Efficient Framework for Genetic-Algorithm-Based Correlation Power Analysis

An Wang, Yuan Li, Yaoling Ding*, Liehuang Zhu, and Yongjuan Wang

***Abstract*—Various Artificial Intelligence (AI) techniques are combined with classic side-channel methods to improve the efficiency of attacks. Among them, Genetic Algorithms based Correlation Power Analysis (GA-CPA) is proposed to launch attacks on hardware cryptosystems to extract the secret key efficiently. However, the convergence rate is unsatisfactory due to two problems: individuals of the initial population generally have low fitnesses, and the mutation operation is hard to generate high-quality components. In this paper, we give an analysis framework to solve them. Firstly, we employ lists of sorted candidate key bytes obtained with CPA to initialize the population with high quality candidates. Secondly, we guide the mutation operation with lists of candidate keys sorted according to fitnesses, which are obtained by exhausting the values of a certain key byte and calculating the corresponding correlation coefficients with the whole key. Thirdly, key enumeration algorithms are utilized to deal with ranked candidates obtained by the last generation of GA-CPA to improve the success rate further. Simulation experimental results show that our method reduces the number of traces by 33.3% and 43.9% compared to CPA with key enumeration and GA-CPA respectively when the success rate is fixed to 90%. Real experiments performed on SAKURA-G confirm that the number of traces required in our method is much less than the numbers of traces required in CPA and GA-CPA. Besides, we adjust our method to deal with DPA contest v1 dataset, and achieve a better result of 40.76 traces than the winning proposal of 42.42 traces. The computation cost of our proposal is nearly 16.7% of the winner.**

***Index Terms*—Side-channel analysis, Correlation power analysis, Genetic algorithm, Key enumeration, Mutation.**

I. Introduction

In 1996, Kocher proposed side-channel attacks [1] which showed great threats on cryptosystems. From 1999 to 2004, several side-channel attack models were put forward, such as template attacks [2], collision attacks [3], mutual information analysis [4], and so on. In the past ten years, Correlation Power Analysis (CPA) [5] was widely used in side-channel attacks and physical security evaluations on cryptographic devices. With the development of hardware technology, the parallel implementation of S-box is more used as a way to improve the performance of cryptographic devices. Besides, high frequency clock will also cause power consumption traces of the S-box to be superimposed. However, when CPA is applied in the analysis of these devices, the attack efficiency will be greatly reduced. This is because identical modules of a cryptographic primitive, such as S-box operation on each bytes of intermediate states, are executed simultaneously. When we focus on one module, the power consumption produced by the other modules is treated as noise and consequently improve calculation complexity. Key enumeration algorithms [6], [7], [8], [9] and rank evaluation methods [10], [11], [12], [13] were proposed to enumerate candidate keys in decreasing order of likelihood or estimate the rank of the correct key based on partial information obtained by classic CPA or template attacks. These methods reduced required number of power traces in attacks, while improve memory and/or calculation complexity a lot.

In recent years, Artificial Intelligence (AI) techniques are increasingly combined with side-channel attacks to improve the efficiency of key recovery. They are roughly divided into two categories. One is classification using machine learning methods, and generally requires profiling, such as support vector machine [14], [15], [16], neural network [17], [18], [19], decision tree [20], [21], rotation forest [20], [21], [22], self-organizing maps [16], [23], naive Bayes [16], [20], [24], and so on. The other category is key recovery using heuristic evolutionary algorithms, and generally doesn't require profiling, such as genetic algorithms [25], [26] and hill climbing [27].

Genetic algorithms [28] are a powerful category of AI techniques in solving optimization problems. In 2015, Zhang et al. presented a Genetic-Algorithm-based CPA (GA-CPA) [25], which took full use of information in the power consumption traces generated by multiple S-boxes. Ding et al. also proposed a method using genetic algorithms to conduct attacks on bitwise linear leakages in 2020 [26]. In these approaches, candidate keys were regarded as individuals, and a group of individuals constructed a population. The correlation coefficients are defined as fitnesses in order to assess the qualities of individuals. With the evolution of a population, the average fitness of individuals becomes higher and higher, and accordingly candidate keys approximate to the target one gradually. Consequently, the target key is recovered. The authors claimed that GA-CPA improved the success rate significantly. However, its convergence rate and result are unsatisfactory due to following problems:

- Genes in the initial population are not so desirable because their fitnesses are generally very low.
- Since the mutation operation usually generates good individuals randomly, the evolution procedure always encounters local optimal problems, and new high-quality individuals appear coincidentally with low probability

An Wang, Yuan Li, Yaoling Ding and Liehuang Zhu are with School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (email: wanganl, ly18, dyl19, liehuangz@bit.edu.cn).

Yongjuan Wang is with Institute of Cyberspace Security, Information Engineering University, Zhengzhou 450001, China (email: pinkywyj@163.com).

* Corresponding author.

Manuscript received April 19, 2005; revised August 26, 2015.

during evolution.
- When the number of power traces is insufficient, the key with the highest fitness is usually not the correct one, and genetic algorithms are prone to fall into local optimum, which makes it more difficult to recover the target key.

A. Our contributions

In this paper, we propose a framework for GA-CPA, which consists of three stages. It solves the three problems mentioned above respectively, and improves the efficiency. AES-128 is taken for example to illustrate the new framework. we list our contributions as follows.

- Classic CPA is employed to rank candidates of each key byte according to their correlation coefficients. Individuals (candidate keys) in the initial population of GA-CPA are generated by combining 16 key bytes which are selected randomly from a set of candidates, whose correlation coefficients is larger than the others. Therefore, average fitness of individuals in the initial population is inherent high.
- The correlation coefficient optimization problem discussed in this paper is affected by S-box (high-order Boolean function). The variation of the key byte has a complex and discontinuous effect on the correlation coefficient. Therefore, we propose a mutation scheme, which is to sort the candidate values of key bytes according to their fitness, and then select the byte with the largest fitness as the mutation direction. Our scheme aims to improve the quality of the mutation results, which improves the mutation efficiency and speeds up the algorithm convergence.
- Due to the ranking procedure of each key byte, key enumeration algorithms can be combined easily. If the best individual of the last generation is not the target key, candidates of each key bytes will be ranked according to fitnesses, which are calculated with other key bytes fixed to the best individual. Key enumeration algorithms are employed to search for the correct key with these ranking lists.

Simulation and real experiments are conducted to compare the efficiency among our method, GA-CPA and CPA with key enumeration. Experimental results show that convergence rate of genetic algorithms is improved by the new framework. Besides, our method requires only 66.70% and 56.14% traces of CPA with key enumeration and GA-CPA respectively to achieve 90% success rate. In addition, we launch attacks on DPA contest v1 dataset, and compare required numbers of traces and computation costs between our method and the winning proposal. The results show that our method performs better than the winning one both in the required number of traces (1.66 less) and the computation cost (83.30% less).

B. Organization

The rest parts of this paper are organized as follows. Section II gives a brief description of classic CPA and GA-CPA. Some imperfections of GA-CPA are exposed in this section. In Section III, our new framework and is proposed, and advantages of the key enumeration scheme used in it are analyzed. Section IV addresses the process of determining parameters and operators employed in our method by experiments. Comparisons of CPA (with/without key enumeration), GA-CPA and our proposal based on simulation and real experiments are shown in Section V. The detail of adjustments that make our method applicable to DES are introduced in the same section. After that, our proposal is compared with hill-climbing-based method by launching attacks on DPA contest v1 dataset. We conclude this paper in Section VI.

II. Preliminaries

A. Correlation Power Analysis

Correlation power analysis proposed by Brier et al. [5] is based on the dependence between power consumption and Hamming distances of handled data of an cryptographic devices. A linear model $W = a \times \mathrm{HW}(D \oplus R) + b$ is presented to define the relationship between the power consumption W and the Hamming distance from current state D to last state R, denoted as H. In the formula, a stands for a scalar gain depending on the circuit, and b encloses offsets, time dependent components and noise of the cryptographic device. The correlation coefficient between W and H,

$$\rho_{W,H} = \frac{\mathrm{cov}(W, H)}{\sigma_W \sigma_H},$$

is used to achieve CPA attacks on cryptographic devices, in which candidate values of a key byte is scanned exhaustively and ranked according to correlation coefficients that they produced with W. As a result, the target key is recognized as the one that corresponds to the maximized $|\rho_{W,H}|$.

B. Genetic Algorithm

Genetic algorithm [28] is a probabilistic optimization method, which is inspired by the model of natural evolution. Potential solutions are defined as individuals, and the fitness of a certain individual is evaluated by an objective function. For each individual, the higher its fitness is, the more likely it is the optimal solution.

