An Intelligent Multiple Sieve Method Based on Genetic Algorithm and Correlation Power Analysis

Yaoling Ding[1], An Wang[2,3](✉), and Siu Ming YIU[4]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
dyl13@mails.tsinghua.edu.cn
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] School of Computer Science, Beijing Institute of Technology, Beijing 100081, China
wanganl@bit.edu.cn
[4] Department of Computer Science, The University of Hong Kong, Hong Kong, China
smyiu@cs.hku.hk

Abstract. Correlation power analysis (CPA) is widely used in side-channel attacks on cryptographic devices. Its efficiency mostly depends on the noise produced by the devices. For parallel implementations, the power consumption during the S-box operation contains information of the whole intermediate state. When one S-box is analyzed by CPA, the others are regarded as noise. Apparently, the information of the remained S-boxes not only is wasted, but also increases the complexity of analysis. If two or more S-boxes are considered simultaneously, the complexity of exhaustive search on the corresponding key words grows exponentially. An optimal solution is to process all the S-boxes simultaneously and avoid traversing the whole candidate key space. Simple genetic algorithm was used by Zhang et al. to achieve this purpose. While, premature convergence causes failure in recovering the whole key, especially when plenty large S-boxes are employed in the target primitive, such as AES.

In this paper, we study the reason of premature convergence, and propose the multiple sieve method which overcomes it and reduces the number of traces required in correlation power attacks. Operators and the corresponding parameters are chosen experimentally with respect to a parallel implementation of AES-128. Simulation experimental results show that our method reduces the number of traces by 63.7% and 30.77% compared to classic CPA and the simple genetic algorithm based CPA (SGA-CPA) respectively when the success rate is fixed to 90%. Real experiments performed on SAKURA-G confirm that the number of traces required to recover the correct key in our method is almost equal to the minimum number that makes the correlation coefficients of correct keys outstanding from the wrong ones, and is much less than classic CPA and SGA-CPA.

Keywords: Multiple sieve · Genetic algorithm · Correlation power analysis · Parallel implementation · AES.

1 Introduction

Side-channel analysis plays an important role in evaluating the performance of cryptographic devices. Many efficient methods are proposed with respect to it, such as correlation power analysis (CPA) [3], template attack [5], collision attack [20], mutual information analysis [7], and so on. Correlation power analysis [3] is a classic approach in the scope of statistical power analysis against cryptographic devices. It improves the former works [12, 17] and inspired many new approaches [13, 23]. A leakage model based on Hamming distance was proposed in that approach. The correlation coefficients between power samples and Hamming distances of intermediate states were calculated to identify the correct key. This leakage model was also applied to partial bits amongst a machine word, when the candidate key space (determined by the architecture of the device) was too large to be searched exhaustively. Experimental results showed that the correlation coefficients decreased significantly and the number of power traces increased a lot in the partial leakage model.

In recent years, artificial intelligence (AI) technology is widely used in side-channel analysis. In 2011, machine learning was firstly used for side-channel analysis by Hospodar et al. [11]. Least square support vector machine was employed to classify intermediate bit value in comparison to template attack. Later, Lerman et al. [14] introduced profiling attacks based on three different algorithms, namely random forest, support vector machine (SVM) and self-organizing maps. Each of the attack outperformed template attack when applied on triple data encryption standard. In 2012, multi-class SVM was used to deal with the attack on multi-bit value (Hamming weight model) [9]. It is improved by Bartkewitz et al. [2] with a new multi-class classification strategy. SVM was also used as a preprocessing tool for feature selection in the same approach. In 2013, Lerman et al. [15] performed an attack requiring only partial knowledge for profiling. A method based on neural network was proposed by Martinasek et al. [16] to recover the key of AES with only one trace in the same year. Bartkewitz [1] introduced leakage prototype learning for profiled differential side-Channel cryptanalysis in 2016. Cagli et al. [4] conducted an profiling attack based on convolutional neural networks and data augmentation techniques to deal with jitter-based countermeasures. In 2017, Picek et al. [18] proposed a hierarchical classification approach for side-channel analysis. Resistance of lightweight ciphers to side-channel analysis was evaluated by Heuser et al. [8] using various machine learning attacks. Parameters involved in side-channel analysis based on machine learning algorithms were studied in [19]. Besides, machine learning algorithms were also employed to design countermeasures against side-channel analysis [21]. And Tsague at al. [22] introduced a method combining side-channel analysis and machine learning techniques to recover malware program executed on a smart card. So far, almost all machine learning based power analysis are profiling attacks.

While, Zhang et al. [23] introduced a non-profiling AI-based power analysis employing simple genetic algorithm to turn the key searching problem into an correlation coefficient optimization problem. The authors claimed that their method (SGA-CPA) was applicable to various cryptographic algorithms imple-

mented with parallel S-boxes. However, our experiments show that when SGA-CPA is confronted to ciphers with plenty large S-boxes, such as AES-128, premature convergence occurs, i.e., the evolution of candidate key population stops before recovering all the key words.

1.1 Our contributions

In this paper, we take AES [6] for instance to illustrate the premature convergence phenomenon and propose our solution to the problem. Our contributions are:

- **Multiple sieve method.** We analyze the reason of premature convergence for SGA-CPA when conducted to cryptographic algorithms with plenty large S-boxes. Based on the theoretical explanation for that the correct key words obtained by each execution of SGA-CPA are random, a method called multiple sieve (MS-CPA) was proposed taking advantage of the outputs of SGA-CPA to sieve all the correct key words. We apply this method to AES-128 implemented in an 128-bit architecture and compare it with classic CPA and SGA-CPA. Experimental results show that our method overcomes premature convergence and requires less traces than the other two methods.
- **Operators and parameters customization of SGA-CPA.** Operators and parameters used in genetic algorithms have huge influences on the efficiency. Unlike the problems generally processed by genetic algorithms, the correlation coefficient optimization problem discussed in this paper involves S-box operation, which makes SGA-CPA inapplicable inside a key word. In order to cover the candidate key word space as much as possible, we initial the population with a large number of individuals. Operators are customized and adjusted to work on these particular populations and individuals. The corresponding parameters are determined experimentally with regard to complexity and efficiency.

