Gini-Impurity Index Analysis

Ye Yuan, Liji Wu, *Member, IEEE*, and Xiangmin Zhang

***Abstract*— In the past few decades, DPA-based side-channel attack strategies, such as DPA and CPA, have shown strong ability to analyze the security of the cryptographic implementations. However, the unpredictability of the leakage model and the correspondence between leakage behavior of the target device and the hypothetical leakage value make it less-effective without prior knowledge. Therefore, in this paper, we present a novel generic side-channel analysis method called Gini-impurity Index Analysis (GIA), utilizing Gini-impurity Index as the distinguisher, which can perform well even without any leakage model and is not sensitive to the existing methods' restrictions about the leakage behavior. Firstly, we introduce the basic idea of GIA. According to the proposed GIA attack strategy, the Gini-impurity index for each key hypothesis should be calculated, determined by the clustered power consumption and the classified subsets based on the key dependent target function. Secondly, we verify the feasibility and evaluate the efficiency of GIA with different target functions by the practical experimental results against AES-128 implemented on an AT89S52 microcontroller. We present one possible multivariate extension of GIA and find the advantage of GIA on leakage information utilization. Thirdly, we present the results of comparisons. On the one hand, we compare GIA with three widely-used distinguishers under simulated traces in various leakage scenarios and practical traces with Hamming-weight-related leakage. Results confirm that GIA can always perform well with different leakage models in most situations. On the other hand, we analyze the relationship between GIA and Mutual Information Analysis (MIA). Theoretical and experimental results confirm that these two methods can obtain similar attack results. However, the guessing entropy of GIA is lower than MIA by up to 21%, and the averaged computational time overhead of GIA is lower than MIA by up to 13.3%, indicating that GIA is more efficient than MIA. Compared to traditional MIA, GIA is easier to operate and more flexible with noise. Therefore, GIA is an efficient and useful alternative to these existed strategies.**

***Index Terms*— Gini-impurity index, side-channel attack, differential power analysis, performance evaluation.**

I. Introduction

THESE years, hardware devices based on cryptography, such as smart cards, mobile phones, Internet of things and the like, have been playing important roles in every aspects of human life, which require information security when operating in a hostile environment. Although it has been proved that nowadays widely-used cryptographic algorithms are mathematically safe, some physical information acquired from side-channel of the operating device shall be a serious threat to security. Since first being put forward in 1996 by Kocher *et al.* [1], side-channel attack (SCA) has been a powerful attack technique and drawn much attention [30], [31]. Under this attack scenario, an attacker can reveal the secret key by exploiting the relations between intermediate results of the executed algorithm and obtained side-channel information, such as power consumption [2]–[4], electromagnetic emanation (EM) [5]–[7], time [1], [8] and so on. Power analysis and EM analysis are similar in essence, and have become the most mature techniques, for the advantages of low cost, high efficiency and wide application range. Up to now, there have been many widely-used power consumption analysis methods such as simple power analysis (SPA) [9], differential power analysis (DPA) [2], correlation power analysis (CPA) [10], [11], template attack (TA) [12], and generic attack methods [29], i.e. Mutual Information Analysis (MIA) [23], Kolmogorov-Smirnov Analysis (KSA) [24] and so on.

A. Related Work

The principal part of SPA is to distinguish the different operations of the processing cryptographic algorithm based on the corresponding different features of power traces [9]. In other words, the attacker tries to recover the key value from a single trace more or less, seeming to be a challenging task. And the attacker needs to have more prior knowledge about the target device, making itself a limit.

TA has been known as a powerful attack technique proposed by Chari *et al.* in 2002 [12], and later template-based DPA was presented with more powerful attack ability [15], [16]. These template-based methods reveal the right key guess by the distinguisher of maximum-likelihood decision rule. However, the necessary precondition for TA is that the attacker shall have access to a device which is the same as or similar to the target one on the features of power consumption and whose key value is known to the attacker. This restricts the application of TA seriously.

DPA can be applied to more attack environment than SPA, for that it is not necessary to have the detailed knowledge about the target device [9]. DPA is based on the fact that the difference of mean value of two power-trace sets which are grouped by the right key guess has the highest peak at the corresponding point [2]. The distinguisher for DPA is the difference of means based on a single bit model. Later, some enhanced versions of DPA based on multiple-bits model have been proposed [13] to overcome the low efficiency caused by the single bit model, but it is still inefficient and limited for using only part of the power consumption.

Manuscript received November 7, 2020; revised March 3, 2021 and April 21, 2021; accepted April 22, 2021. Date of publication April 30, 2021; date of current version May 21, 2021. This work was supported by the National Science and Technology Major Project of China under Grant 2017ZX01030301. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Georg Sigl. *(Corresponding author: Liji Wu.)*

The authors are with the Beijing National Research Center for Information Science and Technology, Institute of Microelectronics, Tsinghua University, Beijing 100084, China (e-mail: yuan-y15@mails.tsinghua.edu.cn; lijiwu@mail.tsinghua.edu.cn; zhxm@mail.tsinghua.edu.cn).

Digital Object Identifier 10.1109/TIFS.2021.3076932

In CHES 2004, Brier *et al.* proposed CPA, where Pearson correlation coefficient was applied as a distinguisher [10]. The basic idea of CPA is to reveal the key value by exploiting the correlation coefficient between the hypothetical leakage of processed intermediate data and the practical power consumption. Some high order CPA techniques have been presented for fighting against high-order masking and other countermeasures [14]. CPA is more efficient in practice for utilizing all available information. However, above-stated techniques are based on the assumption that the relationship between the hypothetical leakage and the measured power consumption is linear, and they require a better leakage model to describe the power consumption of the device. Some generic side-channel attack strategies (e.g., MIA, KSA), were proposed one after another to overcome the restrictive assumption. Among them, MIA, based on the information theory of mutual information, is the most widely-used method with better performance. However, the calculation of mutual information needs the logarithm operation, leading to higher computational time overhead.

Gini coefficient, which combines Pearson and Spearman correlation coefficient, was first introduced to side-channel analysis to improve the attack efficiency in [33]. This state-of-the-art idea can inspire us for a more flexible distinguisher.

B. Contribution

In this paper, we propose a new side-channel attack strategy whose distinguisher is based on Gini-impurity index, called Gini-impurity Index Analysis (GIA). Our contributions can be summarized as follows:

- Firstly, we introduce the basic idea of GIA. According to the proposed GIA attack strategy, the Gini-impurity index for each key hypothesis should be calculated, determined by the clustered power consumption and the classified subsets based on the key dependent target function. GIA combines the traditional side-channel analysis framework with the k-mean algorithm, which is an unsupervised cluster method. As the unsupervised cluster method is sensitive to more features, GIA can make full use of the measurement information. In addition, GIA is not sensitive to the restrictions about the leakage behavior. GIA can work effectively under the generic assumption, and it may be able to recover the key value when other methods fail due to the lack of a suitable leakage model. Our work is related to MIA 23], and these approaches can be well-done without a suitable leakage model but benefit from a good one.
- Secondly, we verify the feasibility and evaluate the efficiency of GIA with different target functions by the realistic experimental results against AES-128 implemented on an AT89S52 microcontroller. Similar to MIA, our method shall fail when the target function is injective and we apply the 'bit drop' trick [21], [22] to evaluate GIA. The results confirm the feasibility of GIA and suggest that the efficiency of GIA is related to the target function and the number of traces. By the multivariate nature of k-means, GIA can be easily extended to utilize more trace points for performance improvement. Moreover, according to the better performance of GIA without averaging operation, we find the advantage of GIA on leakage information utilization.
- Thirdly, we analyze the relationship between GIA and MIA theoretically and experimentally. According to the theoretical analysis, both methods can obtain similar tendencies and results based on the first order Taylor series expansion, confirmed by the experiment results. However, GIA is more efficient than MIA in terms of guessing entropy and time overhead, and GIA combined with k-means shows advantages over traditional MIA in terms of flexibility.
- Finally, we compare the proposed method with some widely-used distinguishers under both simulation-based and practical traces with various leakage scenarios in both noise and noise-free settings. Results from both simulated and practical experiments demonstrate that GIA can always perform well in most situations where other methods fail to recover the key value.

Thus, the universality of GIA indicates that GIA can be an effective alternative to the current commonly used strategies, and has a great application prospect.

C. Organization

The rest part of this paper is organized as follows. In Section II, we introduce some prior knowledge about this work. In Section III, we describe the construction and the scenario of GIA in detail and make some theoretical statements. In Section IV, the experimental verifications of GIA are shown, including the feasibility and efficiency evaluation for different dropping bits models, the multivariate extension of GIA, and the impact of the averaging operation on GIA. In Section V, we present the comparisons results. On the one hand, we demonstrate experimental results of four attack methods. On the other hand, we analyze the relationship between GIA and MIA. The conclusions are provided in Section VI.

II. Preliminaries

A. Basic Model of Side-Channel Analysis

Fig. 1 presents the basic model of traditional side-channel analysis. As shown in this figure, we consider a cryptographic device, such as a smart card, performing a cryptographic operation based on the certain unknown key value with different given plaintexts. From normal channel, different plaintexts or corresponding ciphertexts are available for the attacker. From side-channel, the adversary is able to obtain measurements of power consumption corresponding to the processing data. Information from normal channel and side-channel shall be applied together to the traditional power analysis model.

Traditional power analysis exploits the dependency between the power assumption and the hypothetical leakage value to reveal the key value in a *divide-and-conquer* scheme. As is shown in Fig. 1, for the target function (e.g. the output of the S-box in AES), the adversary should exhaustively test all partial key hypotheses to find the most likely one. In order to map the hypothetical intermediate value of target operation to the hypothetical leakage value, a proper leakage model needs

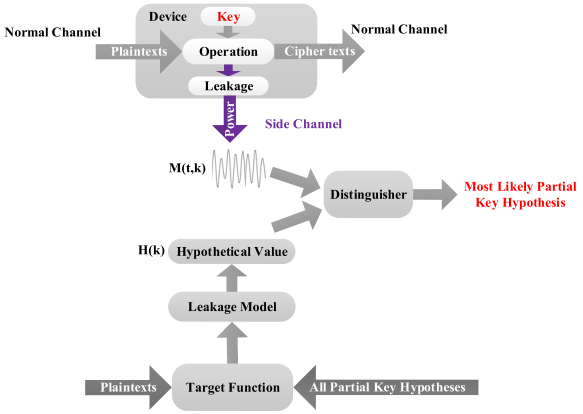


Fig. 1. Basic model of traditional side-channel analysis.

to be proposed. Usually, it is assumed that this leakage model should be surjective. Hamming-distance and Hamming-weight are two widely used power consumption leakage models, which are based on the bit-flips, main contribution to the overall leakage. However, some strategies directly employ the whole or part of the target function results as the hypothetical value, such as the value model or drop-bit models, which is equivalent to no leakage model.

The distinguisher refers to the statistical tool, applied to the measurements and hypothetical value for distinguishing the correct and wrong key hypothesis. Distinguisher can help the attacker to find not only the most likely key hypothesis but also the exact operating point of the target function.

Difference of means is the distinguisher for DPA, which is defined as:

$$\{k^*, t^*\} = \arg\max_{k,t}(\overline{M(t,k)|_{H(k)=1}} - \overline{M(t,k)|_{H(k)=0}}), \quad (1)$$

where M refers to the set of the power consumption measurements, t and k to the operating points and key hypothesis, H to the hypothetical leakage value, and $\bar{M}$ to the mean of M.