The basic procedure of genetic algorithm is that a population constructed by n_{pop} individuals and initialized randomly evolves by three operations, namely selection, crossover and mutation. These operations are executed in a loop, called a generation, and terminated when there is little changes between two generations. At the end of genetic algorithms, the individual that has the highest fitness is outputted as the optimal solution. The major functions of the three operations are as follows:

- **Selection** is an operation that improves the average fitness of a population by selecting individuals with higher fitnesses. There are various schemes designed to handle different problems, such as tournament selection [29], roulette wheel selection [30], truncation selection [31], and so on.
- **Crossover** is an operation that intends to assemble high-quality components in one individual by exchange bit

strings between two selected individuals (called parents). This operation is generally executed by a certain probability p_c.

- **Mutation** is an operation that generates new bit strings by altering one or a few bits in an individual. This operation provides local searching for most problems. However, it is executed by a very low probability p_m, in order not to influence the convergence.

C. Genetic-Algorithm-based Correlation Power Analysis

Zhang et al. [25] presented a GA-CPA method which combined CPA with genetic algorithms to improve the efficiency of power analysis in 2015. Genetic algorithm was employed to obtain the optimal correlation coefficient produced by a whole key but avoid exhausting all candidates.

In GA-CPA, candidate keys are defined as individuals, and the correlation coefficient produced by the intermediate states (encrypted by the candidate key) is the fitness, formally defined as $Fitness := Corr(Trace, Intermediate(Plaintext, Candidate\_Key))$. At the beginning of the algorithm, A group of individuals is generated randomly, called a population. The fitnesses of individuals are calculated and sorted. Then, GA-CPA performs selection, crossover and mutation operations in sequence on this population to obtain a population composed of new individuals. In this process, high-quality individuals are retained and the population fitness improves. Finally, calculate fitnesses of the new individuals. If the optimal solution (correct key) is found, the algorithm ends. Otherwise, the above three basic operations are repeated until the upper limit of iteration is reached. Algorithm 1 describes GA-CPA with pseudo code.

InitPopulation(n_{pop}) initializes population $\mathcal{P}$ with n_{pop} individuals. **ComputeFitness**($\mathcal{P}$, T) computes fitnesses of individuals in $\mathcal{P}$ with a set of power traces generated by target cryptographic device. **Selection**($\mathcal{P}$) selects high-quality individuals $child_1$ and $child_2$ from $\mathcal{P}$ randomly. **Crossover**($child_1$, $child_2$, p_c) recombines $child_1$ and $child_2$ with probability p_c, and **Mutation**($child_{1/2}, p_m$) mutates $child_{1/2}$ with p_m. $\mathcal{P}_{child}$ acts as an intermediate population that holds individuals among iterations. **MaxFitness**($\mathcal{P}$) outputs an individual K_{best} which produces the largest fitness in $\mathcal{P}$. **Verify**(K_{best}) tests whether K_{best} is the correct key.

Algorithm 1 Genetic-Algorithm-based Correlation Coefficient Analysis.

Input: threshold of generation th, size of population n_{pop}, probability of crossover p_c, probability of mutation p_m, a set of power traces T.
Output: the optimal key K_{best}.

```
P :=InitPopulation(n_pop);
ComputeFitness(P, T);
found :=false;
generation := 0;
while !found and generation < th do
    P_child := Φ;
    for j := 1 to n_pop/2 do
        child_1 :=Selection(P);
        child_2 :=Selection(P);
        Crossover(child_1, child_2, p_c);
        Mutation(child_1, p_m);
        Mutation(child_2, p_m);
        P_child:=P_child ∪ {child_1, child_2};
    end for
    P := P_child;
    ComputeFitness(P, T);
    K_best :=MaxFitness(P);
    if Verify(K_best) = true then
        found :=true;
    end if
    generation := generation + 1;
end while
return K_best;
```

D. The Problem of Genetic-Algorithm-based CPA

Firstly, GA-CPA is supposed to converge to the correct key gradually. However, it is infeasible for the mutation operation to generate a correct key byte gradually by altering one (even a few) bit randomly, because the outputs of S-box operations are changed largely and irregularly along with a little altering in the input. Therefore, in a certain attack, GA-CPA is almost reduced to a random search for some key bytes. That is to say the evolution procedure always encounters local optimal problems. Consequently, the convergence rate is unsatisfactory.

Secondly, since the population is initialized randomly, average fitness of individuals are usually very low. It requires quite a lot generations to obtain individuals with high-quality components (i.e. correct key values). What's more, there might be even no correct value for some key bytes in the initial population, which will increase the requirements for the ability of the mutation operation to generate them.

Thirdly, in many practical attacks, it is hard to acquire sufficient power traces, which leads to a high signal to noise ratio. As a consequent, the fitness produced by the target key may be very close to wrong guesses, may even not be the highest. At this condition, GA-CPA fall into local optimum easily, i.e. converge to a wrong or an immature solution.

In our approach, we aim at solving these three problems.

III. A Framework for Genetic-Algorithm-based Correlation Power Analysis

A. Description of New Framework

Similar to GA-CPA, we regard candidate keys as individuals, and initial a population with a group of randomly generated candidates. On the whole, our new framework of GA-CPA consists of three stages, which is illustrated in Figure 1.

At the first stage, random plaintexts are encrypted by AES-128 and the corresponding power traces are acquired. A classical CPA is conducted on these power traces, in order to obtain a preliminary ranking of the candidate values for each key bytes. These rankings are based on the correlation coefficients produced by the power traces and intermediate

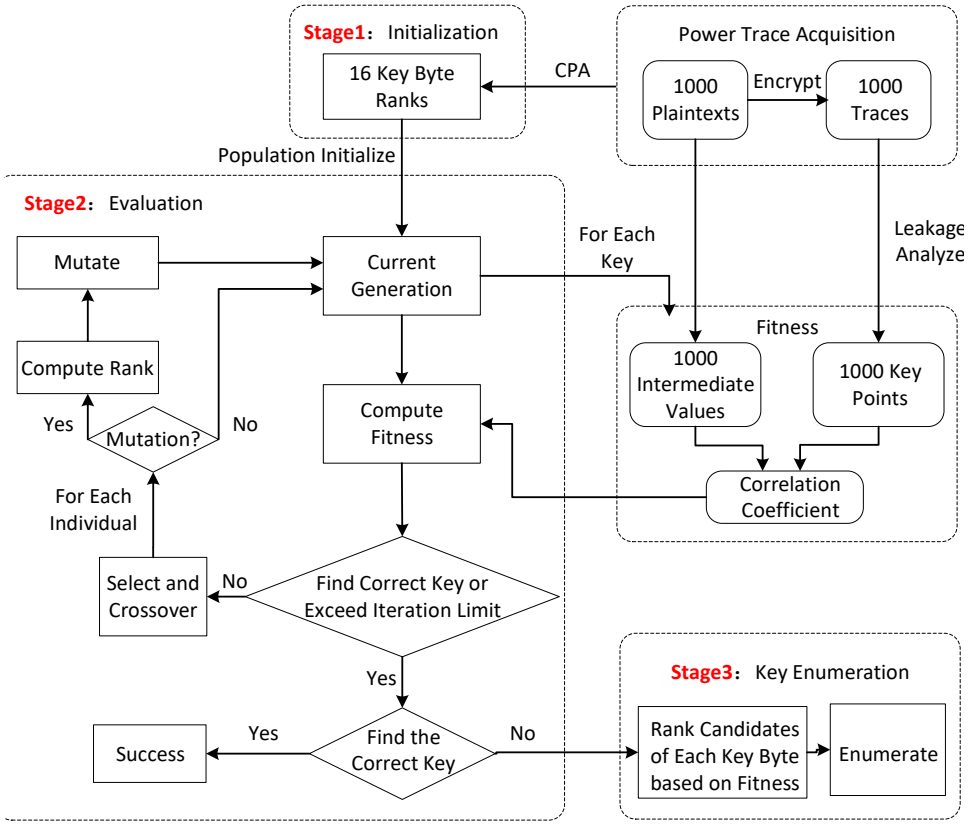


Fig. 1. The flow chart of our new framework for genetic-algorithm-based correlation power analysis.

values corresponding to each key byte, respectively. Then, 16 key byte rankings are obtained. Key bytes are chosen randomly from a set of candidates whose correlation coefficients are larger than the others (according to the ordered list). The size of the set is denoted as n_{init}, implying that these candidates are the Top n_{init} items of each rankings lists. 16 key bytes from 16 sets are combined to constitute the initial population. We give the outline of initialization stage based on rankings of candidates in Algorithm 2, denoted as **InitPopulationBR**(n_{pop}, n_{init}, T).

