Received January 8, 2019, accepted January 21, 2019, date of publication January 31, 2019, date of current version February 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2896256

An Efficient Collision Power Attack on AES Encryption in Edge Computing

YONGCHUAN NIU[1,2], JIAWEI ZHANG[2], AN WANG[1,3], AND CAISEN CHEN[4]
[1]School of Computer Science, Beijing Institute of Technology, Beijing 100081, China
[2]Data Communication Science and Technology Research Institute, Beijing 100191, China
[3]Key Laboratory of Network Assessment Technology, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[4]Military Exercise and Training Center, Army Academy of Armored Forces, Beijing 100072, China

Corresponding authors: Jiawei Zhang (zhangjiaweibj@hotmail.com) 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 has become a promising paradigm for the context-aware and delay-sensitive IoT data analytics. For the sake of security, some cryptographic algorithms such as AES, RSA, and so on, are employed for the encryption communication and authentication. The collision power attack is a typical physical attack to recover the secret key of the AES algorithm. However, almost all collision attacks aim at the detection of internal collisions caused by the output of S-boxes, and the linear layers are not concerned with those protected implementations. The relation between the mask and the masked data has been given little attention and stays as is, where the leakages still exist. In this paper, we focus on three typical AES implementations in edge computing, and propose a new type of collision attack by making use of leakages from linear layers, which is capable of breaking masking schemes with uniformly distributed random masks. In addition, a novel scalable collision attack of general applicability and high-efficiency is proposed and applied to masked linear layers and masked S-boxes. It can reach an equal level of performance compared to the second-order power analysis with acceptable off-line search, which improves the known collision attacks significantly.

INDEX TERMS Edge computing, collision attack, scalable collision, side-channel attack, linear layer.

I. INTRODUCTION

Edge computing has been a promising paradigm for context-aware and delay-sensitive IoT data analytics, through executing data handling in the edge of the network instead of the cloud. In the edge computing, some security requirements are paid much attentions to, such as encryption [1], authentication [2], privacy preservation [3], etc. For the security requirements, security model can be described in Figure 1. Secure communication and some corresponding authentications have been effectively protected, so adversaries cannot mount the traditional attacks. However in the scene of edge computing, the adversary can usually hold or touch the edge servers and IoT devices. So, he can conduct some physical attacks on the edge servers and IoT devices, such as power attack [4], fault injection [5], [6], cache attack [7], etc.

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

Power attack has drawn much attention since it was first introduced by Kocher *et al.* [4]. As one of its main research fields Collision attack has also become a major concern, and many research achievements have been published [8]–[17]. The first collision attack in the open scientific literature was proposed in [8], which makes use of internal collisions caused in three adjacent S-boxes of DES and gains information about the secret key-bits. Based on the idea of [8] and [9] proposed the collision attack towards the output bytes of MixColumns transformation of AES. These original ideas are based on detecting collisions in specific positions of the internal state for different runs of the algorithm. The approach in [9] was further improved by Bogdanov [10] by means of detecting equal inputs to various S-boxes both for different AES executions and the same AES run, and its success probability was largely enhanced. These previous methods act on unprotected implementations.

For secure implementation in edge computing and other applications, masking technique is suggested to blinding the

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

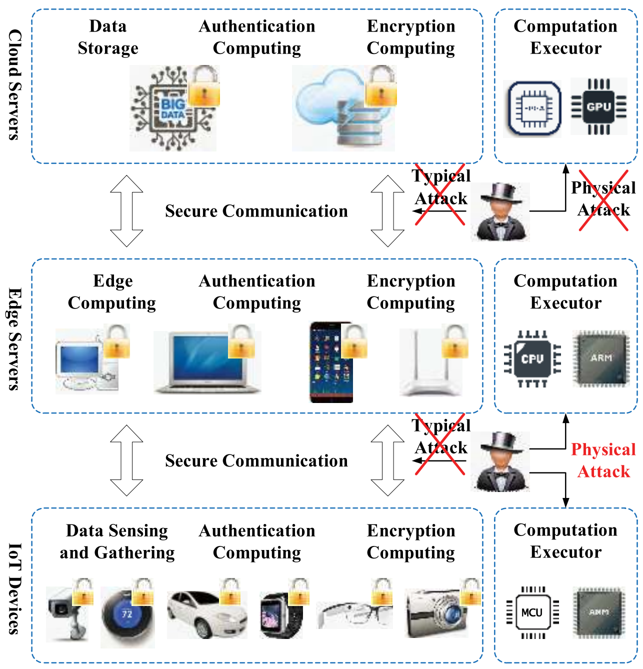


FIGURE 1. Security model and physical attack in edge computing.