CPA utilizes Pearson's correlation coefficient as the distinguisher, which can test the linear dependency between two random variables, defined as:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}, \quad (2)$$

where X and Y are two random variables. The value ranges from -1 to 1, thus the attacker usually uses the absolute value of the correlation coefficient. Based on the Pearson's correlation coefficient, the distinguisher of CPA is as follows:

$$\{k^*, t^*\} = \arg\max_{k,t} |\rho(M(t,k), H(k))|. \quad (3)$$

However, CPA can only test the linear relationship and needs a suitable leakage model.

MIA employs Mutual Information as the distinguisher. Mutual Information can measure statistical relationship (not restricted to the linearity) between two random variables [25], defined as:

$$I(X; Y) = H(X) - H(X|Y), \quad (4)$$

where X and Y are two random variables with n possible values $x_i|_{i=1}^n$ and $y_i|_{i=1}^n$. $H(X)$ is referred to the information entropy [20], and the formula is given by:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i). \quad (5)$$

Larger Mutual Information means the closer relationship between two variables. Based on Mutual Information, the distinguisher of MIA is as follows:

$$\{k^*, t^*\} = \arg\max_{k,t} I(M(t,k); H(k)). \quad (6)$$

B. Gini Value and Gini-Impurity Index

Before introducing the Gini-impurity index, we first give the definition of Gini value. Intuitively speaking, Gini value reflects the probability that two samples randomly chosen from one dataset have different original type labels. Thus, the purity of a dataset denoted as D can be measured by Gini value as:

$$\begin{aligned} \text{Gini}(D) &= \sum_{i=1}^{n}\sum_{i'\neq i} p_i p_{i'} \\ &= 1 - \sum_{i=1}^{n} p_i^2, \end{aligned} \quad (7)$$

where n equals to the number of the types in one dataset D, p_i is denoted as the probability that type i occurs in dataset D. The less Gini value is, the higher the purity of the dataset is. For example, if one dataset D contains 10 samples, among them 3 samples from type A, 4 samples from type B and 3 samples from type C, then the Gini value for dataset D shall be $\text{Gini}(D) = 1 - [(\frac{3}{10})^2 + (\frac{4}{10})^2 + (\frac{3}{10})^2] = 0.66$.

Gini-impurity Index is a widely used concept in Classification and Decision Tree algorithm, as a principle to judge the classification quality [17], [18]. If a dataset D is classified into some subsets based on an attribute (denoted as k), on the basis of Gini value, we can give the definition of Gini-impurity index for this attribute:

$$\text{Gini\_index}(D, k) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} Gini(D^v) \quad (8)$$

where D^v refers to one subset of dataset D classified based on attribute k, V to the total number of the subsets, $|D^v|$ and $|D|$ to the total number of samples in subset D^v and in dataset D respectively. Considering that different subsets contain different number of samples, Gini value of each different subset D^v are endowed with a weight of $\frac{|D^v|}{|D|}$, meaning that the subset with more samples has a greater effect on the result of the final $\text{Gini\_index}(D, k)$. Thus the attribute k corresponding to the least Gini-impurity index is seen as the optimal attribute k^* classifying the dataset most properly, namely:

$$k^* = \arg\min_{k\in K} \text{Gini\_index}(D, k). \quad (9)$$

For a better understanding, we take an example to illustrate this concept. Table I shows a dataset of apple, whose quality depends on two attributes: size and color. In order to figure out which contributes can make a better classification of quality, we should calculate the Gini-impurity index of each attribute.

TABLE I
APPLES DATASET

Number	Size	Color	Quality
1	Big	Red	Good
2	Small	Red	Good
3	Big	Red	Good
4	Small	Green	Bad
5	Big	Green	Bad
6	Small	Red	Good

As for the size attribute, D^1 is referred to the subset of big size, and D^2 to the subset of small size. D^1 contains 3 samples: 1, 3 and 5, among which two samples are of good quality, and one of bad quality; D^2 contains 3 samples: 2, 4 and 6, among which two samples are of good quality, and one sample is of bad quality. Based on (7), the Gini value can be calculated as:

$$\text{Gini}\left(D^1\right) = 1 - ((\frac{2}{3})^2 + (\frac{1}{3})^2),$$
$$\text{Gini}\left(D^2\right) = 1 - ((\frac{2}{3})^2 + (\frac{1}{3})^2).$$

Thus,

$$\text{Gini\_index}\,(D, size) = \frac{3}{6}\text{Gini}\left(D^1\right) + \frac{3}{6}\text{Gini}\left(D^2\right) = \frac{5}{9}.$$

As for the color attribute, it is similar to size attribute. According to Table I, we can obtain: Gini_index $(D, color) = \frac{4}{6}[1 - (1^2 + 0^2)] + \frac{2}{6}[1 - (1^2 + 0^2)] = 0$.

C. K-Means Algorithm

K-means algorithm is a classic clustering algorithm, widely used in the unsupervised learning tasks. K-means algorithm is easy to implement and has a low computational time overhead. It can be utilized to reveal the inherent law and nature of the dataset without knowing much information of the data [28], [32]. Given the number of the clusters and the initial dataset, k-means algorithm is to cluster the dataset $D = \{x_1, x_2, x_3, \cdots, x_m\}$ into k types: $C = \{c_1, c_2, c_3, \cdots, c_k\}$, and minimize the squared error function. The function is defined as:

$$\mathrm{E} = \sum_{i=1}^{k} \sum_{x \in c_i} |x - \mu_i|^2,$$

where $\mu_i = \frac{1}{|c_i|} \sum_{x \in c_i} x$ and $|c_i|$ refers to the total number of data x belonging to type c_i. However, k-means can obtain the optimal results when the shape of the clusters is convex, namely the distribution of the samples within each cluster being normally distributed [36].

GIA applies k-means algorithm to cluster the initial power traces. K-means algorithm is pretty suitable for GIA, because the only parameter k is determined by the target function and any other estimations are unnecessary.

III. GINI-IMPURITY INDEX ATTACK

In this section, we first introduce the general framework of GIA. Then we illustrate the Gini-impurity distinguisher in detail. Finally, we give the theoretical justification of GIA.

A. General Framework of GIA

GIA combines traditional side-channel analysis model with unsupervised cluster method. The main flow of GIA is shown in Algorithm 1. Like most attack methods, the first step of GIA is to run N encryption operations with randomly chosen plaintexts and to obtain N power traces for N encryption runs. Each power trace, denoted as $\{t_{n,o}\}_{o=1}^{T}$ $(1 \le n \le N)$, contains T points which are responding to different operations of the cryptographic algorithm. *Divide-and-conquer* scheme is to ensure a feasible computation complexity, so we only care about parts of the plaintexts and the key hypotheses when computing the intermediate value of the target operation. p^n refers to part of the P^n, for example the first byte of P^n. $D(o)$ is denoted as the dataset for the *o-th* points of all power traces $\{t_{n,o}\}_{n=1}^{N}$.

GIA directly utilizes the value of the target function as the leakage model. The obtained N power traces and the value of the target operation for each key guess are used for Gini-impurity Index Distinguisher, determining the final key value with the minimum value of Gini_index, equivalent to the maximum value of (1- Gini_index). Gini-impurity Index Distinguisher is the core of GIA, and the detail is shown in the following sub-section.

Algorithm 1 Gini-Impurity Index Attack for Part of the Key Value

Result: k^* (one part of the key value K)
1: $\{P^n \mid n = 1, 2, 3, \cdots, N\} \leftarrow$ ***Random_Plaintexts***(N)
2: $\{t_{n,o} | 1 \le o \le T\}_{n=1}^{N} \leftarrow$ ***AcquireTrace***$(\{P^n\}_{n=1}^{N})$
3: $\boldsymbol{k} = \{k^1, k^2, k^3, \cdots, k^M\}$
$\leftarrow$ ***GenerateHypotheticalPartKeyVector***
4: $\boldsymbol{p} = \{p^1, p^2, p^3, \cdots, p^N\}$
$\leftarrow$ ***PartPlaintextVector***
$(\{P^n \mid n = 1, 2, 3, \cdots, N\})$
5: $v_{i,j} = \varphi\left(p^i, k^j\right)$ $(1 \le i \le N; 1 \le j \le M)$
$\leftarrow$ ***TargetOperation***$(\boldsymbol{k}, \boldsymbol{p})$
Gini-impurity Index Distinguisher:
6: **for** $1 \le j \le M$, $1 \le o \le T$
7: GiniIndex$\left(D(o), k^j\right) \leftarrow$
CalculateGI $\left(D(o), \{v_{i,j}\}_{i=1}^{N}, \{p_n\}_{n=1}^{N}\right)$
($D(o)$ is the dataset for the o-th points of all power traces)
8: **end for**
9: **return** $k^* = \arg\max_{k^j, o}$ Gini_index$\left(D(o), k^j\right)$

B. Gini-Impurity Index Distinguisher

There are two main steps of Gini-impurity Index Distinguisher, which are to calculate the Gini-impurity index (GI) for each key guess as an attribute to dataset $D(o)$ and to choose the key candidate minimizing the GI (or maximizing 1-GI) as the final result. Fig. 2 shows the flow of calculating the GI for a key guess k^j.

First, we choose the *o-th* point of each power trace to form the dataset $D(o)$. However, all the datasets $\{D(o)\}_{o=1}^{T}$ should

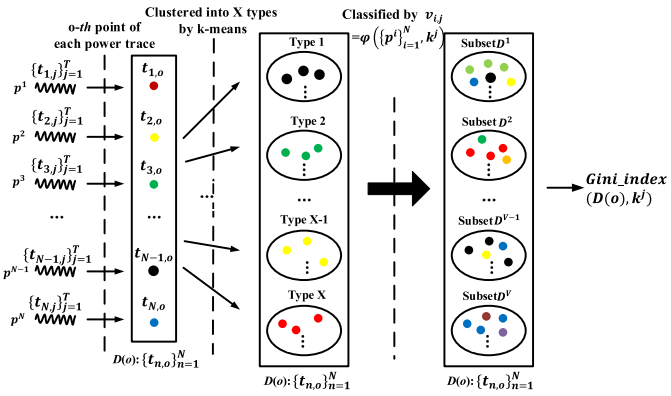


Fig. 2. Flow of calculating the GI for k^j.

be used to calculate the GI. For the chosen point, the relationship between power consumption of the wrong point and the intermediate value of target function is independent.

Then, dataset $D(o)$ is clustered to X types by k-means algorithm, and X equals to the cardinality of the target function value. In Fig. 2, each type is marked with a different color.

Next, the dataset $D(o)$ will be classified in to V subsets based on the intermediate value $\{v_{i,j}\}_{i=1}^{N}$ of target function determined by the key guess k^j, where V refers to the cardinality of the $\{v_{i,j}\}_{i=1}^{N}$. It should be noted that the number of the types should be the same as the number of the subsets. For example, if the target function is the 4 least significant bits of the S-box's output, V equals to 16 and X should also equal to 16.

For each subset $D^i (1 \le i \le V)$, it may contain the elements from type 1 to X. As is shown in Fig. 1, there are some elements with various colors in different subsets. According to (1), we can calculate the Gini value for each subset:

$$\text{Gini}\left(D^i\right) = 1 - \sum_{m=1}^{X} p_m^2, \tag{10}$$

where p_m is denoted as the possibility that the trace point of type m belongs to subset D^i. And p_m can be estimated by:

$$p_m = \frac{|\{t_{n,o}\}_{n=1}^{N} \in \left\{D^i \cap Type\ m\right\}|}{\left|D^i\right|}. \tag{11}$$

After calculating the Gini value for all subsets, GI for a key guess k^j can be calculated as:

$$\text{Gini\_index}\left(D(o), k^j\right) = \sum_{i=1}^{V} \frac{\left|D^i\right|}{|D(o)|} Gini\left(D^i\right). \tag{12}$$

The Gini value reflects the purity of subset D^i. The less the GI is, the higher the quality of classification based on k^j is.