Algorithm 2 Initialization function **InitPopulationBR**(n_{pop}, n_{init}, T) based on CPA.

Input: size of population n_{pop}, selection range in the key ranking n_{init}, a set of power traces T.
Output: the initial population $\mathcal{P}$.

for $i := 1$ **to** 16 **do**
  for $j := 0$ **to** 255 **do**
    $\mathcal{K}_{list}[j].value := j$;
    $\mathcal{K}_{list}[j].cor :=$ **ComputeCor**(j, T);
  end for
  Sort($\mathcal{K}_{list}$);
  for $s := 1$ **to** n_{pop} **do**
    $\mathcal{P}[s][i] :=$ **RandomChoose**($\mathcal{K}_{list}, n_{init}$);
  end for
end for
return $\mathcal{P}$;

$\mathcal{P}$ is defined as a two-dimensional array, in which $\mathcal{P}[s][i]$ represents the i-th byte of the s-th individual. $\mathcal{K}_{list}$ indicates a list of 256 candidates of one key byte. Each element in $\mathcal{K}_{list}$ stores a candidate value $\mathcal{K}_{list}[j].value$ and its correlation coefficient $\mathcal{K}_{list}[j].cor$, which is calculated by **ComputeCor**(j, T) using the Hamming weight of j and the set of power traces T. **Sort**($\mathcal{K}_{list}$) arranges all elements of $\mathcal{K}_{list}$ from largest to smallest according to correlation coefficients. **RandomChoose**($\mathcal{K}_{list}, n_{init}$) selects a candidate value from the top n_{init} elements in $\mathcal{K}_{list}$ and initializes $\mathcal{P}[s][i]$.

The second stage implements the evolutionary process. Compared with Zhang et al.'s method, our new framework mutates individuals based on ranking lists of candidates depending on fitnesses. The specific steps are as follows.

- Compute all fitnesses of individuals in the population. Verify correctness of the key that has the highest fitness in current generation with a pair of plaintext and ciphertext. If it is the target one, attack is successfully done, else perform the following steps.
- Execute traditional selection and crossover operations, and obtain an intermediate population.
- In mutation operation, for each key byte of an individual decide whether to mutate according to a randomly generated decimal in $[0, 1]$ by comparing it to the mutation probability p_m.
- For each mutation key byte, rank 256 candidates according to their corresponding fitnesses, which are calculated with power traces and intermediate states produced by

keys with 1 byte traversing 256 values and the other 15 key bytes fixed to the original individual. Choose the top one as the mutation result of this byte.
- Repeat the steps above until the correct 16-byte key is found, or the number of generations reaches a threshold.

Zhang et al. gave experiments when they proposed GA-CPA, which show that the more correct key bytes, the higher the fitness of the individual. It can also be seen from Figure 3 that the more correct key bytes, the lower the guess entropy of a single key byte, and the easier it is for the correct key to become the direction of mutation operations. Therefore, we propose a mutation scheme, which is to traverse all possible values locally and select the optimal direction for mutation. Pseudo code of the ranking-based mutation is shown in Algorithm 3, denoted as **MutationBR**(K, p_m, T). K stands for an individual, and $K[i]$ is its i-th byte. $\mathcal{K}_{list}$ indicates a list of 256 candidates of K. Each element in $\mathcal{K}_{list}$ stores a candidate value $\mathcal{K}_{list}[j].value$ and its fitness $\mathcal{K}_{list}[j].fit$, which is calculated by setting $K[i]$ to $\mathcal{K}_{list}[j].value$ and maintaining the values of other key bytes. Unlike **ComputeCor**($K_{list}[j]$, T), **ComputeFit**(C_{temp}, T) outputs the correlation coefficient produced by a 16-byte key and the set of power traces T (i.e. fitness), rather than the correlation coefficient produced by a 1-byte candidate and power traces.

Algorithm 3 Mutation Function **MutationBR**(K, p_m, T) based on Ranking.

Input: individual about to mutate K, probability of mutation p_m, a set of power traces T.
Output: updated individual K.

MutationBR(K, p_m)
for $i := 1$ **to** 16 **do**
  $p :=$ **Random**$(0, 1)$;
  if $p < p_m$ **then**
    for $s := 1$ **to** 16 **do**
      $K_{temp}[s] := K[s]$;
    end for
    for $j := 0$ **to** 255 **do**
      $\mathcal{K}_{list}[j].value := j$;
      $K_{temp}[i] := j$;
      $\mathcal{K}_{list}[j].fit :=$ **ComputeFit**(K_{temp}, T);
    end for
    Sort$(\mathcal{K}_{list})$;
    $K[i] := \mathcal{K}_{list}[0].value$;
  end if
end for
return K;

For the third stage, ranked candidates of all key bytes with their fitnesses are feed into key enumeration algorithms if the optimal key recovered by the genetic algorithm is not the correct one. The difference between our proposal and the classic key enumeration algorithms is that candidates of a key byte are ranked according to the fitnesses which are calculated with the Hamming weight of a 128-bit intermediate state, instead of the correlation coefficient of a single byte.

We replace **InitPopulation()** and **Mutation()** in Algorithm 1 with **InitPopulationBR()** and **MutationBR()** respectively, and obtain our framework of genetic-algorithm-based correlation power analysis.

B. Advantage of the Key Enumeration Scheme in New Framework

In this section, we use guessing entropy [32] to illustrate the advantage of the key enumeration scheme in the new framework. The guessing entropy is a common measurement of side-channel attacks. In particular, an attack outputs a key guessing list $\boldsymbol{g} = \{g_1, g_2, ..., g_{|\mathcal{K}|}\}$ in decreasing order of probability with $|\mathcal{K}|$ being the size of the keyspace. So, g_1 is the most likely and $g_{|\mathcal{K}|}$ the least likely key candidate. The guessing entropy is defined as the average position of the correct key in $\boldsymbol{g}$. Obviously, the lower the guessing entropy, the higher the performance of the attack. In addition, we introduce a concept of average guessing entropy in this paper, which refers to the average of guessing entropies of all key bytes to be recovered.

Since the classic key enumeration algorithms focus on one key byte at a time, the power consumption of other key bytes contained in the power traces is regarded as noise, which has a great influence on the ranking of candidate values, especially when the number of traces is small. In our method, the Hamming weight of the other 15 key bytes is considered, which offsets the noise interference to a certain extent and makes the ranking of candidate value closer to the correct one. Taking AES-128 implemented in parallel for example, we show the advantage of our proposal compared with previous applications of key enumeration algorithms in CPA when processing on the same group of power traces in Figure 2. Solid lines indicate average guessing entropies and dotted lines stand for guessing entropies of the worst cases (the one that ranks the last in all key bytes). Red lines with circles indicate our method, and blue ones with triangles stand for CPA. Guessing entropies of our method are calculated by ranking the candidates of each key byte according to correlation coefficients between power consumption and Hamming weights of the whole intermediate states with the other 15 key bytes fixed to the correct ones. Guessing entropies of CPA are calculated by ranking the candidates of each key byte according to correlation coefficients between power consumption and Hamming weights of the corresponding intermediate bytes with the other 15 key bytes regardless. It is obvious that when the number of power traces exceeds a certain threshold, the guessing entropies of our method are lower than classic ones which implies that the key enumeration algorithm will achieve better efficiency.

In order to further illustrate how our method works, we give Figure 3 which shows that the average guessing entropies of key bytes reduce along with the number of correct key bytes increasing. These lines stand for the guessing entropies of a key byte, whose candidate values are ranked according to correlation coefficients between power consumption and Hamming weights of 128-bit intermediate states with x key bytes (random location) being correct and the other bytes being random wrong values. The optimal individual of the last generation of genetic algorithms is supposed as the one

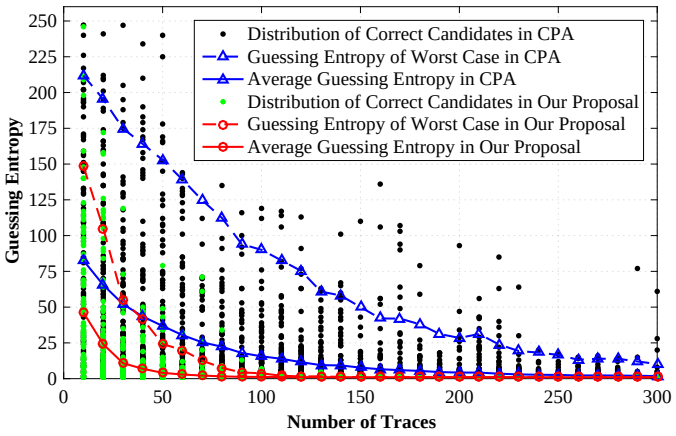


Fig. 2. Comparison of guessing entropy between our proposal and CPA.

that has the most number of correct key bytes, so we feed it with its ranking lists to key enumeration algorithms in order to improve the enumeration efficiency.