1.2 Organization

The remainder of this paper is organized as follows. Sect. 2 starts with an overview of classic CPA and SGA-CPA along with their imperfections. The theoretical explanation of our method is given afterwards. In Sect. 3, the multiple sieve method is put forward. Sect. 4 describes the operators customized for our method in detail, and the corresponding parameters are determined according to experimental results. Comparisons of MS-CPA, SGA-CPA and classic CPA based on simulated and real experiments are exposed in Sect. 5. We conclude this paper in Sect. 6.

2 Preliminary

2.1 Correlation Power Analysis

Correlation power analysis [3] is a statistical side-channel attack based on the dependency between power consumptions of cryptographic devices and the Ham-

ming distances of intermediate states with regard to some reference ones. In the method, a leakage model $W = aH(D \oplus R) + b$ was presented to define the relationship between the power consumption W and the Hamming distance $H(D \oplus R)$, in which a was a scalar gain depending on the circuit and b enclosed offsets, time dependent components and the noises produced by the device. The correlation coefficient between W and H was calculated using the formula:

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

Attacks were conducted on cryptographic devices by scanning the candidate values of key exhaustively and ranking them by the corresponding correlation coefficients. The correct key was the one that maximized (minimized if a is negative) $\rho_{W,H}$. A partial correlation coefficient $\rho_{W,H_{l/m}}$ was also derived by applying the leakage model to l independent bits amongst an m-bit machine word. The ratio of the two correlation coefficients was

$$\frac{\rho_{W,H_{l/m}}}{\rho_{W,H}} = \sqrt{\frac{l}{m}}.$$

For cryptographic algorithms implemented in devices with a large machine word, the partial correlation coefficient was used to recover the key word by word, known as divide and conquer, in order to avoid searching all the 2^m candidate keys which was usually unpractical. Note that the correlation coefficient was reduced by $\sqrt{\frac{l}{m}}$, and there required more power traces to reduce the noise produced by the remaining $(m - l)$ bits.

2.2 CPA Based on Simple Genetic Algorithm

In 2015, Zhang et al. [23] proposed a method basing on an observation that the correlation coefficients between the intermediate states and the power consumptions were related to the number of correct key words in a parallel implementation: the more correct key words, the higher the correlation coefficient. Therefore, they transformed the problem of searching correct key into the problem of optimizing correlation coefficients of candidate keys, and introduced SGA-CPA which combined CPA with simple genetic algorithm to solve it.

Simple genetic algorithm is a basic approach of genetic algorithms [10], which are a group of probabilistic optimization methods based on the model of natural evolution. In genetic algorithms, there are several basic concepts:

- **Individual and Fitness.** The potential solutions to an optimization problem are called individuals. The objective function is defined as fitness function which is used to evaluate the fitness of an individual.
- **Population and Generation.** A population is a group of individuals, which are initialized randomly. After the initialization, the population is modified by three basic operators in a loop until some termination criterion is reached. Each run of the loop is called a generation.

The three operators and their major functions are:

- **Selection.** An operator intends to improve the average fitness of the population by giving individuals of higher fitness a higher probability to be copied into the next generation.
- **Crossover.** An operator exchanges bit strings between two selected individuals (called parents) with probability p_c, in order to generate new solutions.
- **Mutation.** An operator enables genetic algorithms to generate new bit strings by altering one or a few bits randomly in an individual with a low probability p_m.

In SGA-CPA, the candidate keys were defined as individuals, and the correlation coefficient produced by the intermediate states (encrypted by the candidate key) was the fitness, formally defined as:

$$Fitness := Corr(Trace, Intermediate(Plaintext, Key\_guess)).$$

Algorithm 1 shows the implementation of SGA-CPA, denoted as **SGA_CPA** (Th_gen, N, p_c, p_m).

Algorithm 1 CPA combined with simple genetic algorithm.

Input: threshold of generation Th_gen, size of population N, probability of crossover p_c, probability of mutation p_m.
Output: the optimal key $Best\_key$.

1: $Pop :=$ **InitPopulation**(N);
2: **ComputeFitness**(Pop);
3: $found\_key :=$ **false**;
4: $i := 0$;
5: **while** $!found\_key$ **or** $i < Th\_gen$ **do**
6: $ChildPopulation := \Phi$;
7: **for** $j := 0$ **to** $N/2$ **do**
8: $child_1 :=$**Selection**(Pop);
9: $child_2 :=$**Selection**(Pop);
10: **Crossover**$(child_1, child_2, p_c)$;
11: **Mutation**$(child_1, p_m)$;
12: **Mutation**$(child_2, p_m)$;
13: **Add**$(child_1, child_2, ChildPop)$;
14: **end for**
15: $Pop := ChildPop$;
16: **ComputeFitness**(Pop);
17: $Best\_key :=$**MaxFitness**(Pop);
18: $i := i + 1$;
19: **if Verify**$(Best\_key) =$ **true then**
20: $found\_key :=$**true**;
21: **end if**
22: **end while**
23: **return** $Best\_key$;