Finally, the GI for all key candidate and power trace points can be obtained, and k^j with the minimum GI (or maximum 1-GI) is chosen as the final result:

$$\begin{aligned} k^*, o^* &= \arg\min_{k^j, o} \text{Gini\_index}\left(D(o), k^j\right) \\ &= \arg\max_{k^j, o} (1 - \text{Gini\_index}\left(D(o), k^j\right) \end{aligned} \tag{13}$$

The target function for the intermediate value has a great effect on the final result. It should be noted that the attacker can't choose injective function as the target function, otherwise injective function will cause that the result for different key guess is just a permutation. For example, the attacker can't directly choose the output value of S-box in AES encryption as the target function, for that different key candidate just produces a permutation of the output value, and thus the values of GI for all key guesses are the same. However, it is reasonable to choose any bit value of the output as the target function. In [21], [22], 'bit drop' trick is proposed to solve the issues with injective targets. In Section IV, we will evaluate GIA by 'bit drop' trick.

C. Theoretical Justification

From above statements, it is noticed that GI is related to the target function's value of a key hypothesis and the power consumption value at the target operation point. There are four combinations of key hypothesis (k^j) and trace point (o). We respectively use k and t_o for the right key hypothesis and the exact target operation point.

For right key hypothesis $k = k^j$ and right trace point $o = t_o$, the clustered dataset $D(o)$ contains the exact power consumption points and is classified into subsets by the right key hypothesis, so the subsets and the types are dependent, leading to a minimum GI value meaning the highest classification quality.

For right key hypothesis $k = k^j$ and wrong trace point $o \neq t_o$, the clustered dataset $D(o)$ contains the inexact power consumption points meaning the wrong and random type and is classified by the right key hypothesis, so the subsets and the types are independent, leading to a larger GI value. Although the classified subsets are proper in essence, the distinguisher regards it as an awful result.

For wrong key hypothesis $k \neq k^j$ and right trace point $o = t_o$, the clustered dataset $D(o)$ contains the exact power consumption points and is classified by the wrong key hypothesis meaning the wrong and random subsets, so the subsets and the types are independent, leading to a larger GI value.

For wrong key hypothesis $k \neq k^j$ and wrong trace point $o \neq t_o$, the clustered dataset $D(o)$ contains the inexact power consumption points meaning the wrong and random type and is classified by the wrong key hypothesis meaning the wrong and random subsets, so the subsets and the types are independent, leading to a larger GI value for the wrong classified subsets and types.

IV. Verification

In this section, we verify the feasibility of GIA experimentally. We also try one possible multivariate extension of GIA and analyze the impact of averaging operation on GIA. Our experiments were performed on an AT89S52 microcontroller running an implementation of AES-128 without any countermeasures. We obtained 8000 power traces from encrypting 8000 randomly chosen plaintexts for the first round operation. In the following analysis, we won't use any leakage information of our device which means that the leakage behavior is unavailable.

The 'bit drop' trick is applied to overcome the injective function issues mentioned in Section III. Bit drop means that

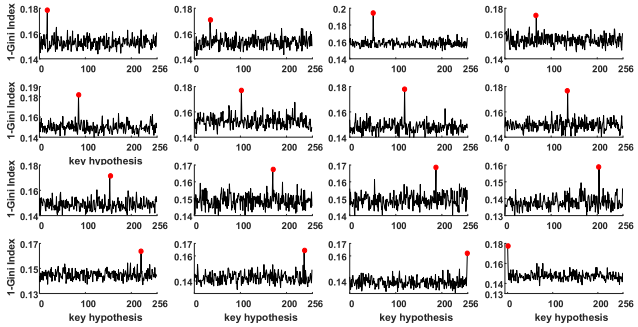


Fig. 3. Attack result for 16 key bytes using $drop_4(Sbox(p^i \oplus k^j))$ as the target function.

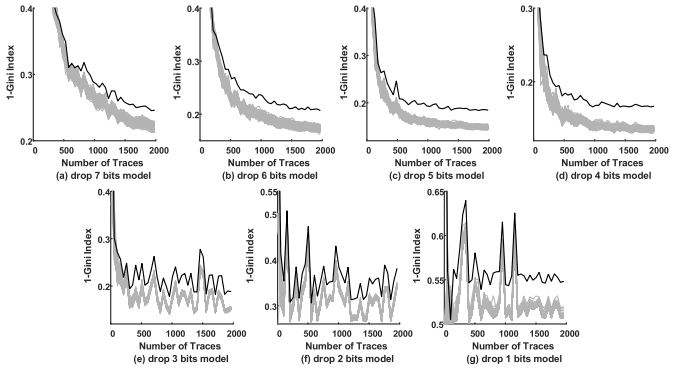


Fig. 4. Seven different attacks for the first key byte using different dropping bits.

when calculating the intermediate value $\{v_{i,j}\}_{i=1}^{N}$ of target function, we only care about some (not all) bits. For example, if the target function is S-box, $drop_1$ is referred to drop 1 bit of S-box's output, meaning that we only use 7 bits of S-box's output. In the following description, the target function is defined as:

$$\varphi\left(p^i, k^j\right) = drop_z\left(Sbox\left(p^i \oplus k^j\right)\right) \text{ z} \in \{7, 6, 5, 4, 3, 2, 1,\}.$$

A. Feasibility of GIA

We take $\varphi\left(p^i, k^j\right) = drop_4(Sbox(p^i \oplus k^j))$ as an example to show the feasibility of GIA. Dropping 4 bits means that we use 4 bits of S-box's output. Thus, there should be 16 candidates for the value. According to our attack strategy, there should be 16 types after clustering and the number of the subset is 16 based on the target function.

Fig.3 shows the attack result for 16 key bytes using 500 traces. For each sub-figure in Fig.3, x-axis refers to 256 key hypotheses with y-axis being the corresponding value of (1-gini_index). The maximum value is marked in red and corresponding to the right key value of each byte.

B. Efficiency Evaluation of GIA for Different Dropping Bits

The experiments from targeting $drop_4$ were performed repeatedly for other $drop_z$ function with z ranging from 1 to 7. The attack strategy is same for all the experiments except that the number of the types and subsets is different. As is shown in Fig. 4, there are 7 sub-plots for 7 different attacks for the first key byte. From left to right and from top to down, each sub-figure is corresponding to the increasing drop bits function

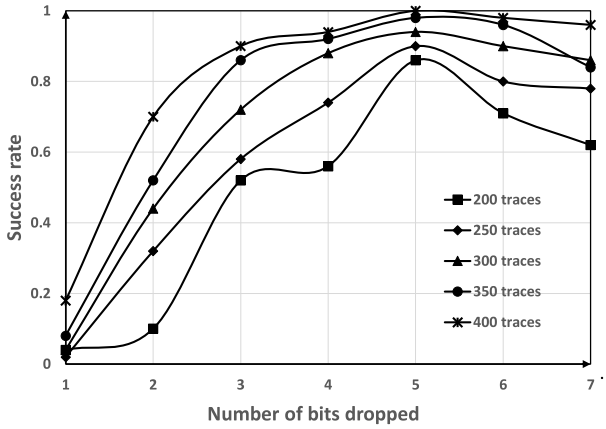


Fig. 5. Success rate vs number of bits dropped under different number of traces.

(the top and leftmost drops 1 bit and the down and rightmost drops 7 bits). The x-axis refers to the number of the traces and y-axis is the corresponding value of (1-gini_index). The correct key hypothesis is marked in black curve with wrong key hypothesis marked in grey curve.

We also study the relationship between the number of the dropped bits and the attack success rate under different number of traces. Each curve in Fig.5 represents a fixed amount of power consumption traces. The number of the power traces starts from 200 to 400 with increasing 50 traces each time.

As can be seen from Fig. 5, when the number of the dropped bits increases from 1 to 5, the success rate increases as well, but when the number of the dropped bits increases from 5 to 7, the success rate actually decreases. The reason may be that when the number of the dropped bits begins to increase, the target function becomes more non-injective, generating a better performance. However, when the number of the dropped bits is too large, the dropped bits are equivalent to noise for our attack resulting in the decreasing success rate. We may conclude that dropping 5 bits model outperforms other situations with the same number of traces.

C. Multivariate Extension of GIA

In k-means algorithm, each sample can be multi-dimensional, suggesting the multivariate nature of k-means. Thus GIA can be easily multivariate extended to utilize more than one trace point. For that the trace points closed to the target points are also related to the target function, we can choose more than one point around the target point to perform GIA. Each chosen point is one dimension of a multi-dimensional sample. The flow of multivariate GIA is the same as that of univariate GIA, except that the samples clustered in k-means algorithm is multi-dimensional. Fig. 6 shows the experimental comparison results for univariate GIA and multivariate GIA of 2 points, indicating that incorporation of more information can make slighter improvement than univariate GIA.

D. Impact of Averaging on GIA

Generally speaking, in traditional side-channel analysis, when pre-processing the power traces, averaging the power traces of the same partial plaintexts is beneficial for the attack performance under noise-setting. However, we found

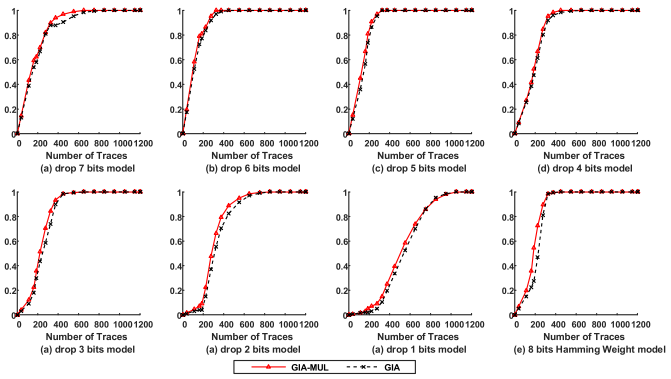


Fig. 6. Comparisons between multivariate GIA and univariate GIA.

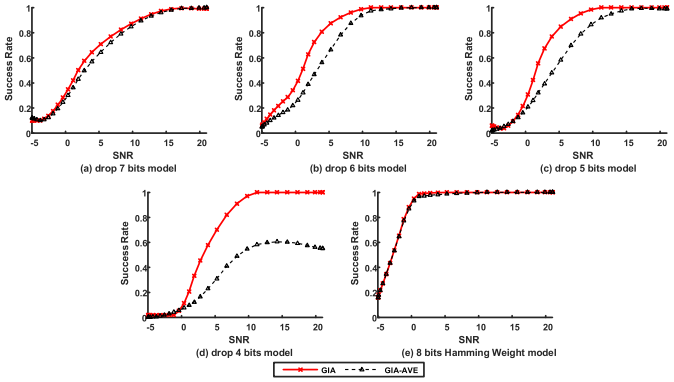


Fig. 7. Comparisons between GIA and GIA-AVE.

that averaging is not suitable for the proposed strategy from experimental results. We take the simulated Hamming-Weight-related leakage behavior targeted AES as an example (seen in Section V-B) to show our findings. For each attack operation, 1000 simulated leakages are utilized for evaluating the performance under Gaussian noise with varying sizes. In Fig. 7, red curves present the performance of GIA without averaging (denoted as GIA), and black curves present the performance of GIA with averaging simulated leakages of the same partial plaintexts (denoted as GIA-AVE). From Fig. 7, GIA always performs better than GIA-AVE in almost all the situations, especially when the number of the clusters is large, in which the superiority of GIA is more significant (shown in Fig. 7 (c) & (d)). This may be related to the k-means algorithm. In k-means algorithm, the centroid of each cluster (seen as μ_i in Section II-C) [28], needs to be calculated in each iteration, equivalent to an averaging operation without reducing the number of the samples. If averaging operation is done before k-means, the number of samples used for clustering will be substantially reduced, affecting the result of the clustering. In fact, the number of the samples has a significant effect on k-means algorithm. It can be concluded that the proposed GIA can make full use of all the leakage information with noise by taking advantage of the k-means algorithm.