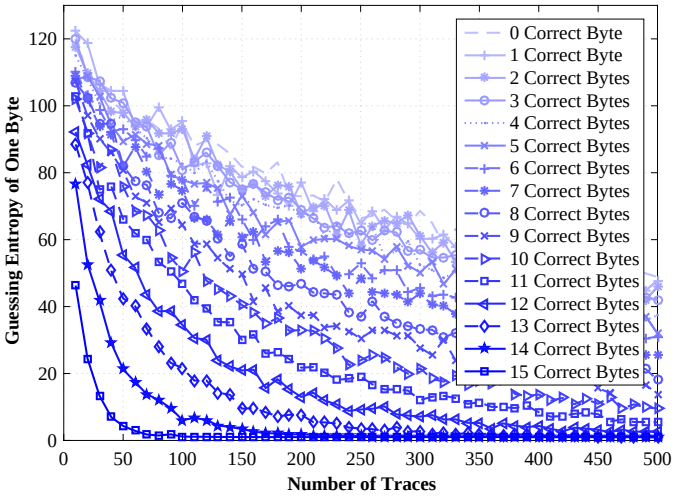


Fig. 3. The relation between the number of traces and the guessing entropy of one key byte calculated with different numbers of correct values for the other 15 key bytes.

IV. Operators and Parameters Selection

A. Operators Employed in the Genetic Algorithm

The mutation scheme used in our method is described in section III-A, so we omit here. For the truncation selection scheme, we test tournament selection, roulette wheel selection and truncation selection. Since the performance of the three schemes is similar, we adopt a relatively simple truncation option. The truncation selection randomly selects individuals from the top $n_{pop} * p_t$ individuals in fitness, where n_{pop} stands for the size of population and p_t is called truncation probability. Byte-wise crossover [33] is employed to recombine individuals, as shown in Algorithm 4. Key bytes with corresponding positions in the two parent individuals are exchanged with probability p_c, which is called crossover probability. Since our method generates relatively high-quality key bytes

Algorithm 4 Byte-wise Crossover Scheme.

Input: two individuals about to be recombined K_1, K_2, crossover probability p_c.
Output: two updated individuals K_1, K_2.

```
for i := 1 to n_word do
    p := Random(0, 1);
    if p < p_c then
        temp := K_1[i]; K_1[i] := K_2[i]; K_2[i] := temp;
    end if
end for
Return K_1, K_2;
```

by means of initialization and mutation operations, the Byte-wise crossover scheme is used to ensures that the internal structure of key bytes is not destroyed while generating new individuals. Therefore, parameters we need to determine are the size of initialization set for each key byte n_{init}, the size of population n_{pop}, the truncation probability p_t,the crossover probability p_c and the mutation probability p_m.

B. Parameters Used in Initialization Stage

Noting that the mutation operation on a single byte needs to traverse and calculate the fitness of 256 candidate values, which is a huge amount of calculation, we empirically set n_{pop} to be 6. Experimental results given later confirms that this value ensures the effective execution of the algorithm while having reasonable computational complexity.

Before determining n_{init}, we prefer to give a threat model where the attacker can obtain only a limited number of traces, which is enough for generating a high-quality initial population but not enough for normal CPA. First, We conduct simulation experiments, in which power consumption of S-box operations is simulated by adding Gaussian noises with standard deviation being $\sigma = 3.0$. Normal CPA is applied on different group of traces. The number of traces ranges from 10 to 1000 by a step of 10, and the experiments is conducted for 100 times. The rankings of correct values for all the key bytes are collected during each execution. Figure 4 shows the relation between the number of traces and the guessing entropies of all key bytes. The green dots stand for the rankings of the correct value of one key byte. The red line denotes the average guessing entropy of all the 16 key bytes. The blue line denotes the guessing entropy of the worst case, i.e. the average location of the correct value that ranks lowest among all the 16 guessing lists. In fact, the guessing entropy of the worst case determines whether normal CPA will recover all the 16 key bytes. The blue line in Figure 4 indicates that when the number of power traces reaches 500, normal CPA still cannot guarantee the ranking of each correct value to be No. 1, i.e. the attack fails.

Then, based on the data shown in 4, we get Figure 5, which shows the ratio of correct values ranking in top i ($i \in \{1, 5, 10, 20, 40, 60\}$). Lines in blue, red, green, cyan, magenta and yellow stand for $i = 1/5/10/20/40/60$, respectively. From Figure 5, we know that when the number of power traces is greater than 500, the probability that the correct

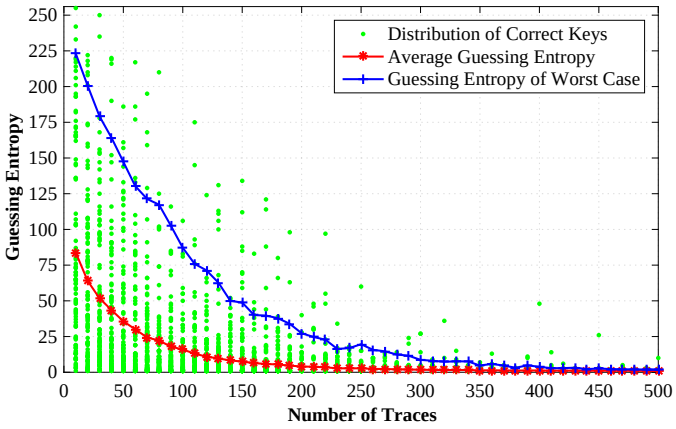


Fig. 4. The relation between the number of traces and guessing entropy.

value ranks in top $5/10/20/40/60$ is 100%. Furthermore, we derive and calculate the probability that each correct value to be selected into the initial population when using our initialization stage with $n_{init} = 1/5/10/20/40/60$, respectively, and show them in Figure 6. The probability of a correct value to be in the initial population by randomly generation is $1-(1-1/256)^{n_{pop}} \approx 0.023$. The initialization stage aims at covering correct values of key bytes as many as possible in the initial population, in order to generate high-fitness individuals in the subsequent generations. The experimental results in Figure 6 confirm that our method improve the probability that initialize the population with correct values of key bytes, even when the number of traces is 10 (20 for $n_{init} = 1$).

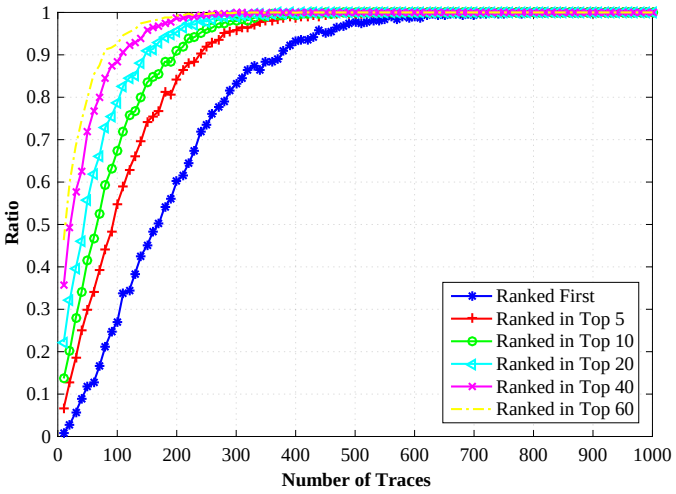


Fig. 5. The relation between the number of traces and the ratio of correct key byte in top 1/5/10/20/40/60.

Finally, we calculate the probability of a correct value in the initial population p_{init} when $n_{init} = 1/5/10/20/40/60$ and $n_{pop} = 2/4/6/.../20$, respectively. The number of traces is set to 220, which is the boundary value between initializing with Top 1 and Top 5 according to Figure 6. From the results shown in Figure 7, we know that when $n_{pop} \geq 6$, p_{init} with $n_{init} = 5$ is the largest (highlight in red). Therefore, we determine the value of n_{init} to be 5. It is an appropriate value for n_{init} to cover the correct values as many as possible and not too much

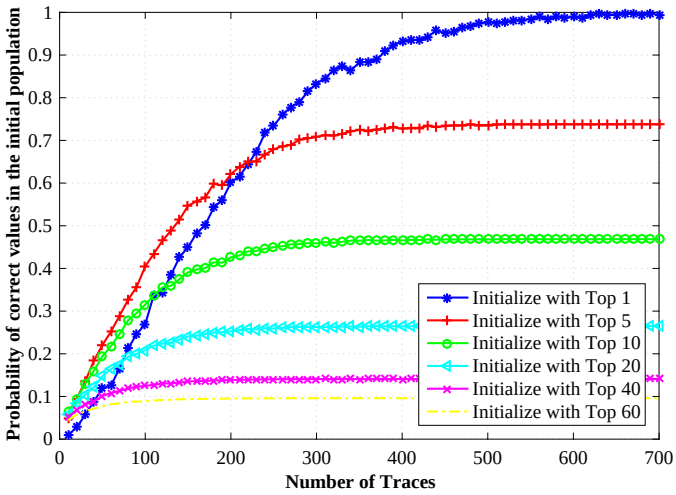


Fig. 6. The relation between the number of traces and the probability of the correct value in the initial population.

to reduce the probability of the correct values to be chosen in the initial population.

