Received April 27, 2019, accepted May 8, 2019, date of publication May 15, 2019, date of current version May 28, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916553

Adaptive Chosen-Plaintext Collision Attack on Masked AES in Edge Computing

YAOLING DING[1], YING SHI[2], AN WANG[2,3], XUEXIN ZHENG[4], ZONGYUE WANG[5], AND GUOSHUANG ZHANG[6]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[3]School of Computer Science, Beijing Institute of Technology, Beijing 100081, China
[4]China Academy of Electronics and Information Technology, Beijing 100041, China
[5]Open Security Research, Shenzhen 518063, China
[6]Science and Technology on Information Assurance Laboratory, Beijing 100072, China

Corresponding authors: Ying Shi (shiying@iie.ac.cn) and An Wang (wanganl@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872040 and Grant U1836101, in part by the National Cryptography Development Fund under Grant MMJJ20170201, and in part by the Foundation of Science and Technology on Information Assurance Laboratory under Grant KJ-17-009.

ABSTRACT Edge computing handles delay-sensitive data and provides real-time feedback, while it brings data security issues to edge devices (such as IoT devices and edge servers). Side-channel attacks main threaten to these devices. Collision attack represents a powerful category of side-channel analysis in extracting security information from embedded cryptographic algorithms. Since its proposition in 2003, plenty of collision detection algorithms are presented, most of which enumerate all the values of target plaintext byte to find a collision. In this paper, we establish a relation between ''Euclidean distance between traces'' and ''Hamming distance between values,'' and take advantage of the distance information leaked from the power traces of encrypting an adaptively chosen plaintext to reduce the candidate plaintext space. Consequently, the collision is detected at a high pace. Moreover, this improvement is fault-tolerant, and its self-correction feature promotes the efficiency of attacks based on our method significantly. We take AES implemented with masks, which is usually employed in edge computing devices, for instance, to introduce our method and conduct experiments to verify its efficiency. According to the experimental results, for whole key recovery attacks, our method requires only 26.5% plaintexts, 32.2% traces, and much less than 10% computations of the collision-correlation attack launched by Clavier *et al.*

INDEX TERMS Adaptive chosen-plaintext collision attack, edge computing, masking, the least square method.

I. INTRODUCTION

Edge computing performs real-time processing of data in distributed devices, such as IoT devices and edge servers, in order to take full advantage of proximity to the physical items (sensors or users) and reduce the reaction time. While, it brings information security issues to these devices. Increasing attention has been paid to encryption [1]–[4], authentication [5]–[8] and privacy preservation [9], [10] in edge computing devices. Since most of these devices are accessible to attackers, side-channel attacks are applicable to the cryptographic algorithms embedded on them. Figure 1 shows the framework of side-channel attacks on edge computing devices. Therefore, it is necessary to evaluate their security with side-channel analysis.

Since Kocher [11] proposed timing attacks in 1996, cryptanalysts have considered various kinds of side-channel information and presented plenty techniques such as differential power analysis [12], collision attack [13], correlation power analysis [14], template attack [15], fault sensitivity analysis [16]–[18], etc. In this paper, we focus on improving the efficiency of collision attack when applied to masked AES in edge computing. Collision attack was firstly proposed by Schramm *et al.* [13] in 2003. Subsequently, a similar attack on AES was proposed to detect collisions in the first MixColumn operation [19]. In 2007, Bogdanov proposed

The associate editor coordinating the review of this manuscript and approving it for publication was Chien-Ming Chen.

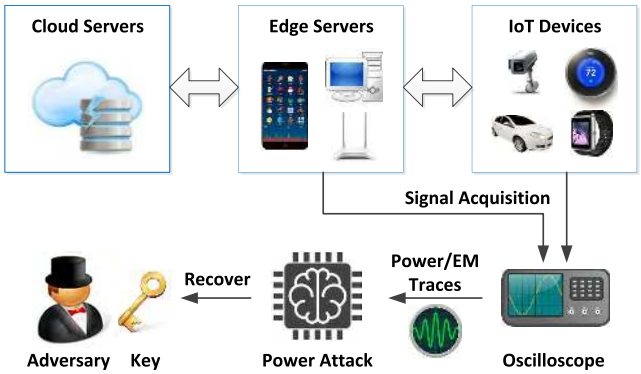


FIGURE 1. Side-channel attacks on edge computing.

linear collision attack on AES in [20], and presented some improvements of collision attacks in [21], [22]. In 2012, Gérard and Standaert [23] showed unified and optimized linear collision attacks based on the low density parity check code. In 2015, Ren *et al.* [24] proposed double sieve collision attack basing on bitwise collision detection. All the methods mentioned above aim at primitives implemented without any countermeasures.

While, in practical applications, various countermeasures are employed to protect the cryptographic devices from side-channel attacks. Masking is regarded as an effective one against first-order power analysis. At CHES 2010, Moradi *et al.* [25] presented a collision-correlation attack against masking. Afterwards, Clavier *et al.* [26] improved it and reduced the number of required plaintexts from 256 to 27.5 at CHES 2011. In 2015, Wang *et al.* [27] mounted collision attacks on masked AES implemented on a smart card. Right or wrong collision rate was used as a distinguisher to detect collision by Wang *et al.* [28] in 2018. So far, most existing collision detection algorithms detect collisions by searching the values of candidate plaintext byte exhaustively, which have a lot improving space in our view.

For collision attacks, statistical models, such as the least square method [13] and correlation coefficient [25], are employed to evaluate the simplicity of two plaintext bytes. Collisions are detected when the corresponding value reaches minimum or maximum. However, plenty of non-extreme values are ignored, which contain a lot information about the distance between the current plaintext byte and the target one. Therefore, two questions cause our interests:

How to measure the distance between the current plaintext byte and the target one?

How many steps it takes for the current plaintext byte "jumping" to the target one?

A. OUR CONTRIBUTION

In this paper, we explore the two questions above and propose an adaptive chosen-plaintext collision attack on the first-order masked AES in which 7.29 plaintexts on average are required to find all the 15 collisions among 16 inputs of S-boxes. An additional cost is precomputation of building nine templates. Contributions of this paper are as follows:

- By an appropriate model, we extract information of the Hamming distance between the current plaintext byte and the target one. With this information, we reduce the candidate plaintext space in a loop and detect a collision at a high pace. Experimental results show that our method requires 2.03% plaintexts and 5.44% traces of collision-correlation attack [26] to detect a collision between two inputs of S-boxes.
- Our method has a self-correction feature which promotes the efficiency further. Experimental results show that attacks with self-correction reduce the number of required traces by 54.7%.
- For the whole key recovery attacks on masked AES, collisions among 16 inputs of S-boxes are detected in parallel, and 7.29 plaintexts on average are required. According to simulation experiments, our method requires 26.5% plaintexts and 32.2% traces of collision-correlation attack [26] respectively, and the computation complexity of the former is much less than 10% of the latter.

B. ORGANIZATION

The remainder of this paper is organized as follows. In Section II, we give a brief description of the classic collision attack and collision-correlation attack on AES. Section III introduces our method based on the least square method, and discusses its correctness. In Section IV, we conduct experiments and compare the efficiency of our method with two classic methods. Some other assessing models such as central moment product etc. are studied as well. Section 5 addresses the self-correction feature of our method which improves its efficiency further. We launch a whole key recovery attack on masked AES-128 in Section VI. Section VII concludes this paper.

II. PRELIMINARY

In edge computing, cryptographic algorithms are usually implemented by software. Therefore, we consider software implementations of masked AES in this paper and assume that the power consumptions are based on Hamming weight model [15]. Note that for some hardware implementations and equipments based on Hamming distance model [14], our method can also be adjusted to work.

A. NOTATION

Taking AES-128 for instance, we denote the 128-bit plaintext and the whitening key by $P = p_1\|p_2\|...\|p_{16}$ and $K = k_1\|k_2\|...\|k_{16}$ respectively. Let $S_1, S_2, ..., S_{16}$ stand for the 16 S-box operations in the first round and denote their inputs by $x_i = p_i \oplus k_i$ $(1 \le i \le 16)$.