2.3 Theoretical Explanation and Existing Problem of SGA-CPA

Assuming D is chosen randomly in $W = aH(D \oplus R) + b$, consider an uniform random variable $\hat{D}$, of which l bits are consistent with D but the other $m - l$ bits are independent with it. Let $H_{l/m}$ represent the Hamming distance between the l bits of $\hat{D}$ and R, and $\hat{H}_{(m-l)/m}$ represent the other $m - l$ bits. Denote $\hat{H} = H(\hat{D} \oplus R)$, then we have $\hat{H} = H_{l/m} + \hat{H}_{(m-l)/m}$. The correlation coefficient between W and $\hat{H}$ is

$$\rho_{W,\hat{H}} = \frac{Cov(W, \hat{H})}{\sigma_W \sigma_{\hat{H}}} = \frac{Cov(W, H_{l/m}) + Cov(W, \hat{H}_{m-l/m})}{\sigma_W \sigma_{\hat{H}}}.$$

Under the assumption that the $m-l$ bits of $\hat{D}$ are independent with D, we have $Cov(W, \hat{H}_{m-l/m}) = 0$. Since the variances of $H_{l/m}$ and $\hat{H}$ are respectively $l/4$ and $m/4$, the correlation coefficient leads to:

$$\rho_{W,\hat{H}} = \frac{Cov(W, H_{l/m})}{\sigma_W \sigma_{\hat{H}}} = \frac{\sigma_{H_{l/m}}}{\sigma_{\hat{H}}} \times \rho_{W,H_{l/m}} = \frac{l}{m} \times \rho_{W,H}. \tag{1}$$

Assume that $m = l \times n$, and $\hat{D}$ has $l \times i$ ($i \in \{0, 1, 2, ..., n\}$) bits consistent with D, then the correlation coefficient between W and $\hat{H}$ is

$$\rho_{W,H_{il/m}} = \frac{l \times i}{m} \times \rho_{W,H} = \frac{i}{n} \times \rho_{W,H}. \tag{2}$$

Taking each l bits as a word, we get that the correlation coefficient increases with the number of correlated words, but is independent with their indexes. In this approach, we take full advantage of this property and propose an efficient method to gain the secret information by sieving and combining the correlated words.

The existing problem of SGA-CPA. The mathematical foundation of genetic algorithms is the schema theorem [10], which states that the number of short, low-order and highly-fit schemas increases exponentially in the subsequent generations. The schemas mentioned in the theorem are defined as building blocks. In the process of genetic algorithms, the individuals with building blocks are identified by selection operators and then combined by crossover operators. New building blocks are generated by mutation operators. After plenty of generations, more and more building blocks accumulate in the individuals according to the schema theorem. Eventually, the best individual is obtained and the genetic algorithm converges to the optimal solution. Obviously, building blocks are critical factors in genetic algorithms.

In the correlation coefficient optimization problem, correct key words (corresponding to S-boxes) are building blocks. The more correct key words an individual has, the higher its fitness will be. However, since most S-boxes are implemented with complicate bool functions, altering any bit of the input usually causes huge changes in the output. Even if a wrong guess has similar bit

string to the correct key word, it still leads to uncorrelated relationship between the power traces and the outputs of the corresponding S-box, i.e., its fitness is similar to the other wrong ones. Hence, there exists no building block smaller than a key word, which make SGA-CPA inapplicable inside a key word. For the cryptographic algorithms that employed small or not so many S-boxes, for example DES and SM4 studied in [23], SGA-CPA works well because it is easy for the initial population to cover all the values of each key words or obtain them by crossover and mutation before the algorithm converge. While, for AES-128 which has 16 8-bit S-boxes, SGA-CAP tends to converge prematurely, because of the insufficient local searching inside each key words.

3 Multiple Sieve Method Based on Simple Genetic Algorithm

In this section, we put forward the multiple sieve method aiming at overcoming premature convergence of simple genetic algorithm when combined with CPA. Our method is focused on the outputs of S-boxes implemented parallel.

According to equation 2, the fitnesses of individuals which have the same number of correct key words are almost equal. A deduction is that the contributions of different correct key words to the fitness of an individual are independent from each other, and are almost the same. As a consequence, the probabilities of correct key words being recovered in an execution of **SGA_CPA()** are nearly equal to each other.

As discussed in Sect. 2.3, **SGA_CPA()** converges before recovering all the key words when applied to cryptographic algorithms with plenty of large S-boxes. Nevertheless, several correct key words are obtained in each execution. By collecting all the optimal individuals obtained by **SGA_CPA()**, we eventually gain all of the correct key words. The problem is how to sieve and combine them to the whole key.

Fig. 1 describes our solution MS-CPA, in which **SGA_CPA()** is executed in a loop, and the correct key words contained in the optimal individuals are sieved and combined until the whole key is recovered. In each loop, **SGA_CPA()** is executed on a new randomly initialized population. A container, denoted as *combine_key*, is initialized with the optimal individual obtained in the first loop. In the subsequent loops, every key word in *combine_key* is replaced by the optimal individual of that loop one by one, and the fitness of *combine_key* is recalculated. If the fitness is larger than the original one, *combine_key* is updated with the new key word. As a result, new correct key words are sieved and maintained in *combine_key*. The pseudo code of MS-CPA is shown in algorithm 2, denoted as **MS_CPA** $(Th\_pop, Th\_gen, N, p_t, p_c, p_m)$.