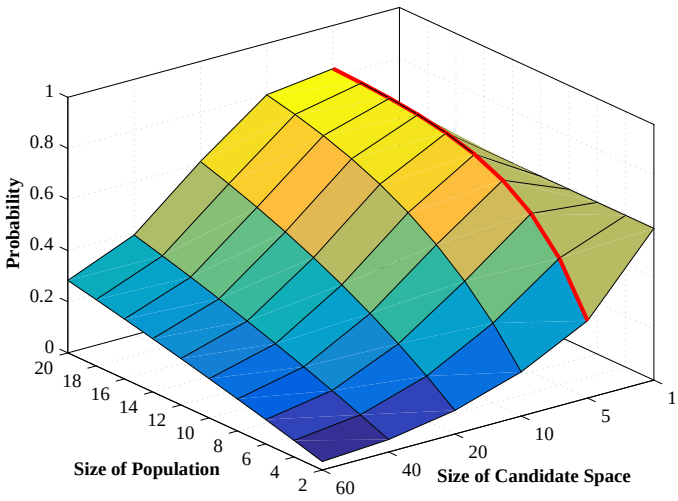


Fig. 7. The probability of the correct value in the initial population with different n_{init} and n_{pop}.

C. Parameters Used in Genetic Algorithm

Since there are only 6 individuals in a population, we determine the truncation probability $p_t = 50\%$ to ensure the choice space while abandoning weaker individuals. The crossover probability p_c and the mutation probability p_m influence the function of genetic algorithms together, so we discuss them at the same time. We test each value of (p_c, p_m) by performing our new framework of GA-CPA on different group of traces for 100 times respectively. Considering that the recombination of two individuals are the same for p_c and $1-p_c$ in our new framework, we have p_c ranging from 0.05 to 0.5 by a step of 0.05. p_m is generally less than 0.1, so we have p_m ranging from 0.01 to 0.1 by a step of 0.01. 220 traces are simulated with Gaussian noise whose standard deviation is $\sigma = 3.0$. The number of recovered key bytes are collected

when the algorithm converge (generally after 50 generations), and averaged for each value of (p_c, p_m). Figure 8 shows the relation between (p_c, p_m) and the number of recovered key bytes. Obviously, p_m is the main factor that influences the function of the new framework. Since the number of recovered key bytes is nearly the same for $p_m \in [0.05, 0.1]$ and a larger p_m means a higher computation complexity, we chose $(p_c, p_m) = (0.5, 0.05)$.

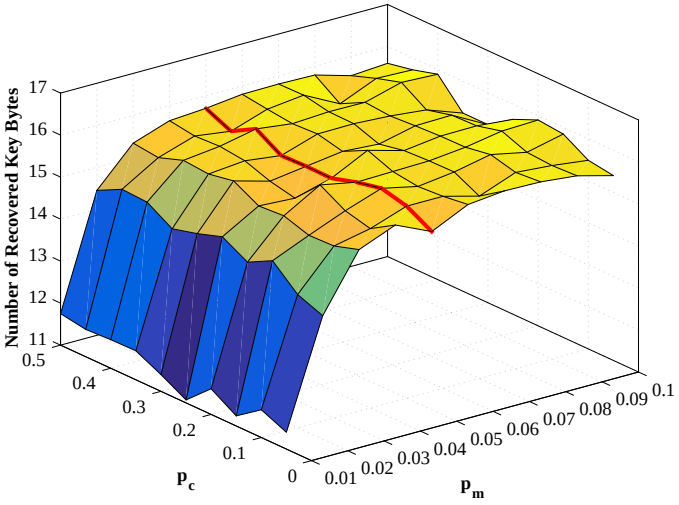


Fig. 8. The relation between the number of recovered key bytes and (p_c, p_m) at the 50th generation.

In practice, parameters mainly depend on cryptographic algorithms and the framework of attacks, so the parameters determined in this section are applicable to the attacks mounted in the following section.

V. Experiments and Efficiency

A. Comparison of Convergence Speed

The convergence rate directly determines the maximal threshold number of generations, and then affects the computation complexity. Therefore, we perform GA-CPA and our new framework on the same group of simulated traces (standard deviation of noise $\sigma = 3.0$) to compare their convergence rates. The experiment is repeated for 100 times, and power traces are randomly generated in each time. Since the average number of evaluation times is $n_{pop} \times 16 \times p_m \times 2^8 + n_{pop} = 6 \times 16 \times 0.05 \times 256 + 6 = 1234.8$, we set the number of individuals in comparison experiments based on normal GA-CPA to 1300. Thus, the convergence rate can be compared based on the number of generations. Numbers of recovered key bytes are collected and averaged for every ten generations (≤ 140). Figure 9 shows the relationship between the number of generations and the number of recovered key bytes for both methods. The red line indicates our proposal, and the blue one stands for the traditional GA-CPA.

From the figure, we know that our proposal converges after 50 generations, while the traditional one's convergence generation is nearly 100. Apparently, our method has a higher convergence rate, and consequently a lower computation complexity.

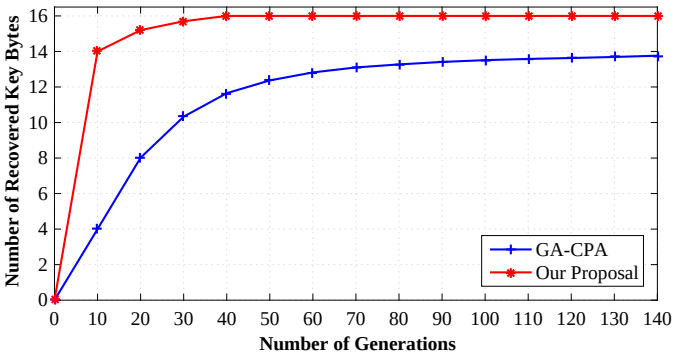


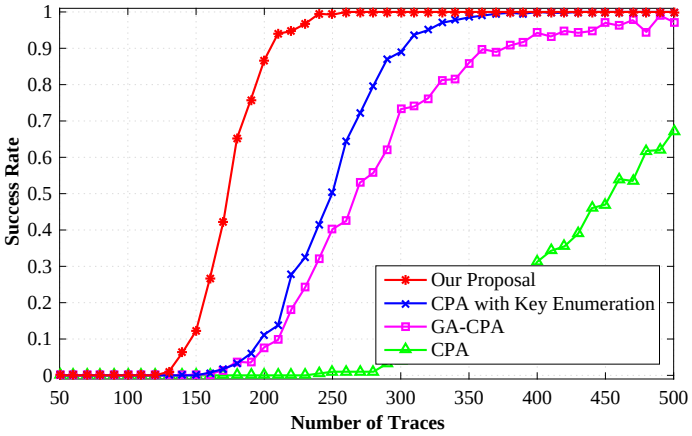
Fig. 9. The comparison of convergence rate.