In order to reduce the influence of noises, we encrypt a certain plaintext several times in a certain attack. The power trace acquired during the j-th encryption is denoted by $T^{(j)}$. Each trace includes 16 segments that correspond to the 16 S-boxes

in the first round. We denote these segments by $T_i^{(j)}$ ($1 \leq i \leq 16$). $T_i^{(j)}$ contains several points that leak information of the operands and have similar x-coordinates to the other segments. With some precomputation, l x-coordinates are selected and aligned. Then, $T_i^{(j)}$ is transformed to an ordered set of y-coordinates corresponding to the l x-coordinates, denoted as $T_i^{(j)} = \{t_{i,1}^{(j)}, t_{i,2}^{(j)}, ..., t_{i,l}^{(j)}\}$. Based on the Hamming weight model [15], we have

$$t_{i,q}^{(j)} = s_{i,q}\mathrm{HW}(x_i^{(j)}) + r_{i,q}, \tag{1}$$

where $s_{i,q}$ is a constant, and $r_{i,q} \sim \mathcal{N}(\mu_q, \sigma_q)$, which means that $t_{i,q}^{(j)}$ follows the normal distribution $\mathcal{N}(s_{i,q}\mathrm{HW}(x_i^{(j)}) + \mu_q, \sigma_q)$.

B. COLLISION ATTACK

In 2003, Schramm *et al.* [13] presented the basic concept of collision attack, which takes advantage of identical intermediate values during the encryption of a certain plaintext to detect the linear relation among key bytes and then recover the whole key. For example, if the inputs of S_1 and S_2 are equal to each other, i.e. $p_1 \oplus k_1 = p_2 \oplus k_2$, we can deduce that $k_2 = k_1 \oplus p_1 \oplus p_2$. Thus, the candidate key space is reduced by 2^8. When all the equations between k_1 and the other key bytes are established, the space of candidate keys is reduced to 2^8 and the whole key can be recovered by searching all the values of k_1. During a collision attack, the adversary usually enumerates the values of a plaintext byte thoroughly to obtain a collision, as shown in Figure 4 (left).

Power consumptions of the operations on two identical intermediate values are supposed to be the same. Therefore, the similarity between the power segments corresponding to two intermediate values is usually used to detect collisions. In practice, power traces are acquired as ordered point sets. Denote the two segments (after averaged) as $\{t_{1,1}, t_{1,2}, ..., t_{1,l}\}$ and $\{t_{2,1}, t_{2,2}, ..., t_{2,l}\}$. Their similarity is estimated by calculating the least absolute deviation (LAD) or the least square (LSM) [29] between the two sets, which are

$$LAD(T_1, T_2) = \sum_{q=1}^{l} |t_{1,q} - t_{2,q}|, \tag{2}$$

$$LSM(T_1, T_2) = \sum_{q=1}^{l} (t_{1,q} - t_{2,q})^2. \tag{3}$$

If $LAD(T_1, T_2)$ (or $LSM(T_1, T_2)$) is less than a threshold, a collision is detected.

C. MASKING SCHEME

Cryptographic algorithms employed in edge computing or any other applications are usually implemented with countermeasures in order to defend from side-channel attacks. Masking is a widely used countermeasure. In this approach, we consider the implementation of AES that employs a masked substitution table (S-box) as proposed by Akkar and Giraud [30]. This masked S-box S' is defined as $S'(x_i \oplus m) = S(x_i) \oplus m'$, with m (resp. m') being the mask of the input byte x_i (resp. output byte $S(x_i)$). Masks are generated randomly in each encryption, so is the masked substitution table. We adopt the masking scheme used in [26], which assumes that the masks involved in all the 16 S-boxes are equal.

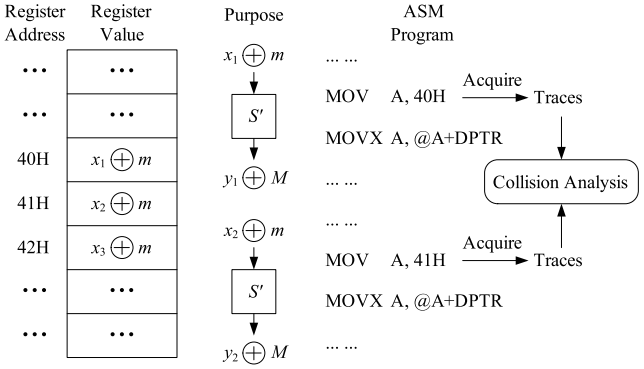


FIGURE 2. 8051 assembly program for look-up table operation (S-box operation) and collision detection.

The look-up table operation (S-box operation) is implemented by the 8051 assembly program as shown in Figure 2. The two "MOV" instructions, whose operands are $x_1 \oplus m$ and $x_2 \oplus m$, are usually used to detect collisions.

D. COLLISION-CORRELATION ATTACK

In 2011, Clavier *et al.* [26] presented a collision-correlation attack on masked S-box of AES. They employed Pearson correlation coefficient, which is

$$\rho_{p_1,p_2,q} = \frac{\mathrm{Cov}(t_{1,q}, t_{2,q})}{\sigma_{t_{1,q}}\sigma_{t_{2,q}}} = \frac{n\sum(t_{1,q}^{(j)}t_{2,q}^{(j)}) - \sum t_{1,q}^{(j)} \sum t_{2,q}^{(j)}}{\sqrt{n\sum(t_{1,q}^{(j)})^2 - (\sum t_{1,q}^{(j)})^2}\sqrt{n\sum(t_{2,q}^{(j)})^2 - (\sum t_{2,q}^{(j)})^2}}, \tag{4}$$

to evaluate the similarity of two segments T_1 and T_2 (corresponding to $x_1 \oplus m$ and $x_2 \oplus m$ respectively) in one trace. Figure 3 describes the collision detection procedure. With plaintext byte p_1 fixed to a constant, p_2 traverses 0-255. For a certain value of p_2, encrypt the plaintext repeatedly until n power traces (i.e. $2n$ segments) are acquired. Assume that each segment contains l points. For each point, calculate the correlation coefficient $\rho_{p_1,p_2,q}(q = 1, 2, ..., l)$ between T_1 and T_2. Sieve the maximal $\rho_{p_1,p_2,max}$ for each value of p_2. If $\rho_{p_1,p_2,max}$ is larger than a threshold, its corresponding value of p_2 may lead to a collision.

For AES-128, there are $C_{16}^2 = 120$ combinations of two key bytes. Thus, for a certain plaintext, collision detections of 120 key byte pairs are executed. All the 15 collisions among 16 inputs of S-boxes are found with 27.5 plaintexts on average.

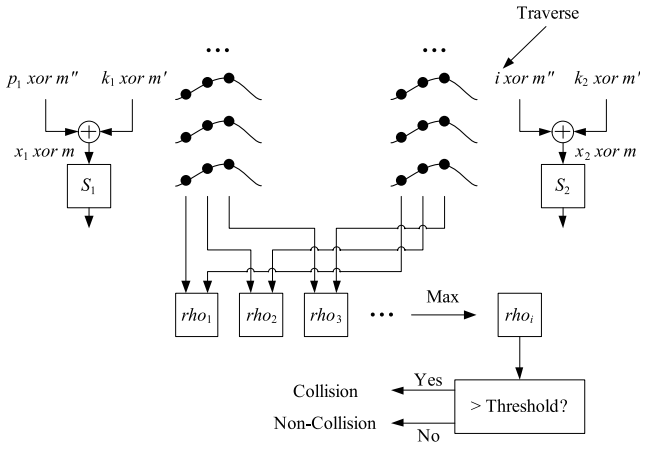


FIGURE 3. Collision-correlation attack on masked AES.

III. ADAPTIVE CHOSEN-PLAINTEXT COLLISION ATTACK

In the last few years, various methods are employed in collision attacks to evaluate the similarity between two sets of segments, such as the least square method, correlation coefficient, mutual information [31] and so on. Intuitively speaking, they quantify the "*distance*" between two segments. For example, the value obtained by the least square method is usually called *Euclidean distance*.

A. BASIC IDEA

Interestingly, according to our observation, this "*distance*" also imply the Hamming distance between x_1 and x_2, which can be deduced to the Hamming distance between the current value of p_2 and the target one, i.e. $p_1 \oplus k_1 \oplus k_2$. Taking the least square method for example, a larger value means a further distance (i.e. lower similarity) between the current value of p_2 and the target one, and vice versa. Therefore, we present a method intending to relate the "*distance*" to Hamming distance by templates and use it to find the target plaintext byte at a high pace.

Denote the target value of p_2 as β_2 when p_1 is fixed to a constant β_1. The outline of our adaptive chosen-plaintext collision attack is as follows:

1) Build templates of Hamming distances for the target devices (in edge computing);
2) Fix p_1 to a constant, initial the candidate value space of p_2 with 0-255;
3) Estimate the Hamming distance between x_1 and x_2 (equal to $HD(p_2, \beta_2)$) with templates;
4) Reduce the candidate value space of p_2 according to the Hamming distance;
5) Repeat the process above until there remains one candidate, which is β_2.

Figure 4 (right) describes this procedure with flow chart.