Assume the average number of correct key words obtained by **SGA_CPA()** is $n_{correct}$ and a key contains n_{word} words. Then, the probability of a key word being recovered is $n_{correct}/n_{word}$ in one loop, and the probability of a key word being unrecovered after n_{pop} loops is $(1-n_{correct}/n_{word})^{n_{pop}}$. Thus, the relationship between the success rate $P_{success}$ of MS-CPA and the minimum number of

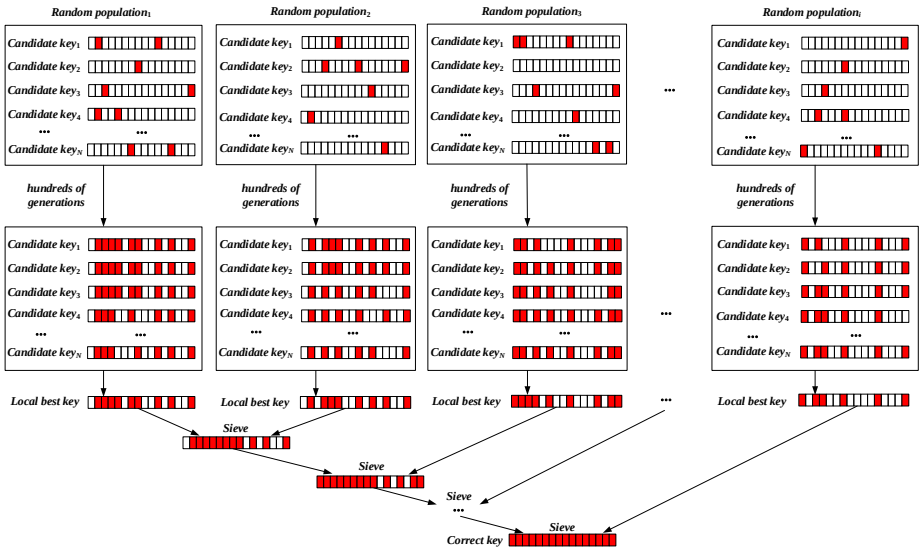


Fig. 1. Multiple sieve method.

Algorithm 2 Multiple sieve method.

Input: threshold number of populations (loops) Th_pop, threshold number of generations Th_gen, the size of a population N, the crossover probability p_c, the mutation probability p_m.

Output: the optimal key $Combine\_key$.

```
found_key := false;
i := 0;
while !found_key or i < Th_pop do
    Candidate_key := SGA_CPA(Th_gen, N, p_c, p_m);
    if i = 0 then
        Combine_key := Candidate_key;
    else
        Temp_key := Combine_key;
        for j := 1 to n_word do
            Temp_key[j] := Candidate_key[j];
            if Fitness(Temp_key) <= Fitness(Combine_key) then
                Temp_key[j] := Combine_key[j];
            else
                Combine_key[j] := Temp_key[j];
            end if
        end for
    end if
    if Verify(Combine_key)= true then
        found_key := true;
    end if
end while
Return Combine_key;
```

loops is

$$P_{success} \approx [1 - (1 - \frac{n_{correct}}{n_{word}})^{n_{pop}}]^n. \quad (3)$$

In order to improve the success rate, larger $n_{correct}$ and n_{pop} should be chosen with regard to computation complexity.

4 Simple Genetic Algorithm Customized for MS-CPA

Operators and parameters are critical factors influencing the efficiency of genetic algorithms. In this section, we customize the simple genetic algorithm used in our method basing on theoretical analysis and experiments.

4.1 Operators Customization

There exist various schemes of the three operators in genetic algorithms. In our method, the simple genetic algorithm is required to converge quickly and recover key words as many as possible.

As discussed in Sect. 2.3, SGA-CAP is inapplicable inside a key word. Thus, we intend to perform the simple genetic algorithm on a large population, which is able to cover the value space of each keyword as much as possible. Several classic schemes, namely proportional selection, roulette wheel selection, truncation selection and tournament selection are tested. With the number of individuals growing, proportional selection and wheel selection are close to random selection. The searching abilities of tournament selection and truncation selection are similar, but tournament selection converge more slowly. Eventually, truncation selection is employed in our method due to its outstanding performance in handling large populations. It works with a parameter p_t. Only the best $N \times p_t$ individuals can be selected, and have the same selection probability. Algorithm 3 shows the outline of this scheme, denoted as **TruncSelection**(Pop, N, p_t).

Algorithm 3 Truncation selection scheme for SGA-CPA.

Input: a sorted population Pop, size of the population N, probability of truncation p_t.
Output: the selected individual C.
1: $index :=$ **Random**$(1, N \times p_t)$;
2: $C := Pop[index]$;
3: **Return** C;

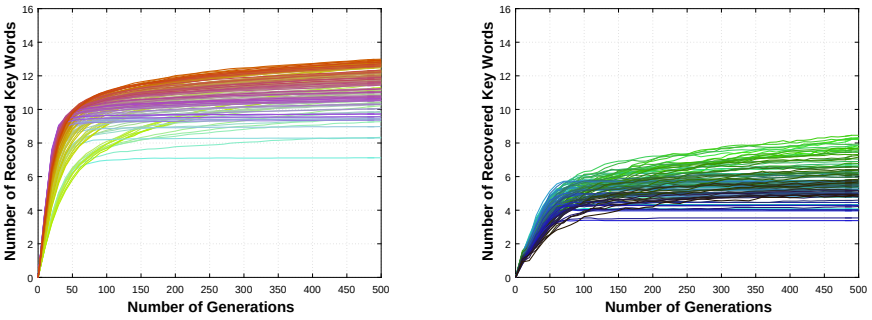
For the crossover operation, we consider three basic schemes generally used in genetic algorithms, namely single point crossover, multiple points crossover and uniform crossover. However, all the three schemes have a drawback which is breaking building blocks easily. In our problem, it is difficult to regenerate a building block, because SGA-CAP is inapplicable inside the key word.

Hence, we customize a scheme for our method which maintains the building blocks and combines them in new individuals by exchanging key words. Algorithm 4 shows the pseudo code of the word-wise crossover, denoted as **WordwiseCrossover**(C_1, C_2, p_c).

Algorithm 4 Crossover scheme for SGA-CPA.