B. Efficiency Comparison with CPA and GA-CPA in Simulation Experiment

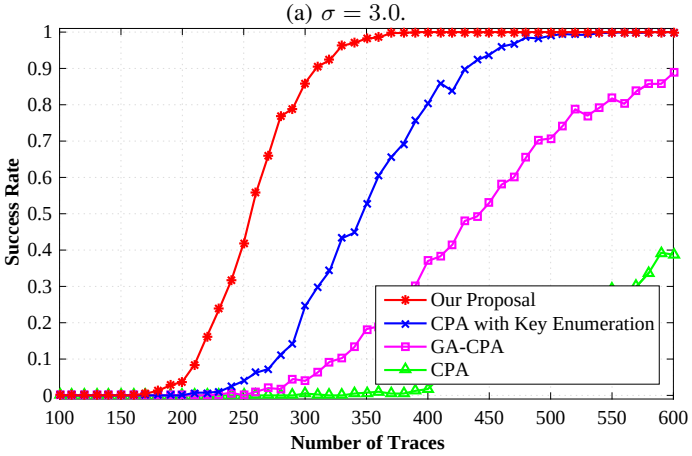
Furthermore, we conduct experiments to compare the efficiency of our proposal with CPA (with/without key enumeration) and GA-CPA by mounting attacks on the same group of traces, whose size ranges from 10 to 1000. We employ success rate to measure the performance of the two methods in this section. The success rate is defined as the average empirical probability that an attack recovered the target key. Each experiment is repeated for 400 times with different group of randomly simulated traces (standard deviation of noise $\sigma = 3.0$). We apply the key enumeration algorithm proposed by Veyrat-Charvillon et al. [6], and set the maximum number of enumerations to be 2^{20}. The parameters of our proposal are described in section IV, and we chose appropriate parameters for the traditional GA-CPA with similar experiments. The size of populations chosen for the GA-CPA is 1000, and the crossover probability and mutation probability are $(p_c, p_m) = (0.5, 0.12)$. According to Figure 9, the maximal threshold numbers of generations of the two methods are 50 and 100 respectively. Their success rates and computation costs are displayed in Figure 10(a), Figure 11(a) and Figure 12(a). The computation cost is estimated as the average number of calculations for correlation coefficients taken by our proposal, CPA and GA-CPA (shown in Figure 11(a)), and enumeration times taken by our proposal and CPA with key enumeration algorithms (shown in Figure 12(a)). In addition, experiments on traces with $\sigma = 5.0$ are also conducted, and the results are displayed in Figure 10(b), Figure 11(b) and Figure 12(b).

The experimental results show that:

- For $\sigma = 3.0$, it requires 210 traces for our proposal to achieve 90% success rate, and the corresponding computation costs are about 4.30×10^4 calculations of correlation coefficients and 0.68×10^5 enumerations. With the same number of traces, the success rates of CPA with key enumeration and GA-CPA are about 14% and 10%, and the corresponding correlation coefficient calculation times are about 0.41×10^4 and 9.81×10^4, respectively. The enumeration times of CPA are 9.28×10^5. The average timing overhead of successful attacks (with 210 traces and $\sigma = 3.0$) based on our method is less than 1 minutes on a PC (Intel Core E7500@2.93GHz 2.00GB).
- For $\sigma = 5.0$, it requires 310 traces for our proposal to

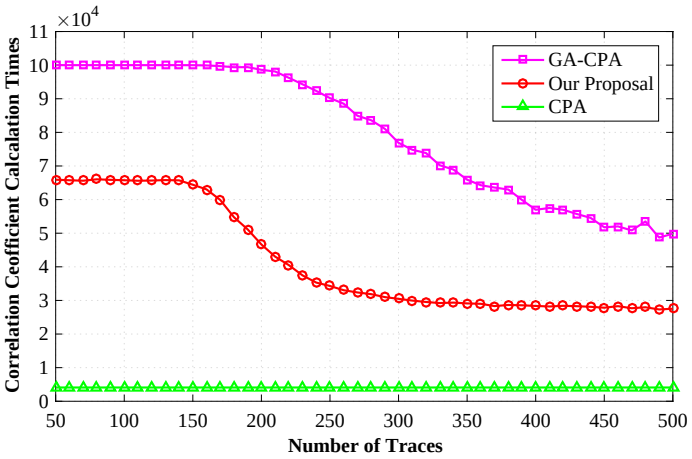


(a) $\sigma = 3.0$.

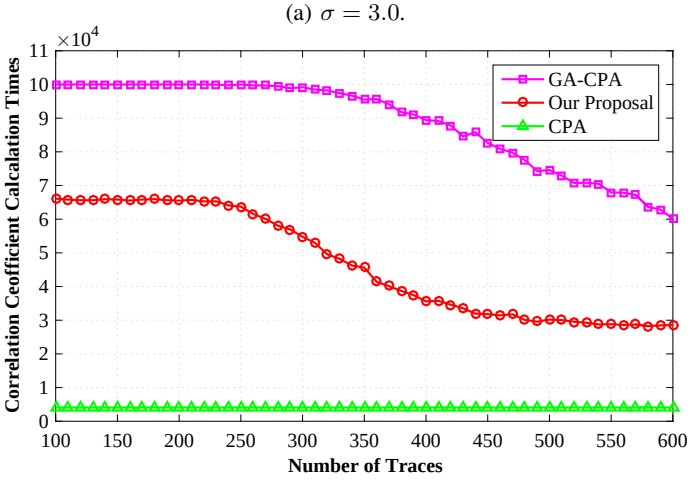


(b) $\sigma = 5.0$.

Fig. 10. The comparison of success rates among our proposal, CPA (with/without key enumeration) and GA-CPA.



(a) $\sigma = 3.0$.



(b) $\sigma = 5.0$.

Fig. 11. The comparison of correlation coefficient calculation times among our proposal, CPA and GA-CPA.

achieve 90% success rate, and the corresponding computation cost is about 5.30×10^4 calculations of correlation coefficients and 1.16×10^5 enumerations. With the same number of traces, the success rates of CPA with key enumeration and GA-CPA are about 30% and 6%, and the corresponding computation costs are about 1.04×0.41^4 and 9.85×10^4, respectively. The enumeration times of CPA are 7.88×10^5.

From the experimental results we know that:

- With the same number of traces, the success rate of our method is much more than the others, calculations of correlation coefficients are less than GA-CPA.
- Our method performs much better than the CPA when combined with key enumeration algorithms.
- Our method has trade off between number of traces and offline computation time. Although its computation costs are larger than CPA, it is still affordable for a key recovery attack.

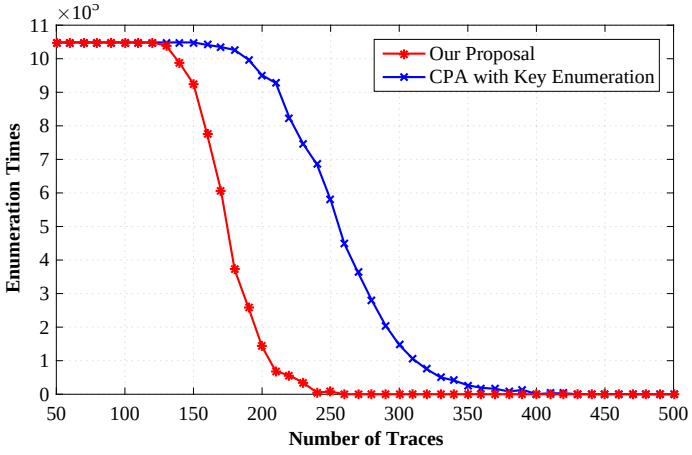
C. Efficiency Comparison with CPA and GA-CPA in Real Experiments

For real experiments, we encrypt random plaintexts with a fixed key using AES-128 implemented on SAKURA-G provided officially. 2000 power traces are acquired and stored with corresponding plaintexts and ciphertexts. Since registers are embedded before the S-box operations in this hardware implementation, our attacks are focused on the intermediate states before S-box operations of the last round and the ciphertexts. A time window that contains information of the target intermediate state is preselected and the points in it are averaged to obtain one value.

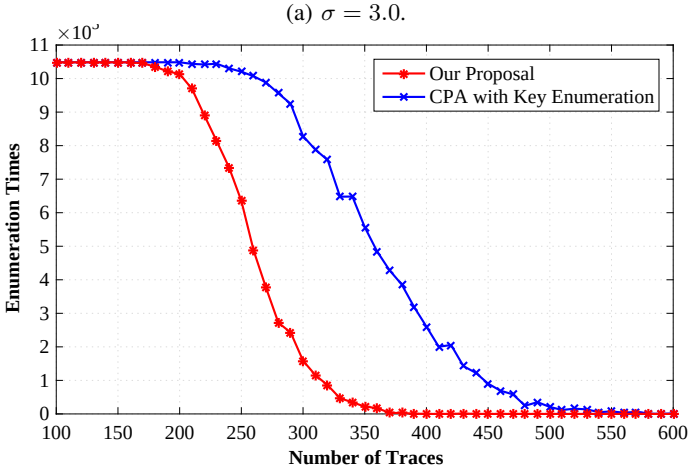
We mount attacks based on the three methods with the same group of traces respectively. The number of traces ranges from 20 to 1000 with a step of 20. Since we take AES-128 as an example to discuss parameter selection in the previous section, we will continue to use these parameters to do real experiments on the same algorithm in this section.

For CPA, we calculate the correlation coefficients corresponding to all the values of each key byte. Figure 13 shows the experimental results of the one that requires the largest number of traces to identify the correct value. The blue lines stand for wrong guesses of this key byte and the red one stands for the correct candidate. Obviously, the minimal threshold for attacks based on correlation coefficients of single key byte is 500.