V. Comparisons

In this section, we present the results of comparisons. In subsection A, B and C, we compare the proposed method GIA with three widely-used distinguishers (CPA, DPA, and MIA) under both simulation-based and practical leakage. The S-boxes of DES and AES, whose functions are non-injective and injective respectively, are chosen as the attack targets. For both simulated and practical experiments, we first compare the performance of these methods in a noise-free setting and then move on to the Gaussian noise setting with varying sizes. We use success rate, correct key ranking [35], and guessing entropy as evaluation metrics. Fig.10-25 in Appendix present the performance of above mentioned distinguishers under both simulation-based and practical leakage in both noise and noise-free settings. Table II reports the outcomes under simulation-based leakage with different scenarios in the noise-free setting and Table III reports the outcomes in the noise setting. In subsection D, we analyze the relationship between GIA and MIA theoretically and experimentally.

A. Simulated-Based Experiments Against First DES S-Box

For simulated experiments targeting the first S-box in DES, three leakage scenarios are employed to simulate the leakage behavior. Firstly, the simulated leakage (referred to SD1) equals to the Hamming weight of the output, which is an ideal and common leakage situation. Secondly, the simulated leakage (referred to SD2) equals to unevenly weighted sum of the bits of the S-box's output. Specifically, the least significant bit (LSB) is allowed to dominate with a relative weight of 10, and others to dominate with a weight of 1 [24], [26]. This leakage scenario is also a reasonable situation, where one bit of a data bus in a microcontroller leaks significantly more than others because of a larger capacitance [24]. Thirdly, the simulated leakage (referred to SD3) equals to the output of a highly nonlinear function of the S-box's output, such as a 4-bit S-box function of PRESENT [27]. This is a challenging but still realistic scenario, which resembles an encrypted bus scenario mentioned in [21]. Because of the non-injective function of DES S-box, we take the value and the Hamming weight of the target function as the leakage models to map the hypothetical leakage value. As for GIA, the leakage model is equivalent to the target function.

When the success rate can't reach 100%, correct key ranking, the position of the correct key ranked by the results of the attack method, is utilized to evaluate the method efficiency. If correct key ranking of the right key hypothesis is position 1, the attack is successful. Otherwise, it is in failure.

1) Experimental Results Under Noise-Free Setting: Fig.10-14 in Appendix and block 1-3 of Table II show the experimental results under three leakage scenarios (SD1, SD2 and SD3). As is shown in Fig.10 and block 1 of Table II, CPA outperforms the other methods with both leakage models and it can reach a success rate of 100% with fewer traces. It confirms that CPA shows a powerful attack ability when the leakage model is fit for the true leakage of the target device and the true leakage is linear distribution. However, for that Pearson's correlation coefficient can't capture all leakage details and only test for equidistance of cluster centroids, i.e. linearity, the performance of CPA is worse than others, and CPA shows no advantage under leakage strategy SD2 and SD3, shown in Fig. 11-12 and block 2-3 of Table II.

TABLE II

OUTCOMES OF DIFFERENT DISTINGUISHERS UNDER SIMULATION-BASED LEAKAGE WITH DIFFERENT SCENARIOS IN THE NOISE-FREE SETTING

	Correct key ranking				Traces for 90% SR				Traces for 100% SR			
	GIA	CPA	DPA	MIA	GIA	CPA	DPA	MIA	GIA	CPA	DPA	MIA
1.Attack against DES under SD1												
Value of S-box's output	1	1	1	1	26	16	36	29	32	21	48	40
HW model	1	1	1	1	46	21	62	46	51	41	64	51
2.Attack against DES under SD2												
Value of S-box's output	1	8	1	1	26	-	20	26	31	-	61	31
HW model	1	1	1	1	53	63	62	52	61	64	64	60
3.Attack against DES under SD3												
Value of S-box's output	1	14	4	1	25	-	-	26	32	-	-	36
HW model	1	10	4	1	56	-	-	59	61	-	-	61
4. Attack against AES under SA1												
Drop 7 bits model	1	1	1	1	95	73	74	98	156	81	83	156
Drop 6 bits model	1	1	1	1	80	53	53	76	99	62	62	96
Drop 5 bits model	1	1	1	1	94	46	46	94	130	58	60	120
Drop 4 bits model	1	1	1	1	100	32	56	92	230	56	76	221
HW model	1	1	1	1	6	6	226	10	10	10	256	26
5.Attack against AES under SA2												
Drop 7 bits model	1	1	1	1	6	6	6	6	10	10	10	10
Drop 6 bits model	1	1	1	1	26	80	36	16	36	200	40	36
Drop 5 bits model	1	2	1	1	30	-	56	28	40	-	60	40
Drop 4 bits model	1	10	1	1	80	-	78	78	145	-	150	136
Drop 3 bits model	1	81	1	1	162	-	93	168	200	-	198	200
HW model	1	1	63	1	16	190	-	16	40	256	-	40
6. Attack against AES under SA3												
Drop 7 bits model	58	58	58	58	-	-	-	-	-	-	-	-
Drop 6 bits model	59	61	41	69	-	-	-	-	-	-	-	-
Drop 5 bits model	44	130	102	53	-	-	-	-	-	-	-	-
Drop 4 bits model	24	39	122	29	-	-	-	-	-	-	-	-
HW model	181	193	197	193	-	-	-	-	-	-	-	-

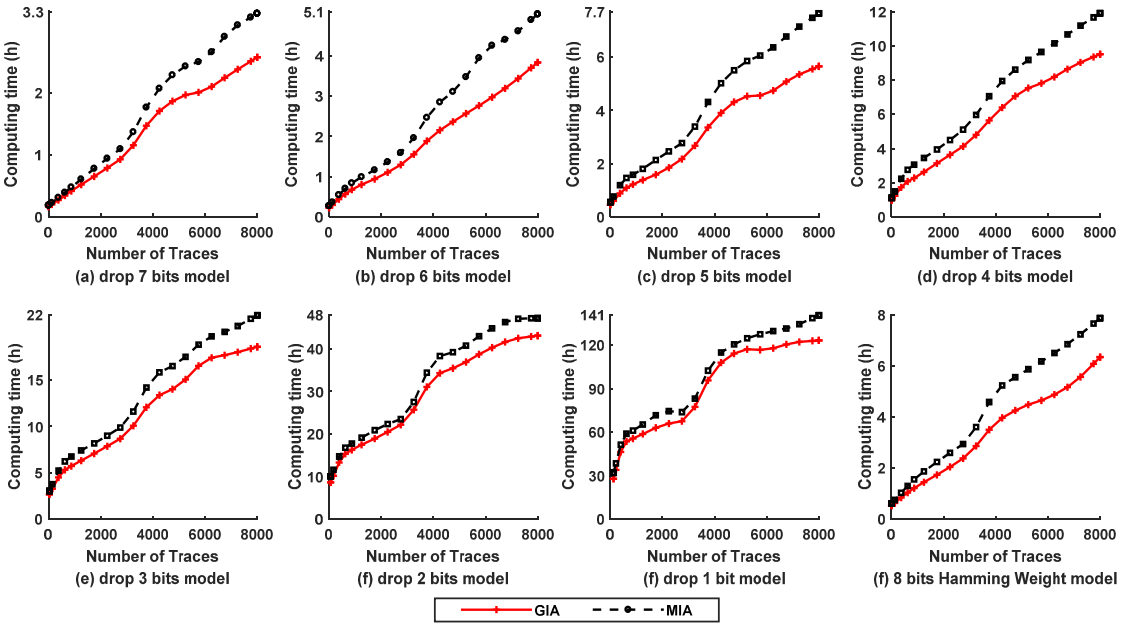


Fig. 8. Information Entropy and Gini Value.

DPA is usually less efficient than other methods because of the limited and inadequate use of leakage information. However, the ability to test relations other than linearity makes DPA outperform CPA on the aspects of success rate and correct key ranking under some leakage strategies (SD2 and SD3), shown in block 2-3 of Table II.

As is shown in Fig. 10-13 and block 1-3 of Table II, GIA and MIA have almost the same performances, and they can succeed in any given leakage scenarios with any leakage models. GIA presents the increasing advantage as the true leakage diverges from the ideal linearity power (SD1). Even under SD3, GIA and MIA are still able to identify the correct key hypothesis whilst both CPA and DPA fail to do that, shown

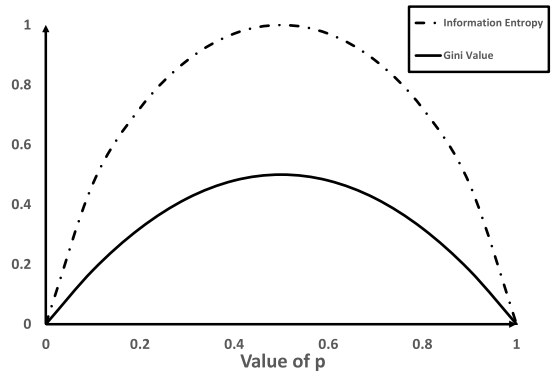


Fig. 9. Computing time of GIA and MIA for different target function and number of traces.

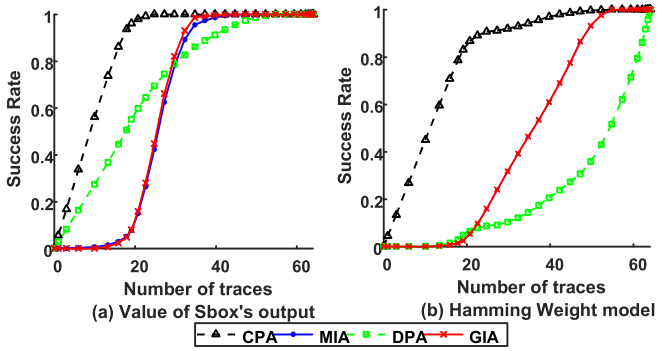


Fig. 10. Success rate of each attack strategy as number of traces varies against the first DES S-box under SD1.

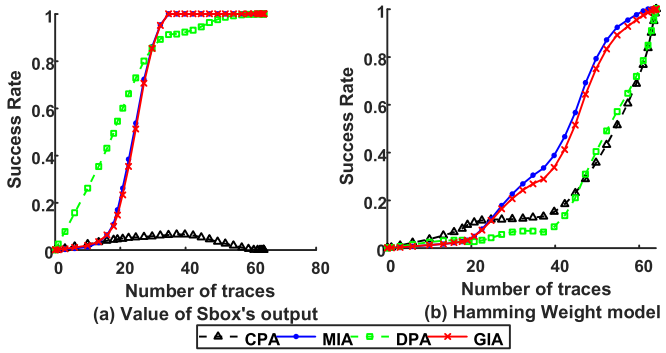


Fig. 11. Success rate of each attack strategy as number of traces varies against the first DES S-box under SD2.