Input: two individuals about to be recombine C_1, C_2, probability of crossover p_c.
Output: two updated individuals C_1, C_2.

```
1: for i := 1 to n_word do
2:     p := Random(0, 1);
3:     if p < p_c then
4:         temp := C_1[i]; C_1[i] := C_2[i]; C_2[i] := temp;
5:     end if
6: end for
7: Return C_1, C_2;
```



(a) (b)

Fig. 2. The number of key words recovered by SGA_CPA with different crossover schemes (a) our word-wise crossover; (b) classic single point crossover.

We compare the single point crossover with our word-wise crossover experimentally. For single point crossover, we test $p_c \in [0.1, 0.9]$ and $p_m \in [0.07, 0.14]$ ranging by 0.1 and 0.005, respectively. For our scheme, we test $p_c \in [0.1, 0.5]$ ranging by 0.05 and $p_m \in [0.07, 0.14]$ ranging by 0.005, considering that the recombination results of C_1 and C_2 are the same for p_c and $1-p_c$. Fig. 2 shows the comparison between single point crossover and word-wise crossover. The experimental results show that the number of key words recovered by **SGA_CPA()** equipped with **WordwiseCrossover()** is much more than that with single point crossover. Similar results can be obtained with the other two schemes, which are omitted here.

The mutation operation usually alters one or a few bits with a very small probability p_t. In our problem, it is mainly used to generate new building blocks. Therefore, we customize a scheme that alters one bit randomly within the key word. The pseudo code of word-wise mutation is given in Algorithm 5, denoted as **WordwiseMutation**(C, p_m).

Algorithm 5 Mutation scheme for SGA-CPA.

Input: the individual about to be mutated C, probability of mutation p_m.
Output: the updated individual C.

```
1: for i := 1 to n_word do
2:    p := Random(0, 1);
3:    if p < p_m then
4:       index := random(1, word_size)
5:       AlterBit(C[i], index);
6:    end if
7: end for
8: Return C;
```

We replace **Selection()**, **Crossover()** and **Mutation()** in Algorithm 1 with the three customized scheme above, and obtain the **SGA_CPA()** employed in our MS-CPA.

4.2 Parameters Selection

The parameters required to be determined for the three operators are the size of population N, the truncation probability p_t, the crossover probability p_c and the mutation probability p_c. Discussing the four parameters simultaneously is complicated, so we classify them into two sets: the selection related ones (N, p_t) and the generation related ones (p_c, p_m).

p_c and p_m are fixed to appropriate values when (N, p_t) are studied. We test $N \in [50, 2000]$ rangeing by 50 and $p_t \in [0.1, 0.8]$ ranging by 0.1. For each couple values of (N, p_t), **SGA_CPA()** is executed for 400 times on different populations and the number of recovered key words $n_{correct}$ are collected and averaged for every 10 generations. The experimental results are shown in Fig. 3, from which we know that **SGA_CPA()** converge after the 150th generation. Therefore, we extract the data at the 150th generation, and exhibit the relationship between $n_{correct}$ and (N, p_t) in Fig. 4. For each value of $N \in [450, 1000]$, $n_{correct}$ reaches the highest value at $p_t = 0.4$, which is highlighted by a red line in the figure. In order to discuss the influence of N on $n_{correct}$, we fix $p_t = 0.4$, and exhibit their relationship in Fig. 5. From this figure, we know that $n_{correct}$ increases along with N, but slowly after $N = 1000$. Considering the computation cost, we chose (N, p_t)=$(500, 0.4)$ in our method. In Sect. 5, $N = 500$ and $N = 1000$ are set for SGA-CPA receptively to be compared with MS-CPA.

For (p_c, p_m), the tested values are mentioned in Sect. 4.2, which are $p_c \in [0.1, 0.5]$ ranging by 0.05 and $p_m \in [0.07, 0.14]$ ranging by 0.005. Similar to

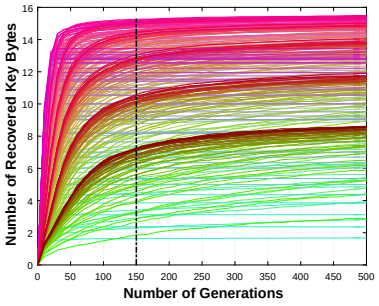


Fig. 3. The relationship between $n_{correct}$ and the generations for different values of (N, p_t).

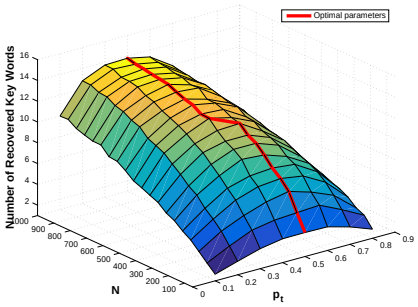


Fig. 4. The relationship between $n_{correct}$ and (N, p_t) at the 150th generation.

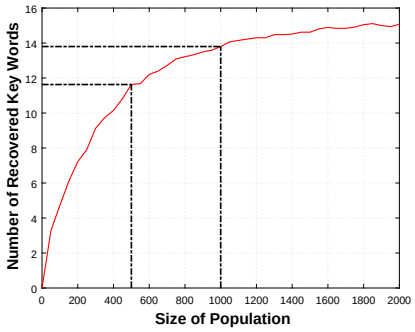


Fig. 5. The relationship between $n_{correct}$ and N with $p_t = 0.4$ at the 150th generation.

(N, p_t), experiments are repeated 400 times for each couple values, and $n_{correct}$ in every 10 generations are collected and averaged. Fig. 6 shows the relationship between $n_{correct}$ and (p_c, p_m), according to which the optimal choice for (p_c, p_m) are $(0.5, 0.12)$ and the corresponding $n_{correct}$ is 11.67.