For GA-CPA, we collect fitnesses of the best individual in each generation. Figure 14 shows the results of experiments with randomly initialized populations. The red line stands for the fitness of the correct key. The blue one stands for the



(a) $\sigma = 3.0$.



(b) $\sigma = 5.0$.

Fig. 12. The comparison of enumeration times between our proposal and CPA with key enumeration.

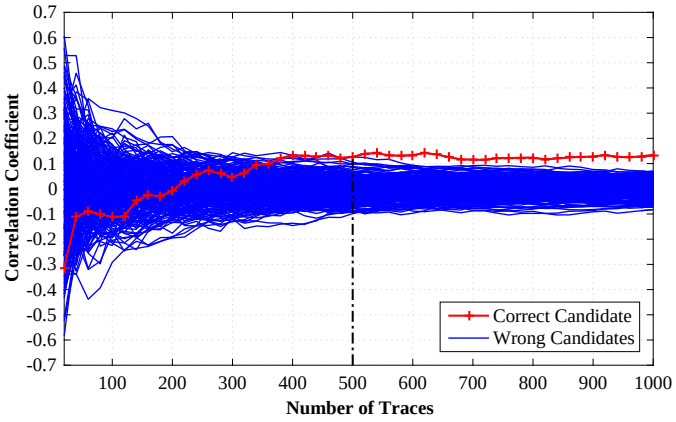


Fig. 13. The relation between correlation coefficients and the number of traces for a single key word in classic CPA.

fitness of the best individual obtained by GA-CPA. The green ones stand for the best fitnesses of each generations. From this figure, we know that even if there are enough traces for GA-CPA to achieve a high success rate, premature convergence will still occur due to poor initial populations.

For our method, we also collect the fitness of the best individual in each generation. Figure 15 shows the experimental results with randomly initialized populations. The red

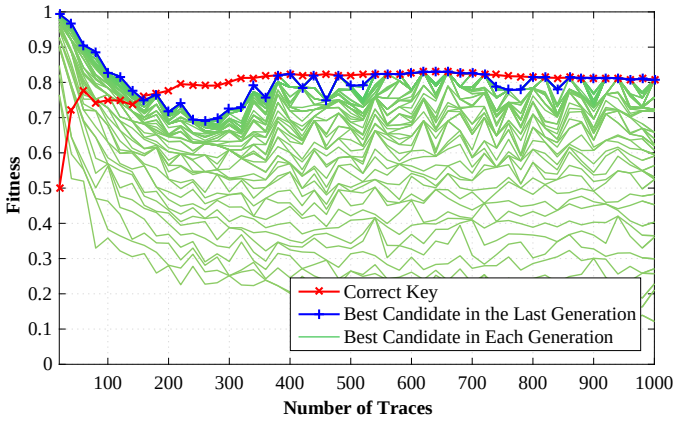


Fig. 14. The relation between the number of traces and fitnesses of the best individuals in the first 100 generations in existing GA-CPA.

line indicates the fitness of the correct key. The blue one indicates the fitness of the best individual obtained in the last generation, and the cyan one stands for fitness of the outputs after key enumeration algorithms. The green ones indicate the best fitnesses of each generations. The cyan line coincides with the red one at 180, telling that our proposal is compatible with enumeration algorithms, and is significantly improved.

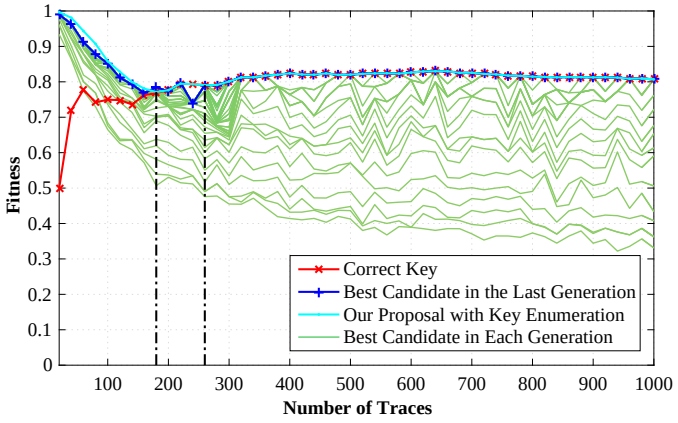


Fig. 15. The relation between the number of traces and fitnesses of the best individuals in the first 50 generations in our proposal.

From these figures, we know that the minimal number of traces required for CPA is 500, for GA-CPA is at least 380 and for our method is only 180. Ratios between our method and the other two are 16.00% and 47.36%, respectively. Note that, at the point $Num\_traces = 420$, the fitness of the best key obtained by GA-CPA is lower than the correct one. In fact, similar phenomenons occur almost in each experiment of GA-CPA at different points. The major reason is that GA-CPA converges before recovering all the key words, and the random local search based on word-wise mutation is not effective enough to generate the correct one after the convergence.

D. Efficiency Comparison with Hill-Climbing-based Method on DPA Contest v1 Dataset

DPA contest [34] is the first international benchmark reference for side-channel attacks, which provides power consump-

tion traces of hardware implementation of DES [35]. The DPA contest v1 dataset was built with parallel implementation of DES, which is suitable for heuristic algorithms. This dataset comprises 81089 power traces of 20003 samples, which were collected during encryptions with random plaintexts and a fixed key.

The hill-climbing-based approach proposed by Clavier et al. [27] is a classic application of heuristic algorithms in side-channel analysis, and is the winning proposal to the first edition of the DPA contest. The authors considered full 56-bit guesses of the main key to optimally exploit the side-channel leakage. They used a maximum likelihood based distinguisher as the objective function, and considered the subkeys (K_1, K_{16}) of the first and the last round in order to cover all bits in the main key.

To compare our proposal with the hill-climbing-based method and GA-CPA, we apply our framework to the DPA contest v1 dataset. We change the fitness function to the maximum likelihood based distinguisher, and adjust basic definitions and operations to handle DES:

- An individual is defined as the 16 6-bit key words in K_1 and K_{16}.
- For the initial process, rankings of candidates for each key words is calculated by the maximum likelihood based distinguisher instead of classic CPA. The one with the lowest value ranks first.
- For mutation operations, candidates of each key word are ranked according to fitnesses produced by fixing the other key words in the individual. The one with the lowest fitness is the mutation result. After all key words in K_1 finish mutation, 40 corresponding bits in K_{16} are set according to K_1. Then, K_{16} are processed in the same way.
- For key enumeration, ranked candidates with fitnesses of all key words in both K_1 and K_{16} are feed into key enumeration algorithms. Notice that the related 40 bits of K_1 and K_{16} in each enumeration are not always the same. But if K_1 and K_{16} are correct, the related 40 bits must be the same.

We conduct experiments with the source code provided by the author of [27]. The parameters of our method are set as $p_c = 0.5$, $p_m = 0.3$, $n_{init} = 5$. We set the maximal threshold number of generation to be 200, and the maximum number of enumerations to be 2^{20}. 100 runs of each method are carried out to get min, average and max number of traces required to recover the main key. According to the evaluation method of the competition, for each run, a random set of $n = 30$ traces is built to launch the first attack. Then, attacks are conducted by adding an extra random trace to the set continuously until all attacks are successful with the number of traces being $n \in \{n, n+1, ..., n+99\}$. n is considered to be the number of traces needed to recover the key (according to the requirements of DPA contest). Table I shows the comparison of n between our proposal and the hill-climbing-based method.

As for the computation cost, we run both methods 100 times with numbers of random traces ranging from 30 to 190 by a step of 2, and record the time used. The experiments are carried out on a PC with Intel(R) Core(TM) i7-9750H @2.59 GHz 32GB RAM. Figure 16 shows the comparison of running time. The blue line stands for the hill-climbing-based method, the green line stands for normal GA-CPA and the red one stands for our proposal. The time is calculated by seconds.

TABLE I
COMPARISON OF OUR PROPOSAL AND THE HILL-CLIMBING-BASED METHOD

Method	Min	Average	Max
GA-CPA	35	61.12	116
Our Framework of GA-CPA	30	40.76	71
Hill-Climbing-based Approach [27]	30	42.42	94

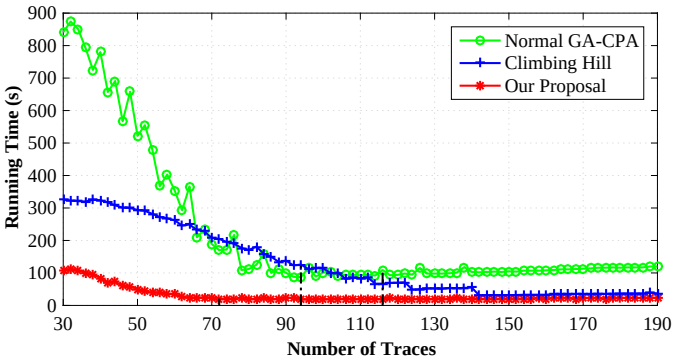


Fig. 16. The comparison of computation cost between our proposal and the hill-climbing-based method.