B. ATTACK SCENARIO

Our adaptive chosen-plaintext collision attack consists of two stages, namely building template stage and online acquisition stage. Algorithm 1 gives the pseudo code of the attack on k_1 and k_2.

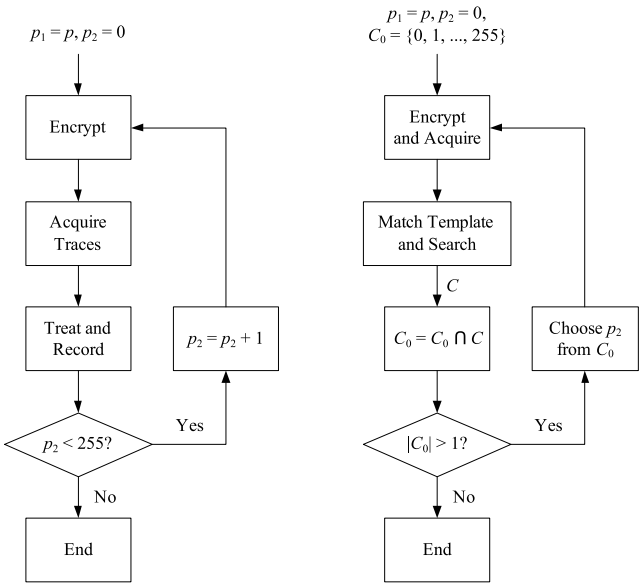


FIGURE 4. Flow chart of classic (left) and our (right) collision attacks on AES. $|C_0|$ stands for the number of elements in set C_0.

In the building template stage, set $p_1 = k_1 = k_2 = 0$. Let p_2 traverse $\{0, 1, 3, 7, ..., 255\}$ such that the Hamming distance h exhausts the values in $\{0, 1, ..., 8\}$. For each value of p_2:

1) Encrypt the plaintext with masked AES for n_τ times and collect the power traces $\{T^{(j)} \mid j = 1, 2, ..., n_\tau\}$ of each encryption. n_τ should be as large as possible in order to decrease the noise to the minimum. We denote the acquisition procedure by **AcquireTrace()**.
2) Assume that the segments corresponding to S_1 and S_2 have been truncated and aligned. Extract l pairs of points $\{(t_{1,q}^{(j)}, t_{2,q}^{(j)}) \mid q = 1, 2, ..., l\}$ from the two segments of each trace $T^{(j)}$ $(j = 1, 2, ..., n_\tau)$. We denote the extraction procedure by **ExtractPoints()**.
3) The least square method (LSM) is adopted to measure the distance $D^{(j)}$ between segments. Average of all $D^{(j)}$ $(j = 1, 2, ..., n_\tau)$ is calculated by **Average()**, which is

$$D_{\mathrm{LSM}} = \frac{1}{n} \sum_{j=1}^{n} \sum_{q=1}^{l} (t_{1,q}^{(j)} - t_{2,q}^{(j)})^2, \tag{5}$$

where D_{LSM} is the value of the template τ_h corresponding to Hamming distance $h = \mathrm{HD}(p_1, p_2)$.

In this stage, nine templates are built in total. Note that for a certain device, this stage is executed only once.

In the online acquisition stage, set $p_1 = 0$ and initialize the candidate value space of p_2 with 0-255, denoted as $C_0 = \{0, 1, 2, ..., 255\}$. Our method works on candidate value space by reducing its size in a loop, instead of searching it exhaustively. In each loop,

1) Chose a value for p_2 randomly from C_0, denoted as **ChooseRandomly()**.

Algorithm 1 Adaptive Chosen-Plaintext Collision Attack on Masked AES

Building template stage:

$p_1 := 0; k_1 := 0; k_2 := 0;$
for $h := 0, 1, 2, ..., 8$ **do**
  $p_2 := 2^h - 1;$ // such that $\mathrm{HD}(x_1, x_2) = h$
  for $j := 1, 2, ..., n'$ **do**
    $T^{(j)} :=$ **AcquireTrace**(p_1, k_1, p_2, k_2, m);
    $\{(t_{1,q}^{(j)}, t_{2,q}^{(j)}) | q := 1, 2, ..., l\} :=$**ExtractPoints**$(T^{(j)})$;
    $D^{(j)} := \sum_{q=1}^{l} (t_{1,q}^{(j)} - t_{2,q}^{(j)})^2;$
  end for
  $\tau_h :=$ **Average**$(\{D^{(j)} \mid j = 1, 2, ..., n'\})$;
end for

Online acquisition stage:

//k_1 and k_2 are fixed in the edge computing device

$C_0 := \{0, 1, 2, ..., 255\}; p_1 := 0;$
while $|C_0| > 1$ **do**
  $p_2 :=$ **ChooseRandomly**(C_0);
  for $j := 1, 2, ..., n$ **do**
    $T^{(j)} :=$ **AcquireTrace**(p_1, k_1, p_2, k_2, m);
    $\{(t_{1,q}^{(j)}, t_{2,q}^{(j)}) | q := 1, 2, ..., l\} :=$**ExtractPoints**$(T^{(j)})$;
    $D^{(j)} := \sum_{q=1}^{l} (t_{1,q}^{(j)} - t_{2,q}^{(j)})^2;$
  end for
  $D :=$ **Average**$(\{D^{(j)} | j := 1, 2, ..., n\})$;
  $h' :=$ **MatchTemplate**$(D, \{\tau_h | h = 0, 1, 2, ..., 8\})$;
  $C := \{p | \mathrm{HD}(p, p_2) = h'\};$
  $C_0 := C_0 \cap C;$
end while
if $|C_0| = 1$ **then**
  return C_0;
else
  return error;
end if

2) Encrypt the plaintext with masked AES for n times and collect the power traces $\{T^{(j)} \mid j = 1, 2, ..., n\}$ of each encryption with **AcquireTrace()**.
3) Extract l pairs of points $\{(t_{1,q}^{(j)}, t_{2,q}^{(j)}) \mid q = 1, 2, ..., l, j = 1, 2, ..., n\}$ from the two segments of S_1 and S_2 in each trace.
4) Match the templates obtained in building template stage with **MatchTemplate()**, which returns the value

$$H = \underset{h \in \{0,1,2,...,8\}}{\arg\min} |D - \tau_h|. \tag{6}$$

H is the estimated Hamming distance between x_1 and x_2, i.e. $HD(p_2, \beta_2)$.
5) Build a set $C = \{p \mid \mathrm{HD}(p, p_2) = H\}$ and update C_0 with $C_0 \cap C$.
6) Repeat the operations above until the size of C_0 is reduced to 0 (return an error) or 1.

The correctness of Algorithm 1 depends on the fact that the Hamming distance of the two intermediate values $\mathrm{HD}(x_1, x_2)$ and the "*distance*" of their power traces are bijective, and the differences among these "*distance*" are non-ignorable. We give Theorem 2 and 5 in Appendix to explain the two issues.

Remark 1: To simplify the attack, we average the power traces to build and match templates in the two stages respectively. While, since $\{t_{1,q}^{(j)} - t_{2,q}^{(j)} | q = 1, 2, 3, ..., l\}$ *follows multivariate normal distribution, it is more preferable to add a covariance matrix* C_h *in the template. Then, the template is changed to*

$$\tau_h = (\mu_h, C_h), \tag{7}$$

where μ_h *denotes the mean value. Correspondingly,* ***MatchTemplate()*** *is turned to*

$$H = \underset{h \in \{0,1,2,...,8\}}{\arg\max} \frac{exp(-\frac{1}{2} \cdot (D - \mu_h)' \cdot C_h^{-1} (D - \mu_h))}{\sqrt{(2\pi)^l \cdot det(C_h)}}. \tag{8}$$

For more details, we refer to the classic template attack [15].

Remark 2: Our method is based on a potential assumption that the adversary owns a device that is identical to the target one and the main key is fully under control. Even if the adversary cannot set the key, the templates can still be built in the following ways. Let $p_1 = 0$ *and* p_2 *traverse 0-255. For each value of* p_2*, execute 4-9 of the building template stage (the difference is that* k_1, k_2 *are unknown). Then, 256 values are obtained, which can be classified into 9 groups. Average all the values of each group, nine templates are built.*

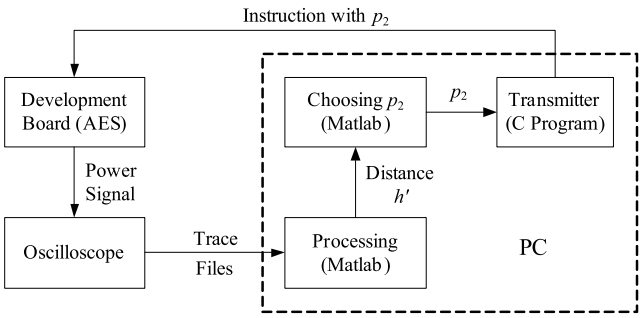


FIGURE 5. The experiment environment of our attack.