The minimum number of populations (loops) n_{pop} can be estimated by equation 3 with a certain success rate $P_{success}$. While, due to the randomness of simple genetic algorithm, the number of correct key words in optimal individuals obtained by SGA-CPA is unstable, so the threshold of populations Th_pop is worth to be discussed. We test $Th\_pop \in [5, 100]$ ranging by 5. **MS_CPA()** is executed 400 times for each value. The numbers of populations are collected and averaged for each execution. The success rates are calculated for each value of Th_pop. Fig. 7 shows the relationship among them. At $Th\_pop = 35$, we have $P_{success} = 90\%$ and $n_{pop} = 8.42$. According to equation 3, n_{pop} is 3.89 when $n_{correct} = 11.67$ and $P_{success} = 90\%$. The reason of the deviation is that the contributions of different key words to the fitness of an individual are slightly

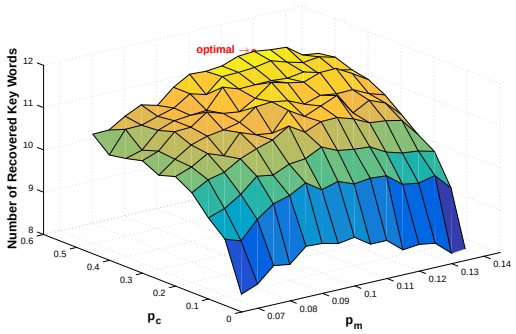


Fig. 6. The relationship between $n_{correct}$ and (p_c, p_m) at the 150th generation.

different from each other in a certain experiment, which leads to some key words being recovered more hardly.

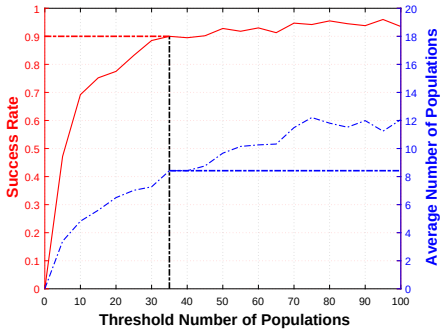


Fig. 7. The relationship among P_{sucess}, Th_pop and n_{pop}.

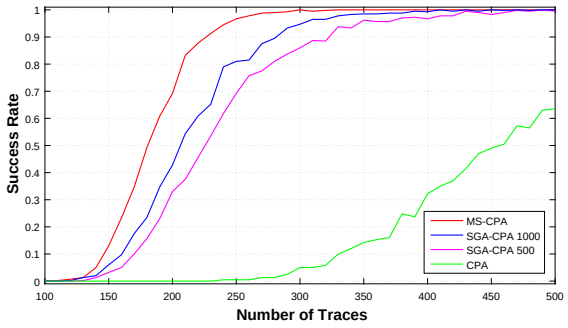
5 Experiments and Comparison

In this section, we take AES-128 implemented in a 128-bit architecture for instance to verify the efficiency of our method. AES-128 is a block cipher that employs a 8-bit S-box to handle the 16 bytes in a block. The plaintexts are masked by a 128-bit white key before the first round. At the end of each round, a 128-bit subkey is xored with the intermediate state. Our experiments are focused on the intermediate states after (before) the S-box operation of the first (last) round.

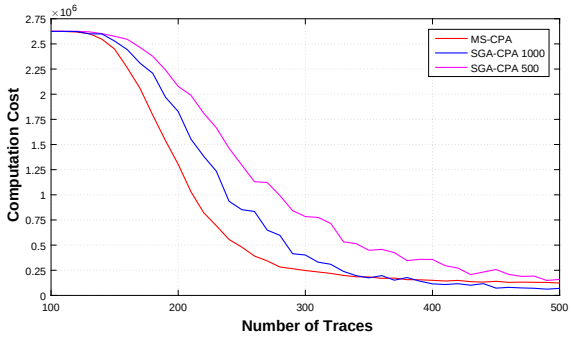
5.1 Simulation experiments

In the simulation experiments, the power consumptions of the S-box operations are simulated by adding two kinds of noises whose standard deviations are $\sigma = 3.0$ and $\sigma = 5.0$, repectively.

We perform MS-CPA, SGA-CPA and classic CPA on the same group of traces. The size of group is ranging from 10 to 1000. For MS-CPA, the parameters are $(N, p_t, p_c, p_t, Th\_gen, Th\_pop) = (500, 0.4, 0.5, 0.12, 150, 35)$. For SGA-CPA, two kinds of parameters are considered here: $N = 500$, $Th\_gen = 150 \times 35 = 5250$ and $N = 1000$, $Th\_gen = 150 \times 35/2 = 2625$, denoted as SGA-CPA 500 and SGA-CPA 1000, respectively. Their success rates and computation costs are displayed in Fig. 8 and Fig. 9. The computation cost is estimated as the average number of calculations for correlation coefficients. The sorting costs in truncation selection are ignored here. The numbers of traces required to achieve 90% success rate are summarized in Tab. 1.

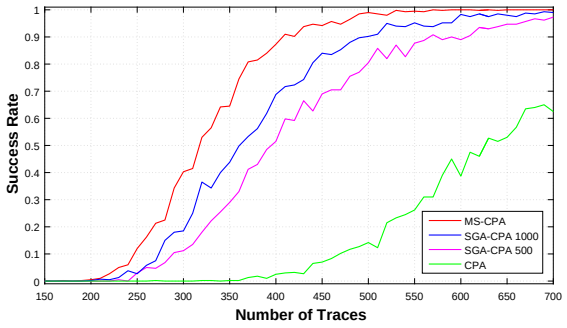


(a) Success Rate.