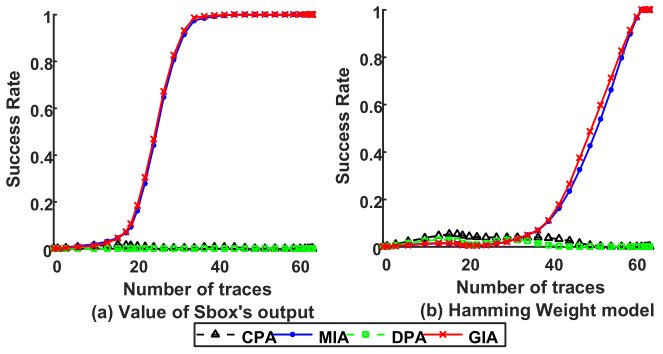


Fig. 12. Success rate of each attack strategy as number of traces varies against the first DES S-box under SD3.

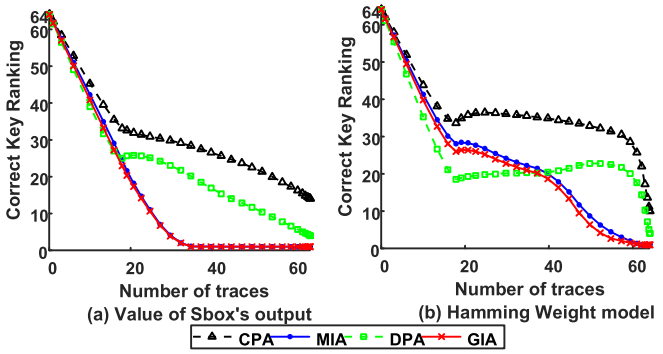


Fig. 13. Correct key ranking of each attack strategy as number of traces varies against the first DES S-box under SD3.

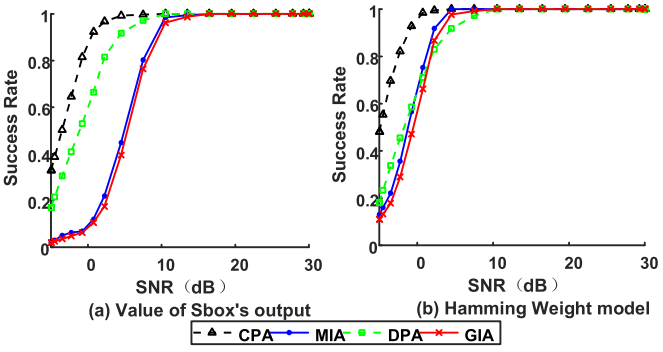


Fig. 14. Success rate of each attack strategy as SNR varies against the first DES S-box under SD1.

in Fig. 12 and block 3 of Table II. The performances of GIA with different leakage models are different. And it seems that *the value model* fits all three leakage scenarios better than Hamming weight model. Furthermore, attack results of GIA with the value model, shown in Fig. 10 (a), Fig. 11 (a) and Fig. 12 (a) demonstrate the same relationship between success rate and the number of traces under three leakage scenarios. It appears that GIA, with value model to recover the right key guess, is independent of the true leakage scenario, which is also demonstrated by the outcomes shown in block 1-3 of Table II. Meanwhile, the value model is easy to come up with and essentially it directly uses the value of target function as the hypothetical leakage. Hence, GIA is a generic and universal attack method regardless of what the leakage scenario is, when the target function is non-injective.

2) Experimental Results Under Gaussian Noise Setting: In order to evaluate how Gaussian noise affects the attack efficiency, we compare the performance of different distinguishers under Gaussian noise environment of varying sizes. SNR (signal-to-noise ration) is used for indicating the intensity of the noise, defined as follows:

$$\mathrm{SNR} = 10\log_{10}\frac{P_{signal}}{P_{noise}}.$$

Generally, lower SNR means more true leakage data and device information are needed to recover the right key hypothesis. Experimental results under three simulated leakage scenarios are shown in Fig. 14-17 and block 1-3 of Table III. The value of SNR ranges from -5dB to 30dB, and 300 simulated leakages are utilized for each attack operation.

TABLE III

OUTCOMES OF DIFFERENT DISTINGUISHERS UNDER SIMULATION-BASED LEAKAGE WITH DIFFERENT SCENARIOS IN THE NOISE SETTING

	GIA					CPA					DPA					MIA				
SNR(dB)	**-5**	**0**	**10**	**20**	**30**	**-5**	**0**	**10**	**20**	**30**	**-5**	**0**	**10**	**20**	**30**	**-5**	**0**	**10**	**20**	**30**
1. DES under SD1		**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**			
S-box's output	2	7	95	100	100	33	90	100	100	100	17	59	100	100	100	2	6	96	100	100
HW model	11	56	100	100	100	48	98	100	100	100	18	65	100	100	100	13	25	100	100	100
2. DES under SD2		**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**			
S-box's output	12	47	100	100	100	5	5	0	4	5	18	40	88	100	100	14	47	100	100	100
HW model	20	59	100	100	100	42	81	100	100	100	17	45	90	100	100	19	57	95	100	100
3. DES under SD3		**Success Rate (%)**					**Correct Key Ranking**					**Correct Key Ranking**					**Success Rate (%)**			
S-box's output	9	12	100	100	100	33	31	23	21	21	33	27	14	10	9.5	6	16	100	100	100
HW model	2	5	68	100	100	34	28	23	18	17	33	25	16	9	9.3	2	4	71	100	100
4. AES under SA1		**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**			
Drop 7 bits model	10	30	86	96	100	22	48	98	100	100	22	48	98	100	100	10	30	86	96	100
Drop 6 bits model	6	48	100	100	100	32	94	100	100	100	40	94	100	100	100	14	56	100	100	100
Drop 5 bits model	6	52	100	100	100	48	96	100	100	100	38	96	100	100	100	8	48	100	100	100
Drop 4 bits model	2	2	100	100	100	52	100	100	100	100	32	84	100	100	100	0	18	100	100	100
HW model	16	98	100	100	100	100	100	100	100	100	0	14	74	96	96	28	100	100	100	100
5. AES under SA2		**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**					**Success Rate (%)**			
Drop 7 bits model	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Drop 6 bits model	100	100	100	100	100	94	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Drop 5 bits model	100	100	100	100	100	18	46	80	82	94	100	100	100	100	100	100	100	100	100	100
Drop 4 bits model	82	100	100	100	100	2	2	8	4	6	64	100	100	100	100	100	100	100	100	100
Drop 3 bits model	18	100	100	100	100	0	0	0	0	0	12	60	98	100	100	22	100	100	100	100
HW model	2	26	82	90	100	74	98	100	100	100	4	4	2	4	72	4	34	90	96	100
6. AES under SA3		**Guessing Entropy**					**Guessing Entropy**					**Guessing Entropy**					**Guessing Entropy**			
Drop 7 bits model	112	111	107	98	102	112	111	107	104	104	112	111	107	104	104	112	112	108	100	103
Drop 6 bits model	109	110	92	72	82	110	106	107	95	93	110	102	95	89	77	109	110	96	91	94
Drop 5 bits model	112	111	107	96	98	113	112	116	117	117	112	111	112	109	110	112	112	108	100	98
Drop 4 bits model	114	110	107	99	93	111	108	100	92	96	115	114	113	111	112	114	113	110	100	91
HW model	109	113	118	118	119	112	116	120	119	120	111	113	116	118	117	111	117	120	121	122

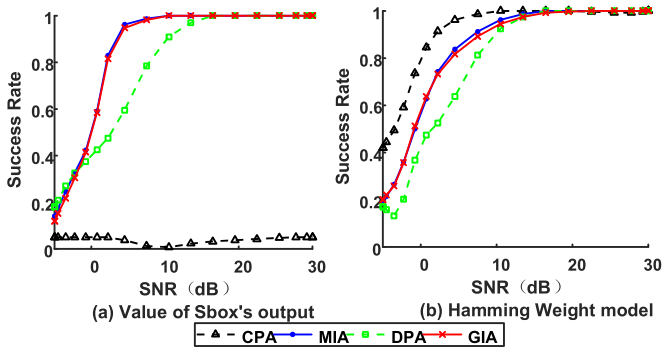


Fig. 15. Success rate of each attack strategy as SNR varies against the first DES S-box under SD2.

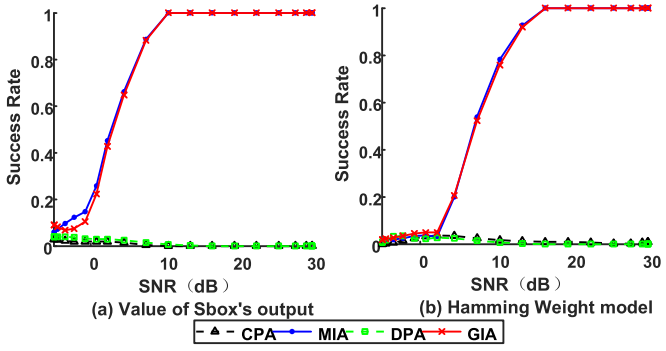


Fig. 16. Success rate of each attack strategy as SNR varies against the first DES S-box under SD3.

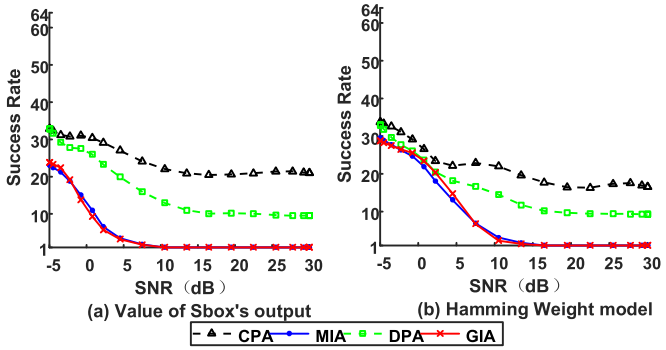


Fig. 17. Correct key ranking of each attack strategy as SNR varies against the first DES S-box under SD3.

Under the ideal leakage scenario (SD1), similar to noise-free setting, CPA outperforms the other methods with both leakage models, as shown in Fig.14 and block 1 of Table III. It can be attributed to the powerful capability of Pearson's correlation coefficient to resist noise when the tested samples are linear distribution. There seems to be an optimal SNR value, from which the success rate can reach up to 100%. The optimal value for DPA, GIA and MIA seems almost the same.

Under the leakage strategy SD2, the distribution is not ideally linear. When the leakage model is not suitable, CPA remains unsuccessful across the test range, shown in Fig.15(a) and block 2 of Table III. As is shown in Fig. 15(b), the optimal SNR value for all four attack methods are the same. But when it comes to the value model, shown in Fig. 15(a), the optimal SNR value for DPA is much larger than that for GIA and MIA.