IV. EXPERIMENTS AND EFFICIENCY

A. EXPERIMENTAL CONFIGURATION

We implement AES-128 with MCS-51 assembly codes on AT89S52 processor of MathMagic side-channel analyzer to simulate the edge computing devices. The power consumption of each encryption is acquired accurately by PicoScope 3403D Oscilloscope with sampling rate 1GSa/s. Instructions are sent to the AT89S52 board by C program from PC, and the power traces are processed by Matlab program. We integrate Matlab and C code together, and build a semi-automatic system, which is shown in Figure 5. Attack are mounted by executing the program of this system several times.

B. ATTACK EXPERIMENTS

In this section, we focus on two segments of a power trace, which correspond to two "MOV" instructions whose operands are $x_1 \oplus m$ and $x_2 \oplus m$ respectively. $l = 50$ aligned points are selected from each segment.

In building template stage, we collected $n_\tau = 10000$ traces to compute τ_h for each h. Figure 6 shows the nine templates, namely 0.0987, 0.1005, 0.1022, 0.1044, 0.1065, 0.1087, 0.1107, 0.1126, and 0.1149, which correspond to the nine Hamming distances between x_1 and x_2 respectively.

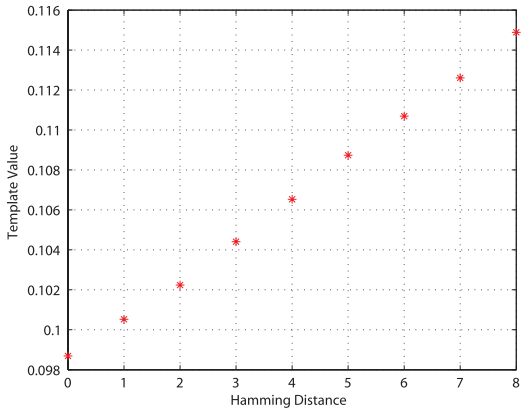


FIGURE 6. Templates corresponding to the nine Hamming distances.

TABLE 1. The intermediate data during our experiment.

Num. of loop	1	2	3	4	5
p_2	0	77	170	150	147
D	0.1059	0.1107	0.1061	0.1019	0.0990
HD(p_2, β_2)	4	6	4	2	0
Num. of candidates	70	16	9	4	1

In online acquisition stage, we set $k_1 = 0, k_2 = 147$, $p_1 = 0$. $n = 1000$ traces are acquired in each loop. In the first loop, $p_2 = 0$ is chosen. Based on equation 3, we get $D = 0.1059$, which implies HD$(p_2, \beta_2) = 4$. Therefore, the number of candidates in C_0 is reduced to 70. Similarly, in the second, third, fourth, and fifth loop, p_2 is randomly chosen from C_0, and the corresponding D and HD(p_2, β_2) are obtained. We list them in Table 1.

Figure 7 shows the five distance values with red stars, and the nine templates with dotted lines. Since only one candidate $p_2 = 147$ is survived after the fifth loop, we finally get the equation $k_1 \oplus k_2 = p_1 \oplus \beta_2 = 147$. The survived candidates in each loop are shown in Figure 8.

Furthermore, Figure 9 shows the relation between the number of traces and the distance D in each loop (i.e. for each chosen value of p_2). For our simulation environment, 800 traces are sufficient to identify D in attacks on masking model.

C. ATTACKS WITH SOME OTHER MODELS

We describe our method by adopting LSM to measure the "*distance*" between segments and estimate the

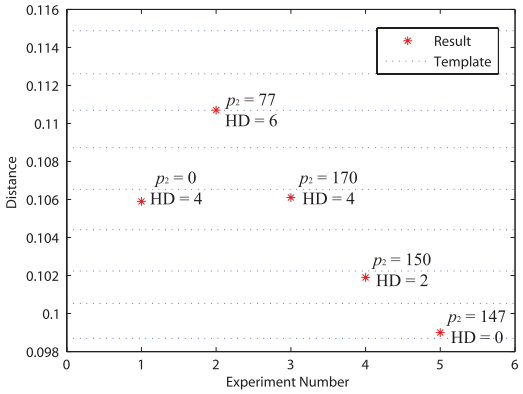


FIGURE 7. Template matching during an attack that uses five plaintexts.

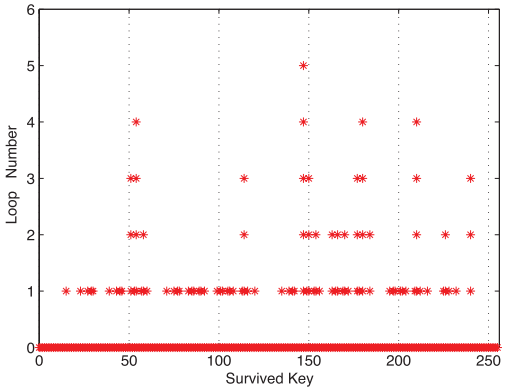


FIGURE 8. Survived candidates after each loop in our attack.

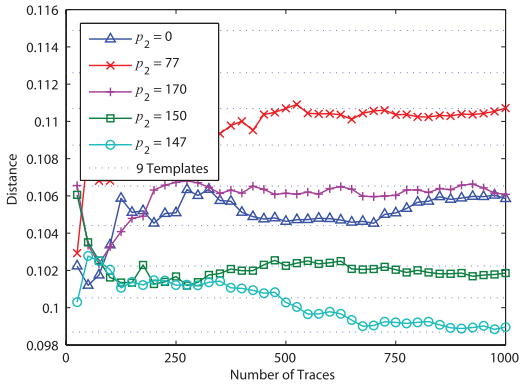


FIGURE 9. The relation between the number of traces and D corresponding to the five candidates of p_2 in our attack.

corresponding Hamming distance in Section III. Actually, various models can be used to achieve the same purpose. We study some of them, in order to obtain the most efficient one.

Templates based on the least absolute deviation (LAD) and the least higher exponent method (denoted by LAD$^\alpha$) are defined as

$$D_{\mathrm{LAD}} = \frac{1}{n}\sum_{j=1}^{n}\sum_{q=1}^{l}|t_{1,q}^{(j)} - t_{2,q}^{(j)}|, \tag{9}$$

$$D_{\mathrm{LAD}^{\alpha}} = \frac{1}{n}\sum_{j=1}^{n}\sum_{q=1}^{l}|t_{1,q}^{(j)} - t_{2,q}^{(j)}|^{\alpha}. \tag{10}$$

And templates based on central moment product method (CMP) [32], [33] is defined as

$$D_{\mathrm{CMP}} = \frac{1}{n}\sum_{j=1}^{n}\sum_{q=1}^{l}(t_{1,q}^{(j)} - \bar{t}_{1,q})(t_{2,q}^{(j)} - \bar{t}_{2,q}), \tag{11}$$

where $\bar{t}_{1,q}$ stands for the mean value of the n points $t_{1,q}^{(j)}(j = 1, 2, ..., n)$.

Building template stage is modified with the above models, namely LAD, LAD^4, LAD^8 and CMP, respectively. With the same 10000 traces acquired in the last subsection, new templates are established, as shown in Figure 10-13.

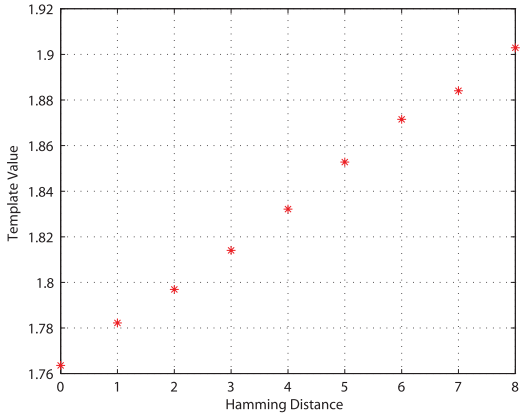


FIGURE 10. The templates based on LAD model.

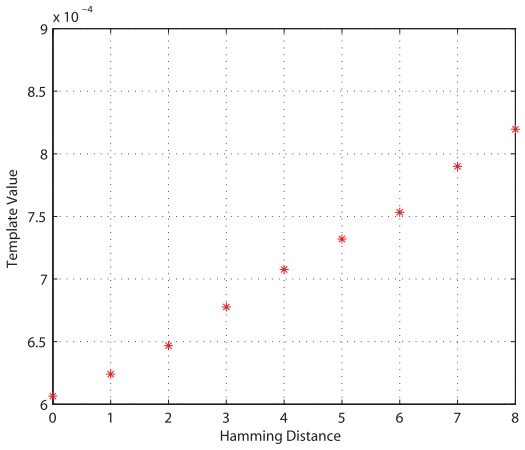


FIGURE 11. The templates based on LAD^4 model.