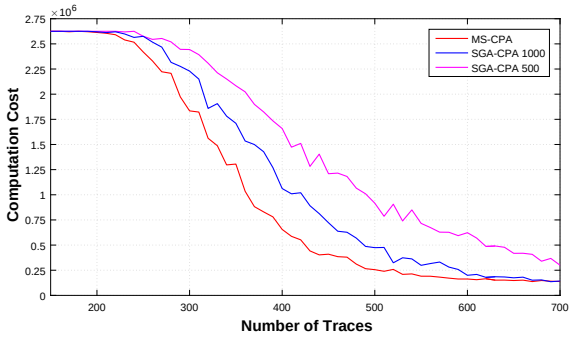


(b) Computation Cost.

Fig. 8. Comparison of MS-CPA, SGA-CPA with $N = 500, 1000$ and classic CPA for $\sigma = 3.0$.



(a) Success Rate.



(b) Computation Cost.

Fig. 9. Comparison of MS-CPA, SGA-CPA with $N = 500, 1000$ and the classic CPA for $\sigma = 5.0$.

The experimental results show that the success rate of our method is much more than the other two methods, and the computation costs of MS-CPA are less than SGA-CPA. For $\sigma = 3.0$, it is required 225 traces for MS-CPA to achieve 90% success rate, the corresponding computation cost is about 0.75×10^6 calculations. With the same number of traces, the success rates of SGA-CAP 1000 and SGA-CPA 500 are about 63% and 49.5%, and the corresponding computation costs are about 1.31×10^6 and 1.74×10^6 calculations, respectively. For $\sigma = 5.0$, it is required 405 traces for MS-CPA to achieve 90% success rate, the corresponding computation cost is about 0.60×10^6 calculations. With the same number of traces, the success rates of SGA-CAP 1000 and SGA-CPA 500 are about 70.1% and 55.7%, and the corresponding computation costs are about 1.04×10^6 and 1.57×10^6 calculations, respectively. The success rate of classic CPA is nearly zero in both situations. Although the success rate of SGA-CPA can be improved by extending the size of populations, it is still less than that of MS-CPA.

Table 1. comparison of MS-CPA with SGA-CPA and classic CPA

Method	$\sigma = 3.0$		$\sigma = 5.0$	
	#Traces	Reduction	#Traces	Reduction
Our MS-CPA	225	-	405	-
SGA-CPA 1000	280	19.64%	490	17.37%
SGA-CPA 500	325	30.77%	565	28.32%
CPA	620	63.70%	875	53.71%

5.2 Experiments on FPGA

For the real experiments, we encrypt random plaintexts with a fixed key using AES-128 provided officially by SAKURA-G, and acquire 2000 power traces. Since the registers are set before S-box operations, our experiments are focused on the intermediate states before S-box operations of the last round and the ciphertexts.

Firstly, we calculate the correlation coefficients between the power traces and the intermediate states which are obtained by decrypting the ciphertexts with candidate keys that have i ($i \in \{0, 1, 2, ..., 16\}$) correct key words. Candidate keys are chosen randomly, and the number of traces used to calculate the correlation coefficients is ranging from 10 to 1000. The relationship between the correlation coefficients and the number of traces is shown in Fig. 10. The blue lines indicate the wrong keys, and the red one indicates the correct key. The saturation of the blue lines helps to distinguish the candidate keys with different number of correct key words: the less the correct key words, the lower the saturation. Apparently, the blue lines with higher saturation have larger correlation coefficients, which implies that the more correct key words a candidate key has, the larger correlation coefficient it leads to, and it is more and more obvious when the number of traces increases. Besides, the red line corresponding to correct key out stands from others when the number of traces is above 280, which indicates that the minimum threshold for attacks based on correlation coefficients of the whole key is 280.

Secondly, classic CPA is conducted on the same group of power traces. Similarly, the size of the group is ranging from 10 to 1000. We calculate the correlation coefficients corresponding to all the values of each key word. Fig. 11 shows the experimental results of the one that requires the most number of traces to identify the correct value. The blue lines indicate the wrong guesses and the red one indicates the correct guess. Obviously, the minimum number of traces required for CPA to recover all the key words is depending on the one shown in Fig. 11, which is 500. Therefore, the minimum threshold for attacks based on the correlation coefficients of single key word is 500, which is 44% more than the one of the whole key.

Finally, we perform MS-CPA and SGA-CPA on these traces. The fitnesses of the best individuals obtained by every generation as well as the combined ones are collected. Fig. 12 and Fig. 13 show the relationship between the fitnesses and the number of traces. The red line indicates the fitness of the correct key. The blue one stands for the fitness of the combined individual in MS-CPA,

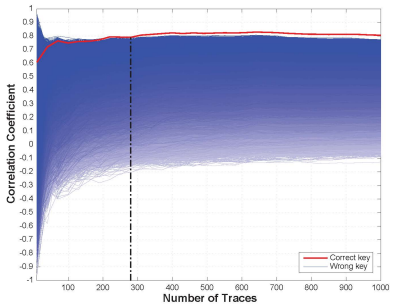


Fig. 10. The relationship between correlation coefficients and the number of traces for different candidate keys.

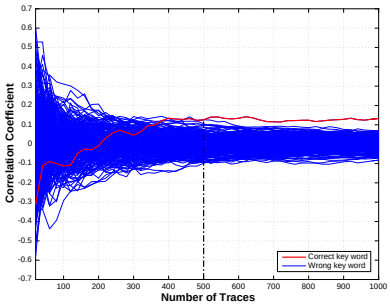


Fig. 11. The relationship between correlation coefficients and the number of traces for a single key word in classic CPA.