In Fig. 16 and block 3 of Table III, both GIA and MIA obtain a low success rate when the value of SNR is small, suggesting

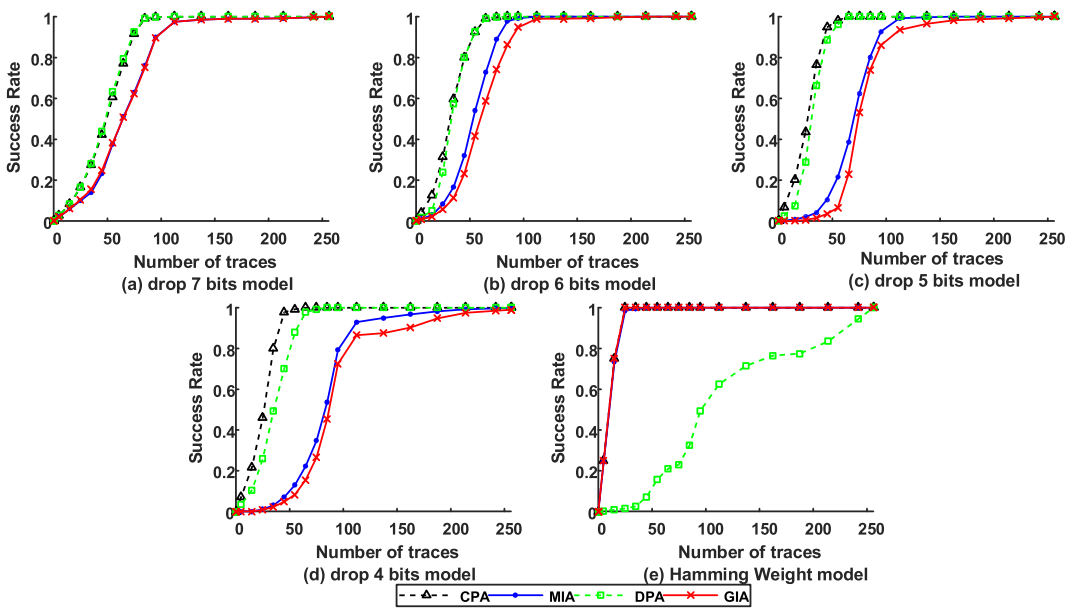


Fig. 18. Success rate of each attack strategy as number of traces varies against the first AES S-box under leakage scenario SA1.

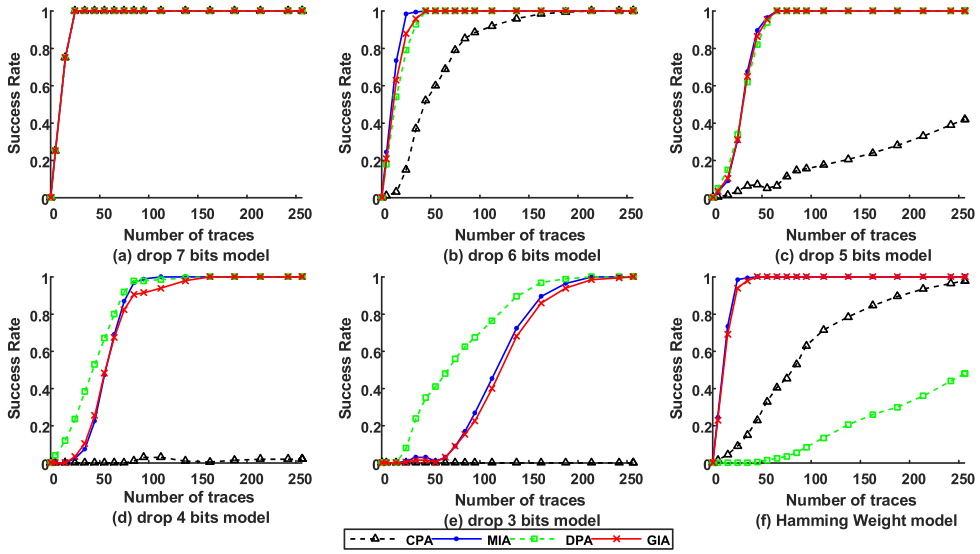


Fig. 19. Success rate of each attack strategy as number of traces varies against the first AES S-box under leakage scenario SA2.

a high level of noise, but the success rates of GIA and MIA increase with the value of SNR and can even reach 100%. That is to say, GIA and MIA have a good performance under the leakage strategy SD3 with the impact of noise while CPA and DPA fail across the test range. In addition, according to the correct key ranking value shown in Fig. 17 and block 3 of Table III, DPA even outperforms CPA.

B. Simulated-Based Experiments Against First AES S-Box

For simulated experiments targeting the first AES S-box, three leakage scenarios are employed to simulate the leakage behavior. The first two leakage scenarios (referred to SA1 and SA2) are the same as SD1 and SD2. However, the third leakage (referred to SA3) is similar to SD3, just changing the S-box function of PRESENT to that of AES. Because the S-box function of AES is injective, we can't directly choose the output value of S-box in AES encryption as a leakage model. Drop bits model mentioned in Section IV and Hamming weight model are used as leakage model. Under some leakage strategies, attack methods remain unsuccessful, so we choose guessing entropy (in Fig. 21 & 23, Table III) and correct key ranking (in Table II and III) to show the attack outcomes. The lower the guessing entropy is, the better the attack outcomes are.

1) Experimental Results Under Noise-Free Setting: Fig. 18-20 and block 4-6 of Table II show the experimental results under three leakage scenarios (SA1, SA2 and SA3). As in the case for DES, CPA outperforms the other methods with both leakage models under SA1, shown in Fig. 18 and block 4 of Table II. But as shown in Fig. 19 and block 5 of Table II, when it comes to the leakage strategies SA2, CPA can

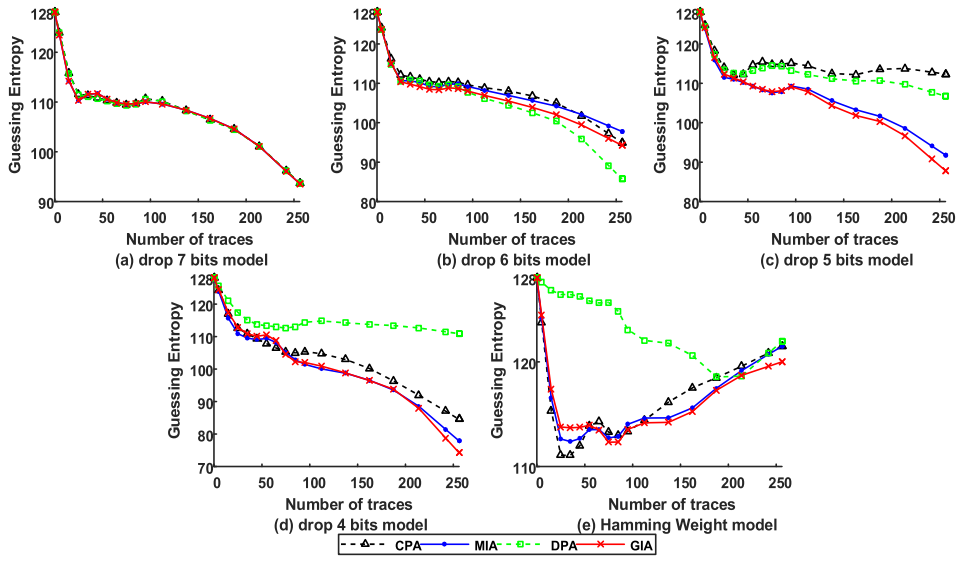


Fig. 20. Guessing Entropy of each attack strategy as number of traces varies against the first AES S-box under leakage scenario SA3.

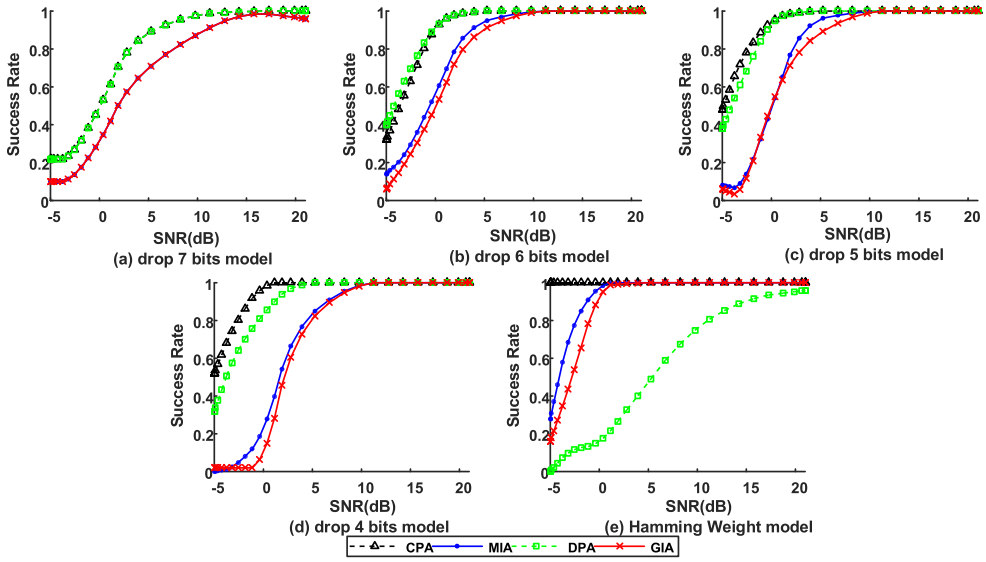


Fig. 21. Success rate of each attack strategy as SNR varies against the first AES S-box under leakage scenario SA1.

only be successful with the drop 6 and 7 bits model as well as the Hamming Weight model, and it fails to recover the correct key hypothesis with other leakage models. DPA shows good performance under the first two leakage strategies SA1 and SA2, except with the Hamming weight model. GIA and MIA still present similar results under SA1 and SA2 and show their generic attack ability with any leakage models. It seems that the performances of GIA and MIA get worse as the number of the dropped bits decreases, especially when it is less than 5.

However, all four attack methods with different leakage models fail to reveal the right key under the challenging SA3. Fig.20 and block 6 of Table II show that GIA and MIA obtain the better outcomes on guessing entropy and correct key ranking. In addition, GIA always has a lower guessing entropy and correct key ranking than MIA. The guessing entropy of GIA is lower than MIA by up to 17%.

2) Experimental Results Under Gaussian Noise Setting: For each attack operation, 1000 simulated leakages are utilized for evaluating the performance under Gaussian noise with varying sizes. Fig. 21-23 and block 4-6 of Table III show the experimental results. The performances of these four attack methods with noise are similar to those without noise. In Fig. 23 and block 6 of Table III, it can be also concluded that GIA always has a lower guessing entropy than MIA. Especially, under the drop 6 bits model (shown in Fig. 23 (b)), the guessing entropy of GIA is lower than MIA by up to 21%.

There is an interesting finding from the experimental results. In Section IV, we give the conclusion from practical experiment results that dropping 5 bits model outperforms other dropping models with the same number of traces. This conclusion can be verified by the results of the simulated experiments in this section. As shown in Fig. 21-22 and block 4-5 of

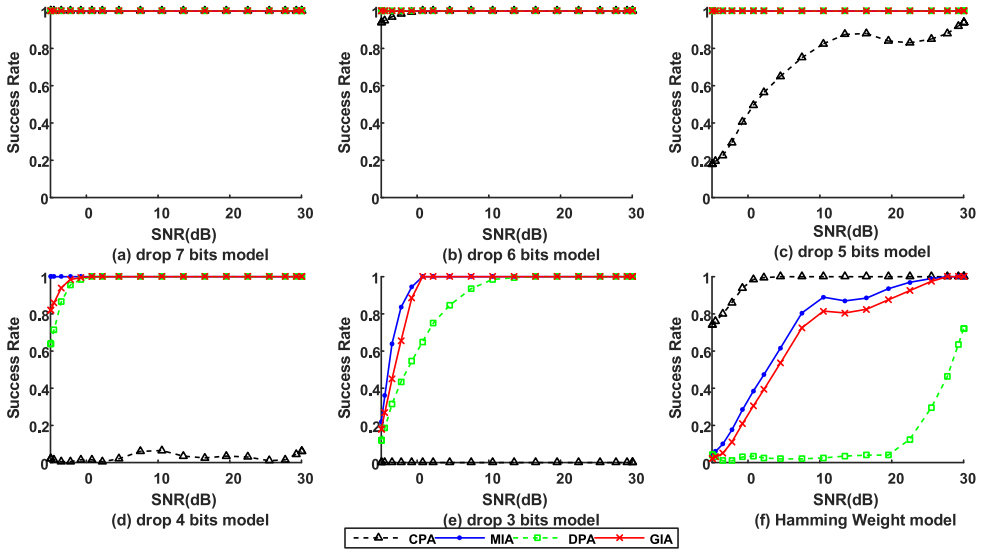


Fig. 22. Success rate of each attack strategy as SNR varies against the first AES S-box under leakage scenario SA2.