Compared with the hill-climbing-based method, our framework benefits from selection and crossover operations, making it easier for correct key values to be aggregated in a single individual. The convergence rate of the algorithm is accelerated, which reduces the computation complexity by more than 50%. In addition, the combined effect of crossover and mutation improves the flexibility of individual evolution, and to a certain extent avoids the loss of the correct key due to the fixed route in the hill-climbing algorithm. Therefore, our method requires nearly 2 less power traces, especially with the function of key enumeration algorithms.

VI. CONCLUSION

In this approach, we discuss the imperfections of using genetic algorithm to solve the correlation coefficient optimization problem, and put forward a GA-CPA framework which not only improves the convergence rate of genetic algorithm, but also reduces the number of traces required in the power analysis on cryptographic algorithms implemented with parallel S-boxes and large noise. Besides, a key enumeration algorithm is involved in the framework to improve the success rate. Experimental results show that our method performs better than CPA with key enumeration algorithm, the classic GA-CPA and the hill-climbing-based method. We expect that in addition to climbing hill and genetic algorithms, more heuristic approaches will be integrated into SCA, especially with fault attacks to improve its efficiency further.

REFERENCES

[1] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Annual International Cryptology Conference*. Springer, 1996, pp. 104–113.

[2] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 13–28.

[3] K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," in *International Workshop on Fast Software Encryption*. Springer, 2003, pp. 206–222.

[4] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2008*, E. Oswald and P. Rohatgi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 426–442.

[5] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.

[6] N. Veyrat-Charvillon, B. Gérard, M. Renauld, and F.-X. Standaert, "An optimal key enumeration algorithm and its application to side-channel attacks," in *International Conference on Selected Areas in Cryptography*. Springer, 2012, pp. 390–406.

[7] D. P. Martin, J. F. O'Connell, E. Oswald, and M. Stam, "Counting keys in parallel after a side channel attack," in *Advances in Cryptology – ASIACRYPT 2015*, T. Iwata and J. H. Cheon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 313–337.

[8] R. Poussier, F.-X. Standaert, and V. Grosso, "Simple key enumeration (and rank estimation) using histograms: an integrated approach," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 61–81.

[9] L. David and A. Wool, "A bounded-space near-optimal key enumeration algorithm for multi-subkey side-channel attacks," in *Cryptographers Track at the RSA Conference*. Springer, 2017, pp. 311–327.

[10] N. Veyrat-Charvillon, B. Gérard, and F.-X. Standaert, "Security evaluations beyond computing power," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 126–141.

[11] A. Duc, S. Faust, and F.-X. Standaert, "Making masking security proofs concrete," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 401–429.

[12] C. Glowacz, V. Grosso, R. Poussier, J. Schüth, and F.-X. Standaert, "Simpler and more efficient rank estimation for side-channel security assessment," in *International Workshop on Fast Software Encryption*. Springer, 2015, pp. 117–129.

[13] R. Poussier, V. Grosso, and F.-X. Standaert, "Comparing approaches to rank estimation for side-channel security evaluations," in *International Conference on Smart Card Research and Advanced Applications*. Springer, 2015, pp. 125–142.

[14] G. Hospodar, E. Mulder, B. Gierlichs, I. Verbauwhede, and J. Vandewalle, "Least squares support vector machines for side-channel analysis," *Center for Advanced Security Research Darmstadt*, pp. 99–104, 2011.

[15] T. Bartkewitz, "Leakage prototype learning for profiled differential side-channel cryptanalysis," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1761–1774, 2016.

[16] H. D. Tsague and B. Twala, "Reverse engineering smart card malware using side channel analysis with machine learning techniques," in *Big Data, 2016 IEEE International Conference on*. IEEE, 2016, pp. 3713–3721.

[17] Z. Martinasek and V. Zeman, "Innovative method of the power analysis," *Radioengineering*, vol. 22, no. 2, pp. 586–594, 2013.

[18] E. Cagli, C. Dumas, and E. Prouff, "Convolutional neural networks with data augmentation against jitter-based countermeasures," in *International Conference on Cryptographic Hardware and Embedded Systems-CHES 2017*. Springer, 2017, pp. 45–68.

[19] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 3, pp. 148–179, 2019.

[20] S. Picek, A. Heuser, A. Jovic, and A. Legay, "Climbing down the hierarchy: hierarchical classification for machine learning side-channel attacks," in *International Conference on Cryptology in Africa*. Springer, 2017, pp. 61–78.

[21] A. Heuser, S. Picek, S. Guilley, and N. Mentens, "Lightweight ciphers and their side-channel resilience," *IEEE Transactions on Computers*, 2017.

[22] L. Lerman, S. F. Medeiros, G. Bontempi, and O. Markowitch, "A machine learning approach against a masked AES," in *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, ser. Lecture Notes in Computer Science, A. Francillon and P. Rohatgi, Eds., vol. 8419. Springer, 2013, pp. 61–75.

[23] L. Lerman, G. Bontempi, and O. Markowitch, "Side channel attack: an approach based on machine learning," *Center for Advanced Security Research Darmstadt*, pp. 29–41, 2011.

[24] S. Picek, A. Heuser, and S. Guilley, "Template attack versus bayes classifier," *J. Cryptographic Engineering*, vol. 7, no. 4, pp. 343–351, 2017.

[25] Z. Zhang, L. Wu, A. Wang, Z. Mu, and X. Zhang, "A novel bit scalable leakage model based on genetic algorithm," *Security and Communication Networks*, vol. 8, no. 18, pp. 3896–3905, 2015.

[26] Y. Ding, Y. Shi, A. Wang, Y. Wang, and G. Zhang, "Block-oriented correlation power analysis with bitwise linear leakage: An artificial intelligence approach based on genetic algorithms," *Future Generation Computer Systems*, vol. 106, pp. 34–42, 2020.

[27] C. Clavier and D. Rebaine, "A heuristic approach to assist side channel analysis of the data encryption standard," in *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, ser. Lecture Notes in Computer Science, P. Y. A. Ryan, D. Naccache, and J. Quisquater, Eds., vol. 9100. Springer, 2016, pp. 355–373.

[28] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1975.

[29] J. Sampson and A. Brindle, "Genetic algorithms for function optimization," 1981.

[30] K. De Jong, "An analysis of the behavior of a class of genetic algorithms," *Dissertation Abstracts International*, vol. 36, 01 1975.

[31] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm i. continuous parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 25–49, 1993.

[32] J. L. Massey, "Guessing and entropy," in *Proceedings of 1994 IEEE International Symposium on Information Theory*. IEEE, 1994, p. 204.

[33] Y. Ding, A. Wang, and S. M. YIU, "An intelligent multiple sieve method based on genetic algorithm and correlation power analysis," Cryptology ePrint Archive, Report 2019/189, 2019, https://eprint.iacr.org/2019/189.

[34] T. P. S. R. Group, "DPA contest (1^{st} edition) (2008-2009)," 2008, http://www.dpacontest.org/.

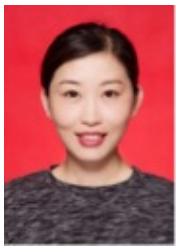
[35] National Bureau of Standards, "Data sncryption standard," *Federal Information Processing Standards Publications*, 1977.

An Wang was born in 1983. He received the Ph.D. degree from Shandong University in 2011. From 2011 to 2015, he held a postdoctoral position at Tsinghua University. He is currently with the Beijing Institute of Technology. His main research interests include side-channel analysis, embedded systems, and cryptographic implementation.

Yuan Li was born in 1997. She is currently in the third year of a master's degree at Beijing Institute of Technology. Her research interest is side-channel attack.

Yaoling Ding was born in 1987. She received her PH.D. degree from Tsinghua University in 2019. She currently holds a postdoctoral position at Beijing Institute of Technology. Her research interests include side-channel attack and cryptanalysis of block cipher.

Liehuang Zhu is a professor in the Department of Computer Science at Beijing Institute of Technology. He is selected into the Program for New Century Excellent Talents in University from Ministry of Education, P.R. China. His research interests include internet of things, cloud computing security, internet and mobile security.



Yongjuan Wang was born in 1982. She received her PH.D. degree in Information Engineering University in 2009. From 2013 to 2015, she worked as a post doctor in 58th Research Institute of general staff. She currently works in Strategic Support Force Information Engineering University. Her main research interests include cryptographic analysis, side-channel analysis and cyberspace security.