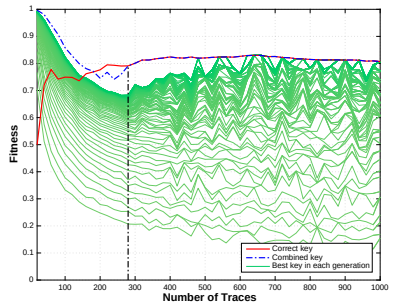


Fig. 12. The relationship between the number of traces and fitnesses of the best individuals in each generations as well as the combined ones in MS-CPA.

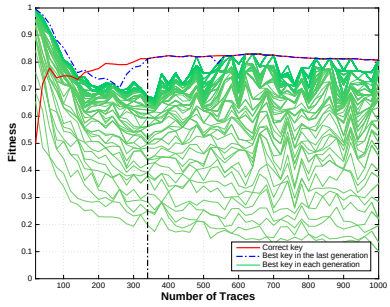


Fig. 13. The relationship between the number of traces and fitnesses of the best individuals in the first 150 and the last generations in SGA-CPA.

and the best individual obtained by the last generation (5250th) in SGA-CPA, respectively. The green ones indicate the fitnesses obtained by the first 150 generations in both methods. For MS-CPA, the blue line coincide with the red one when the number of traces is above 280, which implies that MS-CPA nearly reaches the minimum threshold of traces for attacks against the whole key. For SGA-CAP, the blue line coincide with the red one when the number of traces is above 340, which is 17.64% more than that of MS-CPA. Note that, at the point $Num\_traces = 540$, the fitness of the best key obtained by SGA-CPA is lower than the correct one. In fact, similar phenomenons occur almost in each experiment of SGA-CPA at different points. The major reason is that SGA-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. While, our method MS-CPA is not suffering from this problem. This experimental results show that MS-CPA is robust to SGA-CPA.

6 Conclusion

In this approach, we discuss the imperfections of using simple genetic algorithm to solve the correlation coefficient optimization problem, and put forward the multiple sieve method which not only overcomes the premature convergence of simple genetic algorithm when combined with CPA, but also reduce the number of traces required in the power analysis on cryptographic algorithms implemented with parallel S-boxes. Experimental results verify the efficiency and robustness of our method, when compared to SGA-CPA and classic CPA. The number of traces required in MS-CPA is nearly equal to the minimum threshold of traces for power attacks against the whole key. Our approach extends the application of genetic algorithms in power analysis.

References

1. Bartkewitz, T.: Leakage prototype learning for profiled differential side-channel cryptanalysis. IEEE Transactions on Computers **65**(6), 1761–1774 (2016)
2. Bartkewitz, T., Lemke-Rust, K.: Efficient template attacks based on probabilistic multi-class support vector machines. In: International Conference on Smart Card Research and Advanced Applications. pp. 263–276. Springer (2012)
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 16–29. Springer (2004)
4. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: International Conference on Cryptographic Hardware and Embedded Systems. pp. 45–68. Springer (2017)
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 13–28. Springer (2002)
6. Daemen, J., Rijmen, V.: The design of Rijndael: AES-The Advanced Encryption Standard. Springer-Verlag, Berlin, Heidelberg (2002)
7. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 426–442. Springer (2008)
8. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Lightweight ciphers and their side-channel resilience. IEEE Transactions on Computers (2017)
9. Heuser, A., Zohner, M.: Intelligent machine homicide. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 249–264. Springer (2012)
10. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press (1975)
11. Hospodar, G., Mulder, E., Gierlichs, B., Verbauwhede, I., Vandewalle, J.: Least squares support vector machines for side-channel analysis. Center for Advanced Security Research Darmstadt pp. 99–104 (2011)

12. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Annual International Cryptology Conference. pp. 388–397. Springer (1999)
13. Le, T.H., Clédière, J., Canovas, C., Robisson, B., Servière, C., Lacoume, J.L.: A proposition for correlation power analysis enhancement. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 174–186. Springer (2006)
14. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack: an approach based on machine learning. Center for Advanced Security Research Darmstadt pp. 29–41 (2011)
15. Lerman, L., Medeiros, S.F., Veshchikov, N., Meuter, C., Bontempi, G., Markowitch, O.: Semi-supervised template attack. In: International Workshop on Constructive Side-Channel Analysis and Secure Design. pp. 184–199. Springer (2013)
16. Martinasek, Z., Zeman, V.: Innovative method of the power analysis. Radioengineering **22**(2), 586–594 (2013)
17. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Investigations of power analysis attacks on smartcards. Smartcard **99**, 151–161 (1999)
18. Picek, S., Heuser, A., Jovic, A., Legay, A.: Climbing down the hierarchy: hierarchical classification for machine learning side-channel attacks. In: International Conference on Cryptology in Africa. pp. 61–78. Springer (2017)
19. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.: Side-channel analysis and machine learning: A practical perspective. In: Neural Networks (IJCNN), 2017 International Joint Conference on. pp. 4095–4102. IEEE (2017)
20. Schramm, K., Wollinger, T., Paar, C.: A new class of collision attacks and its application to des. In: International Workshop on Fast Software Encryption. pp. 206–222. Springer (2003)
21. Shan, W., Zhang, S., He, Y.: Machine learning based side-channel-attack countermeasure with hamming-distance redistribution and its application on advanced encryption standard. Electronics Letters **53**(14), 926–928 (2017)
22. Tsague, H.D., Twala, B.: Reverse engineering smart card malware using side channel analysis with machine learning techniques. In: Big Data (Big Data), 2016 IEEE International Conference on. pp. 3713–3721. IEEE (2016)
23. Zhang, Z., Wu, L., Wang, A., Mu, Z., Zhang, X.: A novel bit scalable leakage model based on genetic algorithm. Security and Communication Networks **8**(18), 3896–3905 (2015)