of over 90% in less than 0.75 seconds for all sample sizes according to Table 5. FCC (the extended TOC and FTC) demonstrated a slightly lower success rate than CDGA, with a significantly longer run time of 34∼36 seconds for processing 2,500 to 3,000 traces, as illustrated in Table 5. In comparison, the collision scheme of

Wiemers et al. exhibited a success rate comparable to our CDGA, but took tens of times as long as CDGA.



(a) Wiemers et al. (b) FCC (the extended TOC and FTC)



(c) CDGA

Fig. 12: Time consumption of three schemes under different number of traces on ATS8952 Dataset.

Table 5: Success rate and time consumption under different number of traces on ATS8952 Dataset.

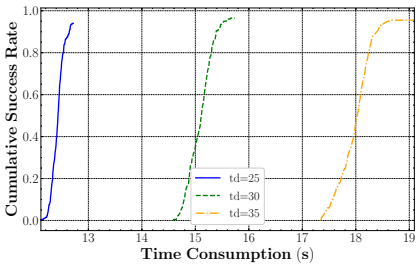
Number of traces	2500	2600	2700	2800	2900	3000
CDGA	0.7162s 93%	0.7191s 93.5%	0.7127s 95%	0.7107s 93%	0.7551s 92.5%	0.7494s 93%
FCC (the extended TOC and FTC)	36.1628s 90.5%	36.16s 90.7%	34.69s 90.7%	35.091s 91%	33.8120s 91.2%	34.5208s 91.5%
Wiemers et al.	97.21s 88.6%	79.85s 89.5%	79.7s 87.6%	79.59s 87.7%	78.91s 97.1%	53.88s 97.5%

Our framework was capable of achieving a success rate of nearly 100% under different τ_d-s within one second according to Table 6 and Fig. 13(c). Our CDGA maintained a stable and efficient performance with success rates ranging from 94.8% to 97.1%. In contrast, the FCC (the extended TOC and FTC) method had a slightly lower success rate and a significantly higher time consumption,

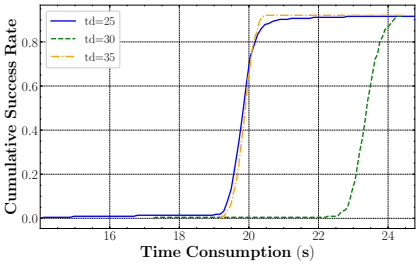
requiring approximately 23 to 34 seconds to process a sample size of 2,500 to 3,000 traces according to Table 6. Moreover, collision scheme of Wiemers et al., while achieving a comparable success rate, required a significantly longer time, with time consumption ranging from approximately 12 to 19 seconds, as illustrated in Fig. 13(a).

Table 6: Success rate and time consumption under different thresholds τ_d on ATS8952 Dataset.

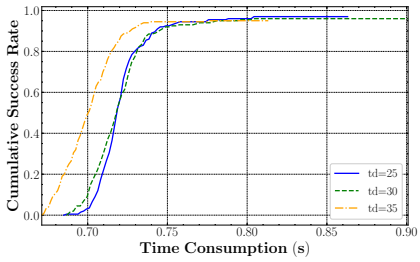
Number of traces	25	30	35
CDGA	0.863s 97.1%	0.899s 95.8%	0.813s 94.8%
FCC (the extended TOC and FTC)	23.46s 92%	29.64s 91.8%	34.44s 93.1%
Wiemers et al.	12.721s 93.6%	15.743s 96.5%	19.094s 95.5%



(a) Wiemers et al.



(b) FCC (the extended TOC and FTC)



(c) CDGA

Fig. 13: Time consumption of three schemes under different thresholds τ_d on ATS8952 Dataset.

The high efficiency of our CDGA is primarily attributable to its distinctive optimization mechanism. CDGA rapidly explores the solution space by simulating the natural evolutionary process through the use of operations such as

selection, crossover and mutation, thereby identifying a near-optimal solution in a shorter period of time. Furthermore, CDGA is capable of effectively parallelizing the process and utilizing multiple individuals in the population to search simultaneously, thereby enhancing computational efficiency. In contrast, FCC (the extended TOC and FTC) and Wiemers et al. methods often necessitate traversing a substantial number of potential combinations during processing, resulting in significant time consumption.

6 Conclusion

The existing collision-chain detection algorithms still face difficulties in threshold setting and low efficiency in collision detection. To address these issues, we proposed a highly efficient framework named CDGA employing a Genetic Algorithm for collision-chain detection. In comparison to the existing methodologies, our CDGA facilitates a higher success rate in a shorter time. The experimental results demonstrated its superiority compared with the existing works.

Key recovery from very huge candidate spaces in side-channel collision attacks remains a highly challenging task. In the future, we will concentrate on further optimizing our CDGA framework with the objective of enhancing the speed of collision-chain detection. Moreover, more advanced Genetic Algorithm strategies, such as multi-objective optimization algorithms [13] and adaptive genetic algorithms [31], will be explored to improve the efficiency of collision-chain detection and optimization. Finally, we will seek to enhance the selection, mutation, and crossover strategies for our CDGA, thereby facilitating more efficient computation and more accurate outcomes.