From these figures, we know that LAD, LAD^4 and CMP distinguish Hamming distances clearly, while LAD^8 is available only when the noise is reduced to a certain extent. Note that the templates based on CMP detect a collision according to its maximum, while the other models detect collisions according to their minimums. We discuss the efficiency of these models in the following subsection.

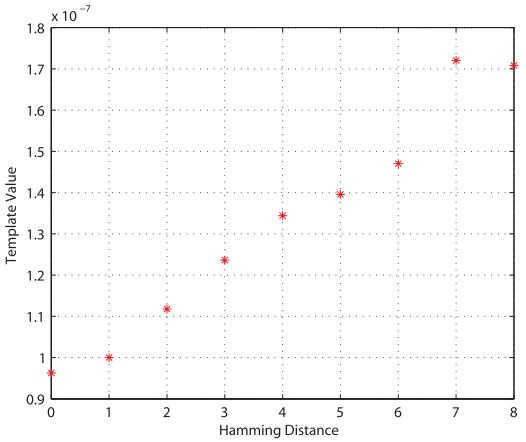


FIGURE 12. The templates based on LAD^8 model.

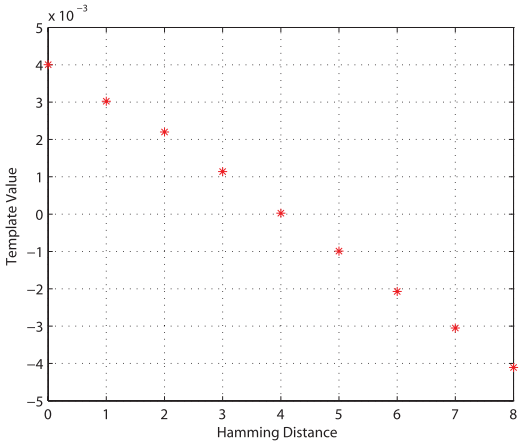


FIGURE 13. The templates based on CMP model.

D. EFFICIENCY COMPARISONS OF ATTACKS ON TWO KEY BYTES

For the first loop of online acquisition stage, the value of p_2 is chosen randomly, so the probability of $HD(p_2, \beta_2) = h$ ($h \in \{0, 1, ..., 8\}$) is $C_8^h/256$. Since there are C_8^h candidate values that have h bits different from the current p_2, the number of candidate values in C_0 is reduced to $C_8^h/256 \times C_8^h$. Thus, after the first loop, the average number of remained candidates in C_0 is

$$\frac{C_8^{0^2} + C_8^{1^2} + C_8^{2^2} + ... + C_8^{8^2}}{256} \approx 50.3.$$

We simulate the whole reduction process in Matlab and find that it takes 5.2 loops on average to obtain the target β_2. That is to say, only 5.2 plaintexts are required in our attack, which is much less than 256 in the classic collision attack. The ratio is nearly 2%.

To verify the efficiency of our method further, we conduct simulation experiments in Matlab. We compare the success rate of attacks based on the five models mentioned above with two classic methods, namely second-order template

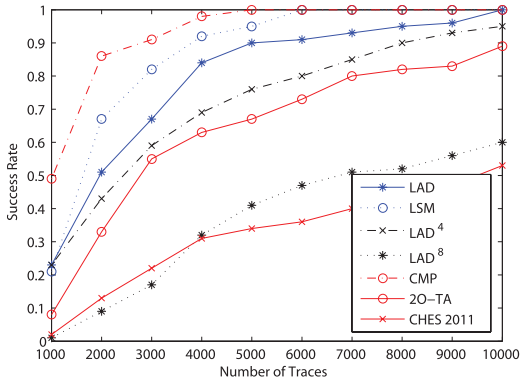


FIGURE 14. The comparison of success rates among the seven attacks with different number of traces ($\sigma = 3, l = 50$).

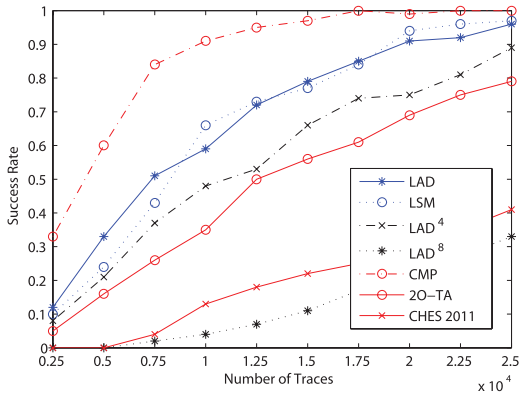


FIGURE 15. The comparison of success rates among the seven attacks with different number of traces ($\sigma = 5, l = 50$).

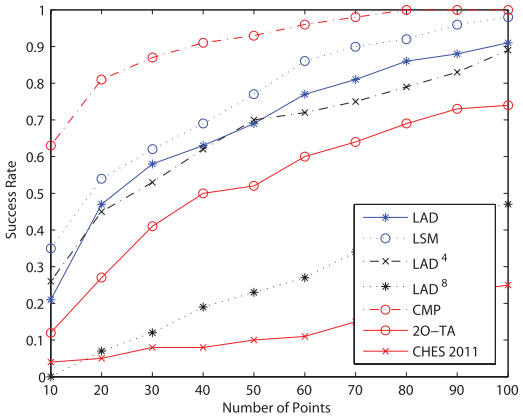


FIGURE 16. The comparison of success rates among the seven attacks with different number of sample points ($\sigma = 3, n = 4000$).

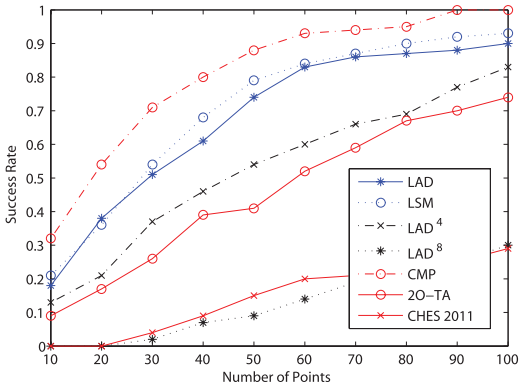


FIGURE 17. The comparison of success rates among the seven attacks with different number of sample points ($\sigma = 5, n = 10000$).

attack [34] and collision-correlation attack [26]. **AcquireTrace()** and **ExtractPoints()** are replaced with a simulation program, which generates 50 points for each segment with regard to the normal distribution $\mathcal{N}(\mathrm{HW}(p_i \oplus k_i \oplus m), 3^2)$. Repeating Algorithm 1 for 100 times with the five models respectively, we obtain their success rates. Collision-correlation attack and second-order template attack are mounted on the same group of simulated power traces respectively, and the corresponding success rates are obtained. Note that for second-order template attack, the experiment is simplified to recover one key byte (two corresponding positions, the masked value and the mask value). Figure 14 shows the comparison of success rates among the seven attacks. Experiments with $\sigma = 5$ (the standard deviation of the noise) are also carried out, and the results are displayed in Figure 15.

From the figures we conclude that our attacks based on CMP, LSM, LAD and LAD^4 show much higher efficiency than second-order template attack and collision-correlation attack. CMP outperforms the other methods with the highest success rate. For $\sigma = 3$, attacks based on the CMP model require about 3900 ($3900/5.2 = 750$ per plaintext) traces to achieve the success rate 95%, while the required number of collision-correlation attack and second-order template attack are about 71680 ($71680/256 = 280$ per plaintext) and 21000 respectively. The ratios between our method and the two classic ones are 5.44% and 18.57%.

Besides, we study the relation between success rate and the number of sample points (i.e. l). Fixing the number of traces to 4000 for $\sigma = 3$ and 10000 for $\sigma = 5$ respectively, we carry out simulation experiments with l ranging from 10 to 100. Figure 16 and Figure 17 show the experimental results. Apparently, with the number of sample points increasing, the success rates of the distance-based models (our method) increase much faster than that of the correlation-coefficient-based model.

V. SELF-CORRECTION FEATURE

We have an interesting observation from the experiments based on our adaptive chosen-plaintext collision attack, which is

Observation 1: In the online acquisition stage, the Hamming distance of p_2 and β_2 is always even after the first loop of a success attack.

We give the following lemma to explain this observation.

Lemma 1: For any $A, B \in \{0, 1, 2, ..., 255\}$ such that $\mathrm{HW}(A) = \mathrm{HW}(B)$, $\mathrm{HW}(A \oplus B)$ is even.