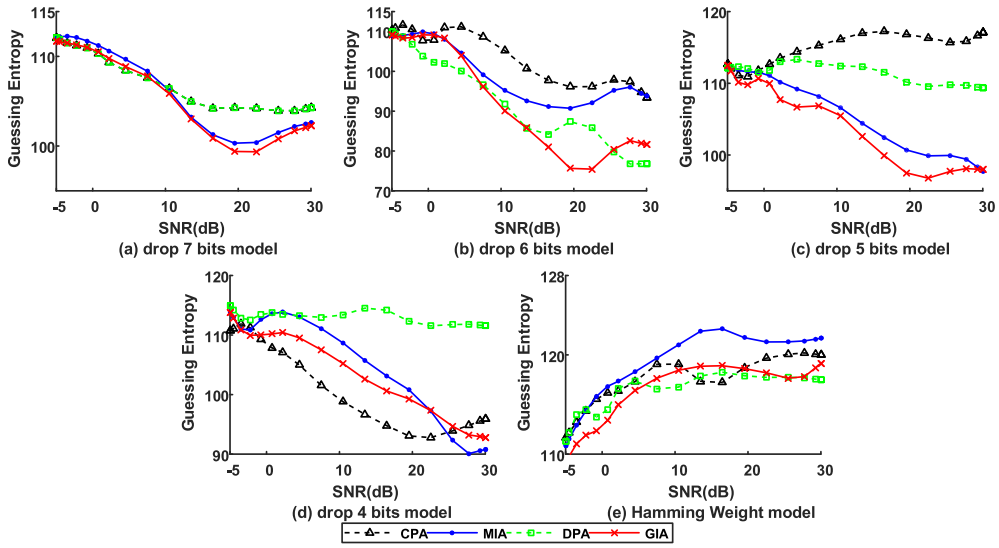


Fig. 23. Guessing Entropy of each attack strategy as SNR varies against the first AES S-box under leakage scenario SA3.

Table III, when the number of bits drops decreased from 7 to 5, the performances of GIA get better or at least not worse. But when the number of bits is less than 5, the performances get worse. The simulated results are similar to the practical results.

C. Practical Experiments

Our practical experiments were also performed on an AT89S52 microcontroller running an implementation of AES-128, the same as that in Section IV. The target function is also the output of S-box. Dropping from 7 to 1 bit models and Hamming weight model are used as the leakage models. For each attack operation, 2000 simulated leakages are utilized for evaluating the performance under noise. Fig. 24 and Fig. 25 demonstrate the comparison results with and without noise, respectively. Due to the target device is of Hamming-weight-related leakage, the results are similar to the simulated experiments targeting AES under leakage scenario SA1 shown in Fig.18 and 21. CPA still shows its powerful ability with the suitable leakage model and linear distributed power consumption. However, when it comes to the drop 1 bit model under noise setting, using 7 bits of S-box's output as the target function, the nearly injective function renders that GIA can't always be successful throughout the whole tested range.

D. Relationship Between GIA And MIA

Above experimental results show that GIA and MIA have similar performances, and they can succeed in most of the given leakage scenarios with any leakage models. In this subsection, we analyze the relationship between GIA and MIA theoretically and experimentally, and discuss the superiority of GIA.

1) Computational Relation: As mentioned in Section II, Mutual Information value depends on information entropy and Gini-impurity index depends on Gini value. The expression of information entropy shown as (5) is almost based on the logarithm operation. According to Taylor series expansion,

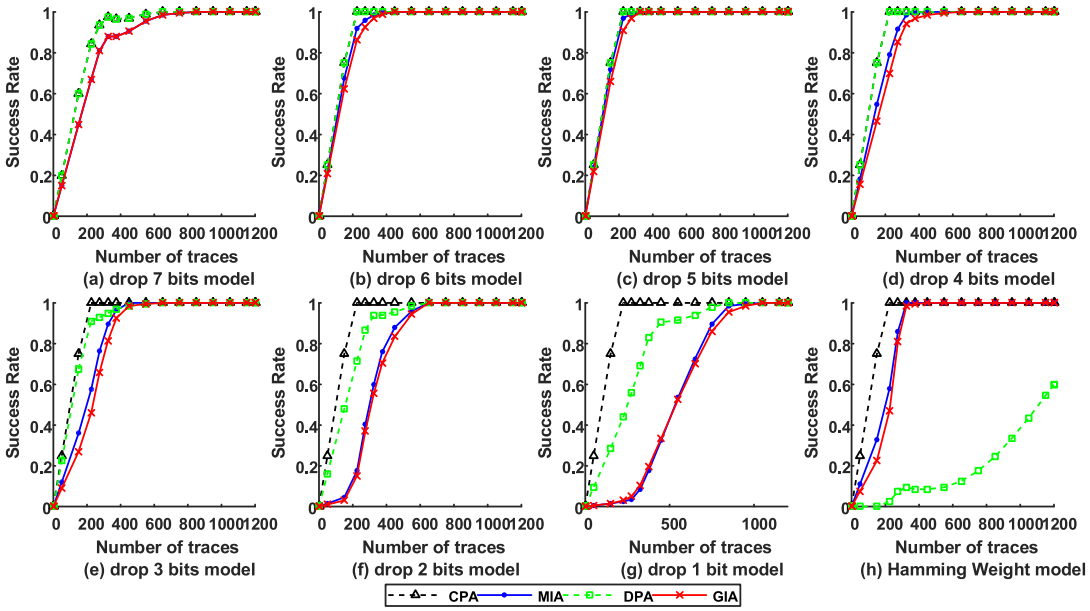


Fig. 24. Success rate of each attack strategy as number of traces varies against practical device.

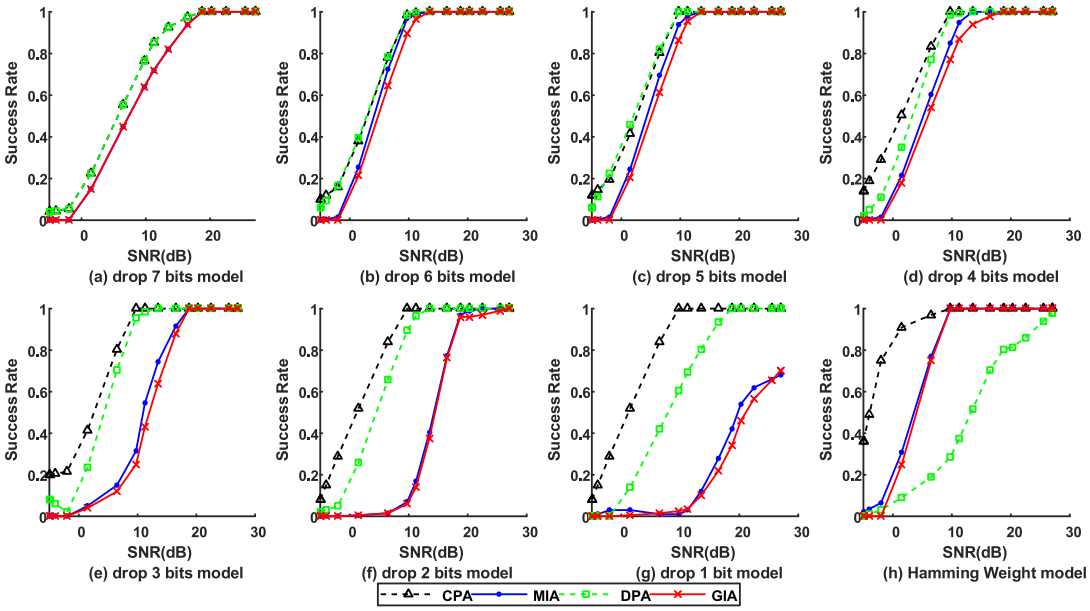


Fig. 25. Success rate of each attack strategy as SNR varies against practical device.

the logarithm operation can be written as:

$$\begin{aligned} f\left(p\left(x\right)\right) &= log_2 p\left(x\right) \\ &\approx f\left(p\left(x_0\right)\right) + f'\left(p\left(x_0\right)\right)\left(p(x) - p(x_0)\right) \\ &= log_2 p(x_0) + \frac{1}{ln2}\frac{1}{p(x_0)}(p(x) - p(x_0)). \end{aligned}$$

Due to $0 \leq p\left(x_0\right) \leq 1$, we choose $p\left(x_0\right) = 1$ and thus

$$log_2 p\left(x\right) \approx \frac{1}{ln2}\left(p\left(x\right) - 1\right). \tag{14}$$

Combined (14) with (5), we obtain that:

$$\begin{aligned} H\left(X\right) &= -\sum_{i=1}^{n} p\left(x_i\right) log_2 p\left(x_i\right) \\ &\approx \frac{1}{ln2}\sum_{i=1}^{n} p\left(x_i\right)\left(1 - p\left(x_i\right)\right) \\ &= \frac{1}{ln2}\sum_{i=1}^{n}\sum_{i' \neq i} p\left(x_i\right) p\left(x_{i'}\right) \\ &= \frac{1}{ln2} Gini\left(D\right) \end{aligned} \tag{15}$$

From (15), we can conclude that the expression of Gini value is equivalent to the first-order Taylor series expansion of the information entropy's expression at the fixed point of $p(x_0) = 1$. Figure 8 shows information entropy as well as Gini value when $n = 2$, from which we can see that both values have the same tendency. Table II, III and Figure 10-25 confirm that GIA and MIA always obtain similar performance under any circumstances. However, according to block 6 of Table II, III and Fig. 20 & 23, under the situation where the adversary fails to recover the key value, the guessing entropy of GIA is always lower than MIA by up to 21%, making brute-force more efficient [34].

2) Computational Time Overhead: Figure 9 shows the time consumption of one GIA or MIA operation. To record the computational time, we utilize the power traces of the AT89S52 microcontroller. An Agilent MSO-X 9104A oscilloscope is employed to collect original power traces. And each power trace contains about 60000 points. The attack target is the AES S-box. We choose 8 models (drop from 1bit to 7bits models and Hamming Weight model) to make the comparisons, and we do the simulations in Matlab to record and compare the offline computing time, reflecting the computational complexity and overhead of the attack strategy. The offline computational time is the total time for calculating the distinguisher values of all points of the traces with all key hypotheses. As is shown in Fig. 25, the offline computational time of both methods increases with the number of the power traces and the subsets. However, under any situation, the computational complexity and overhead of MIA is significantly higher than GIA. GIA have a lower computational time overhead than MIA by up to 13.3%. The logarithm operation makes MIA low efficient in time consumption, and zero value can cause logarithm operation failure. However, the adversary often increases the sampling frequency for enhancing attack performance and the number of the sample points that each trace contains increases greatly with the sampling frequency, causing the attack more time-consuming. Thus, reducing computational time overhead is crucial for efficiency improvement.

3) Discussion: As mentioned above, in terms of the guessing entropy and time overhead, GIA is more efficient than MIA.

Traditional MIA utilizes methods such as histogram estimation and kernel estimation to estimate the probability density function. The quality of the pdf estimation has a significant effect on the result and noise in the leakage can significantly influence which estimation tool is the best [37]. However, for GIA combined with k-means, the Gini-impurity index value can be calculated directly rather than estimating probability density function like MIA. And as mentioned in Section IV-D, GIA can make full use of leakage information with noise. Thus, in terms of flexibility, GIA combined with k- means algorithm is easier to operate and more flexible with noise than traditional MIA.