References

1. DPA *contest v*4.1. http://www.dpacontest.org/home/.
2. A. Bogdanov. Multiple-differential side-channel collision attacks on AES. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 30–44. Springer, 2008.
3. A. Bogdanov and I. Kizhvatov. Beyond the limits of DPA: combined side-channel collision attacks. *IEEE Trans. Computers*, 61(8):1153–1164, 2012.
4. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.
5. N. Bruneau, C. Carlet, S. Guilley, A. Heuser, E. Prouff, and O. Rioul. Stochastic collision attack. *IEEE Trans. Inf. Forensics Secur.*, 12(9):2090–2104, 2017.
6. S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. K. Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002,*

Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

7. Y. Ding, L. Zhu, A. Wang, Y. Li, Y. Wang, S. M. Yiu, and K. Gai. A multiple sieve approach based on artificial intelligent techniques and correlation power analysis. *ACM Trans. Multim. Comput. Commun. Appl.*, 17(2s):71:1–71:21, 2021.
8. 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 B. Thuraisingham, D. Evans, T. Malkin, and D. Xu, editors, *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. ACM, 2017.
9. Y. Fei, A. A. Ding, J. Lao, and L. Zhang. A statistics-based success rate model for DPA and CPA. *J. Cryptogr. Eng.*, 5(4):227–243, 2015.
10. B. Gérard and F. Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version. *J. Cryptogr. Eng.*, 3(1):45–58, 2013.
11. B. Gérard and F. Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting: extended version. *J. Cryptogr. Eng.*, 3(1):45–58, 2013.
12. A. B. A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. M. Hammouri, and V. B. S. Prasath. Choosing mutation and crossover ratios for genetic algorithms - A review with a new dynamic approach. *Inf.*, 10(12):390, 2019.
13. A. M. Hernández, I. V. Nieuwenhuyse, and S. Rojas-Gonzalez. A survey on multi-objective hyperparameter optimization algorithms for machine learning. *Artif. Intell. Rev.*, 56(8):8043–8093, 2023.
14. P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.
15. A. Lipowski and D. Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, 2012.
16. J. Long, C. Ou, Y. Ma, Y. Fan, H. Chen, and S. Zheng. How to launch a powerful side-channel collision attack? *IEEE Transactions on Computers*, 2023.
17. S. Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In T. Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.
18. A. Moghimi, G. Irazoqui, and T. Eisenbarth. Cachezoom: How SGX amplifies the power of cache attacks. In W. Fischer and N. Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 69–90. Springer, 2017.
19. 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 W. Rosenstiel and L. Thiele, editors, *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1173–1178. IEEE, 2012.

20. C. Ou, S. Lam, and G. Jiang. The science of guessing in collision-optimized divide-and-conquer attacks. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(6):1039–1051, 2021.
21. C. Ou, S. Lam, C. Zhou, G. Jiang, and F. Zhang. A lightweight detection algorithm for collision-optimized divide-and-conquer attacks. *IEEE Trans. Computers*, 69(11):1694–1706, 2020.
22. R. Poussier, F. Standaert, and V. Grosso. Simple key enumeration (and rank estimation) using histograms: An integrated approach. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016.
23. D. W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.
24. F. Standaert, T. Malkin, and M. Yung. A unified framework for the analysis of side-channel key recovery attacks. In A. Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 443–461. Springer, 2009.
25. N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F. Standaert. An optimal key enumeration algorithm and its application to side-channel attacks. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2012.
26. A. Vié. Qualities, challenges and future of genetic algorithms: a literature review. *CoRR*, abs/2011.05277, 2020.
27. D. Wang, A. Wang, and X. Zheng. Fault-tolerant linear collision attack: A combination with correlation power analysis. In X. Huang and J. Zhou, editors, *Information Security Practice and Experience - 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings*, volume 8434 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 2014.
28. 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. Inf. Forensics Secur.*, 13(5):1301–1316, 2018.
29. A. Wiemers and D. Klein. Entropy reduction for the correlation-enhanced power analysis collision attack. In A. Inomata and K. Yasuda, editors, *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, volume 11049 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2018.
30. A. Wiemers and D. Klein. Entropy reduction for the correlation-enhanced power analysis collision attack. In A. Inomata and K. Yasuda, editors, *Advances in Information and Computer Security - 13th International Workshop on Security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, Proceedings*, volume 11049 of *Lecture Notes in Computer Science*, pages 51–67. Springer, 2018.
31. J. Zhao, J. Zhang, Y. Shi, and L. Shi. Based on adaptive improved genetic algorithm of optimal path planning. In *ICMIP 2022: 7th International Conference on Multimedia and Image Processing, Tianjin, China, January 14 - 16, 2022*, pages 225–230. ACM, 2022.