Proof: Denote $A = a_1\|a_2\|...\|a_8$ and $B = b_1\|b_2\|...\|b_8$, we have

$$\mathrm{HW}(A \oplus B) = (a_1 \oplus b_1) + (a_2 \oplus b_2) + ... + (a_8 \oplus b_8).$$

If $a_i = b_i$ ($i \in \{1, 2, ..., 8\}$), $a_i \oplus b_i = 0$, otherwise $a_i \oplus b_i = 1$. Assume $\mathrm{HW}(A) = \mathrm{HW}(B) = w$, then there are w pairs of (a_i, b_i) such that $a_i \oplus b_i = 1$. Denote these pairs as (a_{j_1}, b_{j_1}), (a_{j_2}, b_{j_2}),..., (a_{j_w}, b_{j_w}).

Since $a_i, b_i \in \{0, 1\}$, we have $b_{j_l} = 1 - a_{j_l}$ ($l \in \{1, 2, ..., w\}$). Then,

$$\begin{aligned} \mathrm{HW}(A) &= \mathrm{HW}(B) \\ a_1 + a_2 + ... + a_8 &= b_1 + b_2 + ... + b_8 \\ a_{j_1} + a_{j_2} + .. + a_{j_w} &= b_{j_1} + b_{j_2} + ... + b_{j_w} \\ a_{j_1} + a_{j_2} + .. + a_{j_w} &= (1 - a_{j_1}) + (1 - a_{j_2}) + .. + (1 - a_{j_w}) \\ w &= 2(a_{j_1} + a_{j_2} + .. + a_{j_w}). \end{aligned}$$

Since $a_{j_1} + a_{j_2} + .. + a_{j_w}$ is an integer, w is even. □

Basing on Lemma 1, we give the explanation of Observation 1 as follows.

Denote the randomly chosen value of p_2 in the i-th loop as p_2^i, and the Hamming distance of p_2^i and β_2 as H^i ($i = 1, 2, 3, ...$).

We know that, at the end of the i-th loop, C_0 is reduced to a set in which all the items have H^i bits different from p_2^i. Since p_2^{i+1} is chosen from C_0, we have

$$\mathrm{HW}(p_2^{i+1} \oplus p_2^i) = H^i = \mathrm{HW}(p_2^i \oplus \beta_2).$$

Therefore, $H^{i+1} = \mathrm{HW}(p_2^{i+1} \oplus \beta_2) = \mathrm{HW}(p_2^{i+1} \oplus p_2^i \oplus p_2^i \oplus \beta_2)$ is even.

Basing on Observation 1, we add a self-correction mechanism to our method and promote its efficiency further: **If H^i ($i > 1$) is odd, re-execute the i-th loop to revise this error.** While, there is an issue that if an error occurs in the first loop, H obtained in the subsequent loop is always odd. We set a counter to detect this error: **If a certain loop is re-executed more than a threshold such as three, return failure directly.** In practice, attacks usually require a success rate over 90%, so errors occur with very low probability. Thus, the self-correction mechanism significantly decreases the number of required traces with negligible extra computation and no extra plaintext.

We carry out simulation experiments to verify the improvements brought by the self-correction mechanism. Attacks based on six methods are launched on masked AES, namely LSM with and without self-correction (SC_LSM and LSM), CMP with and without self-correction (SC_CMP and CMP), second-order template attack, and collision-correlation attack. Figure 18 and 19 shows the comparison of the success rates among them with $\sigma = 3$ and 5 respectively. From the figures, we get that our attacks with self-correction outperform the ones without it. For example, SC_CMP requires 1768 traces to achieve a success rate of 95%, which is 54.7% less that of CMP.

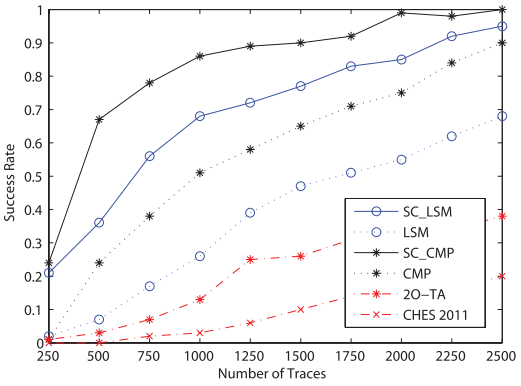


FIGURE 18. The comparison of success rates among four attacks based our method with/without self-correction and two classic methods ($\sigma = 3, l = 50$).

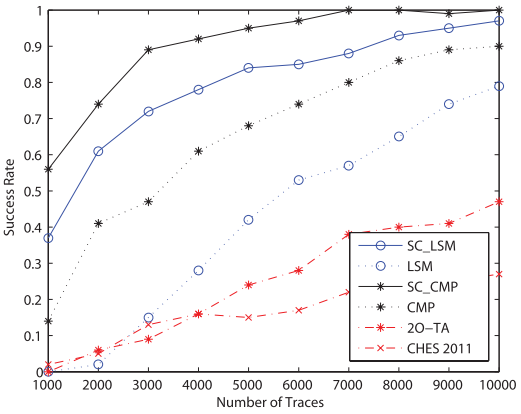


FIGURE 19. The comparison of success rates among four attacks based our method with/without self-correction and two classic methods ($\sigma = 5, l = 50$).

Remark 3: If an error occurs in the first loop, the attack can still continue by re-executing the first loop with the H corresponding to the second minimal "distance". Thereby, the attack is more error-tolerant.

VI. ATTACKS ON 16 S-BOXES

In this section, we discuss the efficiency of the whole key recovery attack on AES based on our method.

A. ATTACK SCENARIO

The building template stage is similar to that of two S-boxes, so we omit it here. In the online acquisition stage, fix p_1 to a constant, and initialize C_i for p_i ($i = 2, 3, ..., 16$) with $\{0, 1, ..., 255\}$ respectively. Denote the target value of p_i as β_i ($i = 2, 3, ..., 16$). Execute the following steps until the termination criterion is reached.

1) Randomly chose the value of p_i ($i = 2, 3, ..., 16$) from C_i;
2) Calculate D_i for each pair of power segments of (S_1, S_i) basing on the template model (such as CMP);
3) Match D_i with the templates obtained in the building template stage and obtain the Hamming distance H_i between p_i and β_i;

4) Update C_i according to H_i;
5) Repeat the procedure above until 15 collisions are detected or there occurred an error.

Note that if $|C_i| \leq 1$, step 1 to 4 can be omitted for p_i.

According to our simulation experimental results, 7.29 plaintexts are required on average to recover the whole key of AES-128. The average number of collision detections for each plaintext is 10.7.

B. COMPARED WITH COLLISION-CORRELATION ATTACK

We adopt the number of required plaintexts and traces to compare the efficiency of our method (CMP with self-correction) and collision-correlation attack, when applied to recover the whole key of masked AES-128. Table 2 summarizes the average number of plaintexts involved in a success attack, the number of traces required to achieve a success rate 95% and the computation cost of each plaintext for both methods. We explain it subsequently.

TABLE 2. The comparison between collision-correlation and our method.

Method	C.C. [26]	Our attack	Ratio
Num. of req. plaintexts	27.5	7.29	26.5%
Num. of req. traces ($l = 50$)	7700	2478.6	32.2%
Num. of req. traces ($l = 100$)	7150	1822.5	25.5%
Num. of req. traces ($l = 200$)	6050	1239.3	20.5%
Computation cost per plaintext	$120lC_{\rho_n}$	$10.70C_{\mathrm{CMP}_{nl}}$	$\ll$10%

l: the number of sample points in each power segment.
C_{ρ_n}: the computation cost of correlation coefficient calculated by equation 4.
$C_{\mathrm{CMP}_{nl}}$: the computation cost of central moment product calculated by equation 11.

Collision-correlation attack [26] requires 27.5 plaintexts and 120 calculations of correlation coefficient per plaintext to reduce the key space from 2^{128} to 2^8 (described in subsection II-D). While, our attack based on SC_CMP only requires 7.29 plaintexts and 10.7 calculations of central moment product per plaintext (described in subsection VI-A).

The number of traces required by the whole key recovery attack is estimated basing on the experimental results of attacks on two key bytes (given in subsection IV-D). Taking $\sigma = 3, l = 50$ for example, collision-correlation attack requires 280 traces per plaintext to achieve a success rate of 95%. Thus, $280 \times 27.5 = 7700$ traces are required in total. While, our attack based on SC_CMP requires $340 \times 7.29 = 2478.6$ traces on average, which is 32.2% of collision-correlation attack. For the cases of $l = 100$ and $l = 200$, the ratios are 25.5% and 20.5% respectively.