VI. Conclusion

In this paper, we present a novel generic side-channel analysis method called GIA. We introduce the basic idea and process of GIA. We also verify the feasibility and evaluate the efficiency by the practical experimental results. One possible multivariate extension of GIA with slighter improvement is also presented. And by analyzing the impact of averaging operation, we find that GIA can make full use of all the leakage information with noise by taking advantage of k-means algorithm. In addition, we compare GIA with three widely-used distinguishers under both simulation-based and practical leakages. Results confirm that GIA is a generic method and can always perform well under different situations. Furthermore, we analyze the relationship between GIA and MIA. We theoretically prove that GIA and MIA can obtain similar results, confirmed by the experimental results. However, computation without logarithm operation makes GIA have a lower computational time overhead than MIA by up to 13.3% and when the attack fails, the guessing entropy of GIA is lower than MIA by up to 21%. In terms of flexibility, GIA combined with k-means shows advantages over traditional MIA. The aim of proposing GIA is to make attack more universal with limited extra overhead. Therefore, GIA is an efficient and useful alternative to these commonly-used strategies.

Appendix
Comparisons Among GIA, CPA, DPA and MIA

Fig.10-25 present the performance of GIA, CPA, DPA and MIA under both simulation-based and practical leakage in both noise and noise-free settings.

Acknowledgment

The authors would like to thank Yijun Yang, Yanzhao Yin, and Ya Han for their meaningful contributions about this work and thank the reviewers for providing their valuable suggestions.

References

[1] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Advances in Cryptology—CRYPTO'96* (Lecture Notes in Computer Science), vol. 1109. Berlin, Germany: Springer, 1996, pp. 104–113.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology—CRYPTO'99* (Lecture Notes in Computer Science), vol. 1666. Berlin, Germany: Springer, 1999, pp. 388–397.

[3] C. D. Walter and S. Thompson, "Distinguishing exponent digits by observing modular subtractions," in *Topics in Cryptology—CT-RSA 2001* (Lecture Notes in Computer Science), vol. 2020. Berlin, Germany: Springer, 2001, pp. 192–207.

[4] A. Moradi and A. Wild, "Assessment of hiding the higher-order leakages in hardware," in *Cryptographic Hardware and Embedded Systems—CHES 2015* (Lecture Notes in Computer Science), vol. 9293. Berlin, Germany: Springer, 2015, pp. 453–474.

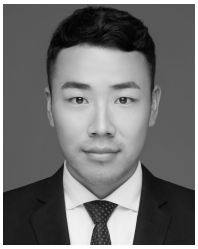
[5] J. J. Quisquater and D. Samyde, "ElectroMagnetic analysis (EMA): Measures and counter-measures for smart cards," in *Smart Card Programming and Security* (Lecture Notes in Computer Science), vol. 2140. Berlin, Germany: Springer, 2001, pp. 200–210.

[6] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side—Channel(s)," in *Cryptographic Hardware and Embedded Systems—CHES 2002* (Lecture Notes in Computer Science), vol. 2523. Berlin, Germany: Springer, 2002, pp. 29–45.

[7] K. Gandolfi, C. Mourtel, and F. Olivier, "Electromagnetic analysis: Concrete results," in *Cryptographic Hardware and Embedded Systems—CHES 2001* (Lecture Notes in Computer Science), vol. 2162. Berlin, Germany: Springer, 2001, pp. 251–261.

[8] J. F. Dhem, F. Koeune, P. A. Leroux, P. Mestré, J. J. Quisquater, and J. L. Willems, "A practical implementation of the timing attack," in *Smart Card Research and Applications* (Lecture Notes in Computer Science), vol. 1820. Berlin, Germany: Springer, 1998, pp. 167–182.

[9] P. Kocher, J. Jaffe, P. Rohatgi, and B. Jun, "Introduction to differential power analysis," *J. Cryptograph. Eng.*, vol. 1, no. 1, pp. 5–27, 2011.
[10] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems—CHES 2004* (Lecture Notes in Computer Science), vol. 3156. Berlin, Germany: Springer, 2004, pp. 16–29.
[11] E. Oswald, S. Mangard, C. Herbst, and S. Tillich, "Practical second-order DPA attacks for masked smart card implementations of block ciphers," in *Topics in Cryptology—CT-RSA 2006* (Lecture Notes in Computer Science), vol. 3860. Berlin, Germany: Springer, 2006, pp. 192–207.
[12] S. Chair, J. R. Rao, and P. Rohatgi, "Template attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2002* (Lecture Notes in Computer Science), vol. 2523. Berlin, Germany: Springer, 2002, pp. 13–28.
[13] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Investigations of power analysis attacks on smartcards," in *Proc. USENIX Workshop Smartcard Technol. USENIX Workshop Smartcard Technol. (WOST)*, 1999, p. 17.
[14] T. S. Messerges, "Using second-order power analysis to attack DPA resistant software," in *Cryptographic Hardware and Embedded Systems—CHES 2000* (Lecture Notes in Computer Science), vol. 1965. Berlin, Germany: Springer, 2000, pp. 238–251.
[15] D. Agrawal, J. R. Rao, and P. Rohatgi, "Multi-channel attacks," in *Cryptographic Hardware and Embedded Systems—CHES 2003* (Lecture Notes in Computer Science), vol. 2779. Berlin, Germany: Springer, 2003, pp. 2–16.
[16] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. New York, NY, USA: Springer-Verlag, 2007, pp. 40–42.
[17] F. Mola and R. Siciliano, "A fast splitting procedure for classification trees," *Statist. Comput.*, vol. 7, no. 3, pp. 209–216, 1997.
[18] L. Rokach and O. Maimon, "Top-down induction of decision trees classifiers—A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev*, vol. 35, no. 4, pp. 476–487, Nov. 2005.
[19] L. Batina, B. Gierlichs, and K. Lemke-Rust, "Differential cluster analysis," in *Cryptographic Hardware and Embedded Systems—CHES 2009*. Berlin, Germany: Springer, 2009, pp. 112–127.
[20] J. Yi, M. Yin, Y. Zhang, and X. Zhao, "A novel recommender algorithm using information entropy and secondary-clustering," in *Proc. 2nd IEEE Int. Conf. Comput. Intell. Appl. (ICCIA)*, Beijing, China, Sep. 2017, pp. 128–132.
[21] O. Reparaz, B. Gierlichs, and I. Verbauwhede, "Generic DPA attacks: Curse or blessing?" in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design (COSADE)*. Berlin, Germany: Springer, 2014, pp. 98–111.
[22] N. Veyrat-Charvillon and F. X. Standaert, "Generic Side-Channel Distinguishers: Improvements and Limitations," in *Advances in Cryptology—CRYPTO 2011*. Berlin, Germany: Springer, 2011, pp. 354–372.
[23] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Cryptographic Hardware and Embedded Systems—CHES 2008*. Berlin, Germany: Springer, 2008, pp. 426–442.
[24] N. Veyrat-Charvillon and F. X. Standaert, "Mutual information analysis: how, when and why?" in *Cryptographic Hardware and Embedded Systems—CHES 2009*, vol. 5747. Berlin, Germany: Springer, 2009, pp. 429–443.
[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Chichester, U.K.: Wiley, 1991, pp. 103–175.
[26] C. Whitnall and E. Oswald, "A fair evaluation framework for comparing side-channel distinguishers," *J. Cryptogr. Eng.*, vol. 1, pp. 145–160, Aug. 2011.
[27] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems—CHES 2008*. Berlin, Germany: Springer, 2008, pp. 450–466.
[28] S. Shah and M. Singh, "Comparison of a time efficient modified K-mean algorithm with K-mean and K-medoid algorithm," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, Rajkot, India, May 2012, pp. 435–437.
[29] W. Wang, Y. Yu, F.-X. Standaert, J. Liu, Z. Guo, and D. Gu, "Ridge-based DPA: Improvement of differential power analysis for nanoscale chips," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1301–1316, May 2018.
[30] H. Gupta, S. Sural, V. Atluri, and J. Vaidya, "A side-channel attack on smartphones: Deciphering key taps using built-in microphones," *J. Comput. Secur.*, vol. 26, no. 2, pp. 255–281, Jan. 2018.
[31] H. Kim, H. Yoon, Y. Shin, and J. Hur, "Cache side-channel attack on mail user agent," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Barcelona, Spain, Jan. 2020, pp. 236–238.
[32] P. S. Bhattacharjee, A. K. Md Fujail, and S. A. Begum, "A comparison of intrusion detection by K-means and fuzzy C-means clustering algorithm over the NSL-KDD dataset," in *Proc. IEEE Int. Conf. Comput. Intell. Comput. Res. (ICCIC)*, Coimbatore, India, Dec. 2017, pp. 1–6.
[33] Y. Souissi, S. Bhasin, S. Guilley, M. Nassar, and J.-L. Danger, "Towards different flavors of combined side channel attacks," in *Topics in Cryptology—CT-RSA 2012* (Lecture Notes in Computer Science), vol. 7178. Berlin, Germany: Springer, 2012, pp. 245–259.
[34] M. O. Choudary and P. G. Popescu, "Back to massey: Impressively fast, scalable and tight security evaluation tools," in *Cryptographic Hardware and Embedded Systems—CHES 2017*. Berlin, Germany: Springer, 2017, pp. 367–386.
[35] F. X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Advances in Cryptology—EUROCRYPT 2009* (Lecture Notes in Computer Science), vol. 5479. Berlin, Germany: Springer, 2009, pp. 443–461.
[36] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl, "Clustering algorithms for non-profiled single-execution attacks on exponentiations," in *Smart Card Research and Advanced Applications* (Lecture Notes in Computer Science), A. Francillon and P. Rohatgi, Eds, vol. 8419. Berlin, Germany: Springer, 2014, pp. 79–93.
[37] E. Prouff and M. Rivain, "Theoretical and practical aspects of mutual information based side channel analysis," in *Applied Cryptography and Network Security. ACNS 2009* (Lecture Notes in Computer Science), vol. 5536. Berlin, Germany: Springer, 2009, pp. 499–518.



Ye Yuan was born in 1993. He received the B.S. degree from the School of Astronautics, Harbin Institute of Technology, in 2015. He is currently pursuing the Ph.D. degree with the Institute of Microelectronics, Tsinghua University. His research interests include side-channel attack and side-channel attack resistant implementation in cryptography.

Liji Wu (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1988, 1991, and 1997, respectively. From April 1997 to May 2000, he worked with the Center for Advanced Technology in Telecommunications (CATT), Polytechnic Institute of New York University (NYU-Poly), Brooklyn, NY, USA, as a Postdoctoral Fellow, and worked on design and implementation of high-speed control circuits and systems utilized in WDM ATM multicast optical switching systems sponsored by DARPA. Then he worked in High-Tech industry, USA, for more than four years, including TyCom Laboratories (former AT&T Bell Labs on Undersea Optical Fiber Communications), Eatontown, NJ, USA, as a Senior Member of Technical Staff. Since 2005, he has been a full-time Faculty with Tsinghua University. He received Tsinghua University Outstanding Graduate Award and Medal in 1988. He is the General Co-Chair of Conference on Cryptographic Hardware and Embedded Systems (CHES), in 2021.

Xiangmin Zhang was born in Beijing, China, in 1966. He received the B.S. degree in microelectronics from Peking University, Beijing, in 1988, and the M.S. degree in electronic engineering from Tsinghua University, Beijing, in 1991. Since 1991, he has been joined the Institute of Microelectronics, Tsinghua University. His main research interests are in information security and automotive electronics.