The computation costs of both methods are estimated basing on the number of multiplications. Denote the number of sample points in each power segments as l, the computation cost of correlation coefficient calculated by equation 4 as C_{ρ_n} and the computation cost of central moment product calculated by equation 11 as $C_{\mathrm{CMP}_{nl}}$.

- In collision-correlation attack, 120 collision detections are executed on each plaintext, and each collision detection has l calculations of correlation coefficient. So the computation cost of collision-correlation attack is $120lC_{\rho_n}$.
- In our method based on SC_CMP, 10.7 collision detections are executed on each plaintext, and each collision detection have 1 calculations of central moment product. So the computation cost of our method is $10.70C_{\mathrm{CMP}_{nl}}$.

There are $3n_{cc}$ multiplications in equation 4, so we have $C_{\rho_n} = 3n_{cc}$, where n_{cc} denotes the number of traces acquired with each plaintext in collision-correlation attack. There are ln_{cmp} multiplications in equation 11, so we have $C_{\mathrm{CMP}_l} = ln_{cmp}$, where n_{cmp} denotes the number of traces acquired with each plaintext in our method. As mentioned above, to achieve the success rate of 95%, the numbers of plaintexts required by two methods are 27.5 and 7.29, and the numbers of traces acquired with each plaintext are $n_{cc} = 280$ and $n_{cmp} = 340$ respectively. Therefore, the computation costs of collision-correlation attack and our method based on SC_CMP is $360l \times 280 \times 27.5 = 2772000l$ and $10.7l \times 340 \times 7.29 = 26521.02l$ multiplications respectively. The ratio between them is much less than 10%.

VII. CONCLUSIONS

In this paper, we focus on cryptographic algorithms implemented in edge computing and propose a new method to explore the extra information obtained from the traditional LSM-based collision attack on masked AES. This information helps to find a collision at a high pace instead of searching the plaintexts exhaustively. Furthermore, some models such as LAD, LAD$^\alpha$, CMP etc. are studied and compared. The new proposal can be combined with linear collision attack [20] (to deduce the relation between k_1 and k_2) and the fault-tolerant attack [23] (as mentioned in section V) naturally.

The attack proposed in this paper aims at two "MOV" instructions, which leak the information of $x_1 \oplus m$, $x_2 \oplus m$ and are executed individually. So it is applicable to all the reused-mask schemes in software implementation even if the masks are generated by other schemes or instructions. Moreover, for hardware implementation and the Hamming distance model, our method can also be competent for the attack, so long as the power traces leak the masked input values of S-boxes.

Some correlation-coefficient-based collision attacks [25], [26] could be modified as an adaptive chosen-plaintext ones, since the correlation coefficient also contains the information of Hamming distance between two intermediate values. However, this information is not as exact as that in our attack because its templates are not arithmetic progression and not distinguishable.

APPENDIX
CORRECTNESS OF OUR ATTACK

The correctness of Algorithm 1 depends on the fact that different $\mathrm{HD}(x_1, x_2)$ corresponds to different value of template, and their differences are non-ignorable. We give Theorem 2 and 5 to explain the two facts.

Theorem 2: In Algorithm 1, assume that l is fixed and Δ is a certain value from $\{0, 1, 2, ..., 8\}$. For any $x_1, x_2 \in \{0, 1, 2, ..., 255\}$ such that $\mathrm{HD}(x_1, x_2) = \Delta$,

the expectation

$$\xi_\Delta = E[\sum_{q=1}^{l}(t_{1,q} - t_{2,q})^2]$$

is determined.

According to the Hamming weight model, the expectation is expanded as

$$\begin{aligned}\xi_\Delta &= E[\sum_{q=1}^{l}(s_{1,q}\mathrm{HW}(x_1 \oplus m) + r_{1,q} \\ &\quad - s_{2,q}\mathrm{HW}(x_2 \oplus m) - r_{2,q})^2] \\ &= E[(\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m))^2 \sum_{q=1}^{l} s_q^2 \\ &\quad + \sum_{q=1}^{l}(r_{1,q} - r_{2,q})^2 + 2(\mathrm{HW}(x_1 \oplus m) \\ &\quad - \mathrm{HW}(x_2 \oplus m)) \sum_{q=1}^{l} s_q(r_{1,q} - r_{2,q})].\end{aligned}$$

Note that since the segments are aligned, the constants $s_{i,q}$ ($i = 1, 2, 3, ..., 16$) are equal to each other and simplified to s_q.

Before discussing the three terms in the formula, we give the following two lemma:

Lemma 3: Given a $\Delta \in \{0, 1, 2, ..., 8\}$*, for any* $x_1, x_2 \in \{0, 1, 2, ..., 255\}$ *such that* $\Delta = \mathrm{HD}(x_1, x_2)$*, the distribution of*

$$\delta = \mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m)$$

is determined if m follows uniform distribution in the interval $[0, 255]$.

Lemma 4: Given a $\Delta \in \{0, 1, 2, ..., 8\}$*, for any* $x_1, x_2 \in \{0, 1, 2, ..., 255\}$ *such that* $\mathrm{HD}(x_1, x_2) = \Delta$, $\{\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m) \mid m = 0, 1, 2, ..., 255\}$ *and* $\{\mathrm{HW}(x_1' \oplus m) - \mathrm{HW}(x_2' \oplus m) \mid m = 0, 1, 2, ..., 255\}$ *are equal if* x_1' *and* x_2' *are obtained by either of the following two operations*:

1) *Swap the i-th and j-th bits of* x_1 *to get* x_1'*, and swap the i-th and j-th bits of* x_2 *to get* x_2'.
2) *Alter the i-th bit of* x_1 *to get* x_1'*, and alter the i-th bit of* x_2 *to get* x_2'

Any pair of (x_1, x_2) that have the same Hamming distance, i.e. $\Delta = \mathrm{HD}(x_1, x_2)$ ($\Delta \in \{0, 1, 2, ..., 8\}$), can be transformed to each other by a serial operations given in Lemma 4. Thus, if the corresponding sets $\{\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m) \mid m = 0, 1, 2, ..., 255\}$ are equal to each other, the distributions of $\delta = \mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m)$ must be the same. Therefore, Lemma 3 is actually equivalent to Lemma 4. We give the proof of Lemma 4 as follows.

Proof: Denote $x_1 = u_1 \| u_2 \| ... \| u_8$, $x_2 = v_1 \| v_2 \| ... \| v_8$ and $m = w_1 \| w_2 \| ... \| w_8$. We have

$$\begin{aligned}\delta &= (u_1 \oplus w_1) + ... + (u_8 \oplus w_8) - (v_1 \oplus w_1) - ... - (v_8 \oplus w_8) \\ &= [(u_1 \oplus w_1) - (v_1 \oplus w_1)] + ... + [(u_8 \oplus w_8) - (v_8 \oplus w_8)].\end{aligned}$$

Since the formula follows the commutative law of addition, case 1) holds.

For case 2), we have $(x_1' \oplus 0, x_2' \oplus 0) = (x_1 \oplus (2^{9-i} - 1), x_2 \oplus (2^{9-i} - 1))$. So $\mathrm{HW}(x_1' \oplus 0) - \mathrm{HW}(x_2' \oplus 0) = \mathrm{HW}(x_1 \oplus (2^{9-i} - 1)) - \mathrm{HW}(x_2 \oplus (2^{9-i} - 1))$. Since m exhaust 0-255 in both sets, case 2) holds. □

According to Lemma 3, $E[(\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m))^2 \sum_{q=1}^{l} s_q^2]$ is determined with a given Δ. Since the expectation of $\sum_{q=1}^{l} s_q(r_{1,q} - r_{2,q})$ is 0, and $(r_{1,q} - r_{2,q})$ is independent with $(\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m))$, we have $E[2(\mathrm{HW}(x_1 \oplus m) - \mathrm{HW}(x_2 \oplus m)) \sum_{q=1}^{l} s_q(r_{1,q} - r_{2,q})] = 0$. Besides, $E[\sum_{q=1}^{l}(r_{1,q} - r_{2,q})^2]$ is determined by the noise in a certain device. Therefore, Theorem 2 is proven.

Then we discuss the non-negligible differences among the nine templates.

Theorem 5: In Algorithm 1, for all $\Delta \in \{0, 1, 2, ..., 8\}$*, the corresponding values of* ξ_Δ *form an arithmetic progression approximatively if sufficient sample points are obtained.*

Proof: According to the statistical model of Euclidean distance in [22], the expectation of Euclidean distance between two traces corresponding to fixed $x_1 \oplus m$ and $x_2 \oplus m$ is approximated as a normal distribution $\mathcal{N}(2\sigma^2(l+\lambda), 8\sigma^4(l+2\lambda))$ for sufficiently large l, where σ^2 denotes the noise variance of power consumption (it is roughly the same for all sample points [22]) and $\lambda = \sum_{q=1}^{l}(s_q\delta/\sqrt{2}\sigma)^2$.

Let λ_i, δ_i denote the values of λ, δ when $i = \mathrm{HW}(x_1 \oplus m)$. Based on the distribution of $\mathrm{HW}(x_1 \oplus m)$, we have

$$\begin{aligned}\xi_\Delta &= (C_\Delta^0/2^\Delta) \cdot 2\sigma^2(l + \lambda_0) + (C_\Delta^1/2^\Delta) \cdot 2\sigma^2(l + \lambda_1) \\ &\quad + ... + (C_\Delta^\Delta/2^\Delta) \cdot 2\sigma^2(l + \lambda_\Delta) \\ &= \frac{2\sigma^2 l}{2^\Delta}\sum_{i=0}^{\Delta} C_\Delta^i + \frac{2\sigma^2}{2^\Delta}\sum_{i=0}^{\Delta}(C_\Delta^i \lambda_i) \\ &= 2\sigma^2 l + \frac{2\sigma^2}{2^\Delta}\sum_{i=0}^{\Delta}(C_\Delta^i \cdot \frac{(i - (\Delta - i))^2}{2\sigma^2}\sum_{q=1}^{l} s_q^2) \\ &= 2\sigma^2 l + \frac{1}{2^\Delta}\sum_{q=1}^{l} s_q^2 \cdot \sum_{i=0}^{\Delta}(C_\Delta^i \cdot (2i - \Delta)^2) \\ &= 2\sigma^2 l + \Delta \cdot \sum_{q=1}^{l} s_q^2.\end{aligned}$$

Therefore, $\{\xi_\Delta \mid \Delta = 0, 1, 2, ..., 8\}$ forms an arithmetic progression with common difference of $\sum_{q=1}^{l} s_q^2$.

□

ACKNOWLEDGMENT

The main corresponding author An Wang, on behalf of all the authors, would like to thank the reviewers and the editors who

contribute to the great improvement of the original version of this paper with their valuable comments and suggestions.

REFERENCES

[1] A. A. Diro, N. Chilamkurti, and Y. Nam, "Analysis of lightweight encryption scheme for fog-to-things communication," *IEEE Access*, vol. 6, pp. 26820–26830, 2018.
[2] H. Xiong, Y. Zhao, L. Peng, H. Zhang, and K.-H. Yeh, "Partially policy-hidden attribute-based broadcast encryption with secure delegation in edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 453–461, Aug. 2019.
[3] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," *J. Chin. Inst. Eng.*, vol. 42, no. 1, pp. 20–28, Jan. 2019.
[4] T.-Y. Wu, X. Fan, K.-H. Wang, J.-S. Pan, and C.-M. Chen, "Security analysis and improvement on an image encryption algorithm using Chebyshev generator," *J. Internet Technol.*, vol. 20, no. 1, pp. 13–23, Jan. 2019.
[5] Z. Liu, K.-K. R. Choo, and J. Grossschadl, "Securing edge devices in the post-quantum Internet of Things using lattice-based cryptography," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 158–162, Feb. 2018.
[6] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, "A secure authentication protocol for Internet of vehicles," *IEEE Access*, vol. 7, pp. 12047–12057, 2019.
[7] K.-H. Wang, C.-M. Chen, W. Fang, and T.-Y. Wu, "On the security of a new ultra-lightweight authentication protocol in IoT environment for RFID tags," *J. Supercomput.*, vol. 74, no. 1, pp. 65–70, Jan. 2018.
[8] H. Xiong, Q. Mei, and Y. Zhao, "Efficient and provably secure certificateless parallel key-insulated signature without pairing for IIoT environments," *IEEE Syst. J.*, to be published.
[9] C. Xu, J. Ren, D. Zhang, and Y. Zhang, "Distilling at the edge: A local differential privacy obfuscation framework for IoT data analytics," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 20–25, Aug. 2018.
[10] H. Xiong, H. Zhang, and J. Sun, "Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing," *IEEE Syst. J.*, to be published.
[11] P. C. Kocher, "Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol. Conf.*, Jul. 1996, pp. 104–113.
[12] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Annu. Int. Cryptol. Conf.*, Dec. 1999, pp. 388–397.
[13] K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," in *Proc. Int. Workshop Fast Softw. Encryption*, Feb. 2003, pp. 206–222.
[14] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2004, pp. 16–29.
[15] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2002, pp. 13–28.
[16] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2010, pp. 320–334.
[17] Q. Wang, A. Wang, L. Wu, and J. Zhang, "A new zero value attack combined fault sensitivity analysis on masked AES," *Microprocessors Microsyst.*, vol. 45, pp. 355–362, Sep. 2016.
[18] Q. Wang, A. Wang, G. Qu, and G. Zhang, "New methods of template attack based on fault sensitivity analysis," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 3, no. 2, pp. 113–123, Jun. 2017.
[19] K. Schramm, G. Leander, P. Felke, and C. Paar, "A collision-attack on AES: Combining side channel—And differential-attack," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2004, pp. 163–175.
[20] A. Bogdanov, "Improved side-channel collision attacks on AES," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, Aug. 2007, pp. 84–95.
[21] A. Bogdanov, "Multiple-differential side-channel collision attacks on AES," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2008, pp. 30–44.
[22] A. Bogdanov and I. Kizhvatov, "Beyond the limits of DPA: Combined side-channel collision attacks," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1153–1164, Aug. 2012.
[23] B. Gérard and F.-X. Standaert, "Unified and optimized linear collision attacks and their application in a non-profiled setting," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Sep. 2012, pp. 175–192.
[24] Y. Ren, L. Wu, and A. Wang, "Double sieve collision attack based on bitwise detection," *Ksii Trans. Internet Inf. Syst.*, vol. 9, no. 1, pp. 296–308, Jan. 2015.
[25] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2010, pp. 125–139.
[26] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Sep. 2011, pp. 49–62.
[27] A. Wang *et al.*, "Efficient collision attacks on smart card implementations of masked AES," *Sci. China Inf. Sci.*, vol. 58, no. 5, pp. 1–15, May 2015.
[28] A. Wang, Y. Zhang, W. Tian, Q. Wang, G. Zhang, and L. Zhu, "Right or wrong collision rate analysis without profiling: Full-automatic collision fault attack," *Sci. China Inf. Sci.*, vol. 61, no. 3, Mar. 2018, Art. no. 032101.
[29] A. A. Sveshnikov, *Problems in Probability Theory, Mathematical Statistics and Theory of Random Functions*. New York, NY, USA: Dover, 1979.
[30] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Sep. 2001, pp. 309–318.
[31] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2008, pp. 426–442.
[32] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Proc. Annu. Int. Cryptol. Conf.*, Dec. 1999, pp. 398–412.
[33] F.-X. Standaert *et al.*, "The world is not enough: Another look on second-order DPA," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Dec. 2010, pp. 112–129.
[34] E. Oswald and S. Mangard, "Template attacks on masking—Resistance is futile," in *Proc. Cryptograph. Track Conf. (RSA)*, Feb. 2007, pp. 243–256.

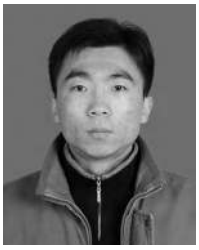
YAOLING DING received the B.S. degree in computer science and technology from Jilin University, in 2004. She is currently pursuing the Ph.D. degree in computer science and technology with Tsinghua University. Her research interests include side-channel attack and cryptanalysis of symmetric cipher.

YING SHI was born in 1989. She received the M.S. degree in information security from Shangdong University, in 2015. She is currently with the Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include side-channel analysis and cryptographic applications.

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

XUEXIN ZHENG was born in 1985. She received the Ph.D. degree in information security from Shangdong University, in 2014. She is currently with the China Academy of Electronics and Information Technology. Her research interests include lattice-based cryptography and side-channel attack.

ZONGYUE WANG received the Ph.D. degree in information security from Shandong University, in 2015. From 2015 to 2017, he was a Research Engineer with the China Academy of Information and Communications Technology. He is currently with Open Security Research, Shenzhen, China. His main research interests include side-channel analysis and cryptographic engineering.



GUOSHUANG ZHANG was born in 1982. He received the M.S. degree from the Zhengzhou Information Science and Technology Institute, in 2009. He is currently a Research Assistant with the Science and Technology on Information Assurance Laboratory. His main research interests include lattice-based cryptography and cryptanalysis.