information leakage [18]. Accordingly, collision attacks on masked implementations turned up. Moradi *et al.* [12] discovered the leakage inside the masked circuits due to uncontrolled hardware glitches. This method is restricted to the hardware implementation since the software platform may not generate glitches that are closely related to certain delicate operations of algorithms. Clavier *et al.* [13] utilized the re-use of masks to build the relation of masked data in various S-boxes, and reduced search scopes of the secret key dramatically. However, it is infeasible to break the countermeasure that uses uniformly distributed random masks. As long as uniformly distributed random masks are employed, it is easy to resist these tactics in existence. Moradi *et al.* [19] addressed that processing the mask and the masked data in the same way is one of the two major drawbacks which is not the most dominant research field and stays as is for most of the masking schemes. This on-the-fly mask compensation strategy may lead to a severe potential risk of leaking side-channel information.

In this work, we give three typical AES implementations with first-order resistant masking schemes in edge computing, and propose an innovative collision attack against these implementations. Then we improve our method and further propose the scalable collision attack which can be applied to all the known collision attacks to improve their corresponding success rate greatly. As a result, some experiments are made for verification of our proposed attack.

- Our collision attack methods target at the masking strategy of linear transformations for the first time. For general masking strategies of symmetric ciphers, the linear operations must be performed on the masked data as well as the mask. As long as the on-the-fly mask compensation strategy of linear layers is employed, it is easy to detect collision by using our collision attacks, which makes the masking strategy not secure any more.
- Most collision attacks take advantage of the re-use of masks, while the masking strategy with uniformly distributed random masks makes collision attacks invalid and only second-order attacks may function. Using the drawback of the mask compensation process, our proposed collision attacks are able to defeat this strategy no matter whether the masks are re-used or not. That is, our methods work effectively even on those implementations that have a different mask per byte and that change masks from S-box input to output and such implementations are considered secure in previous publications.
- The scalable collision attack is the first fault-tolerant collision attack. That is, the strictness of collision attack can be loosened to keep more useful key-related information that may lead to collision. As a result, the number of needed power traces is reduced significantly, which makes it the most efficient collision attack so far, compared to other known collision attacks.
- In scalable collision attack, the strictness of collision detection can be adjusted according to computation capability of the attacker. By properly choosing the variable parameters, scalable collision attack can meet the equal level of performance to second-order CPA with acceptable off-line search.

The remainder of this paper is organized as follows: Section II shows our targeted implementations of AES algorithm in edge computing. In Section III we introduce our mask-based collision attack method on linear layers. The general principle is presented, and an instance is followed to illustrate our new method. Then scalable collision attack is proposed and its basic idea is explained by taking the attack on linear layers as an example in Section IV. Section V introduces the scalable collision attack on masked S-boxes. Section VI concludes the paper and gives the comparison of typical collision attacks from the aspect of effectiveness.

II. AES ENCRYPTION IMPLEMENTATIONS IN EDGE COMPUTING

In order to illustrate proposed collision attack, we take the AES-128 algorithm as target. It contains 10 rounds, and each round consists of four stages: AddRoundKey, SubBytes, ShiftRows and MixColumns. The attack approaches we presented are also applicable to AES-192, AES-256 and other symmetric block ciphers.

We employ MathMagic side-channel analyzer with an AT89S52 micro-controller as an IoT device. In AT89S52 chip, AES algorithm is implemented by C codes, which can encrypt data transmitted among cloud server, edge server, and IoT devices. In practice, in order to acquire the side-channel information of IoT device, the adversary can refit the device, connect a power probe, and hold an EM probe close to the micro-controller. The side-channel information such as EM or power signal can be storaged in an

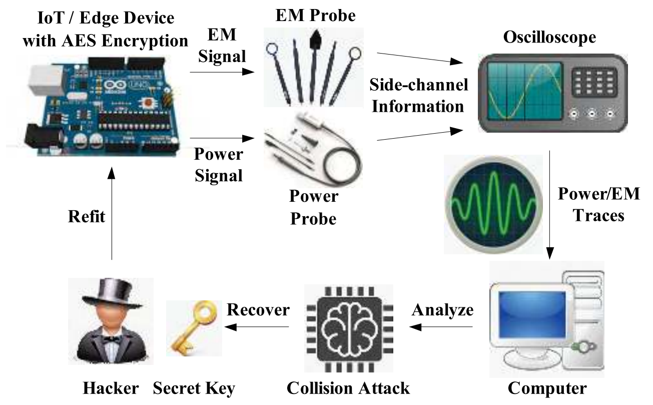


FIGURE 2. AES encryption in edge computing and its collision power attack.

oscilloscope, and then transmitted to a computer. The adversary can analyze the plaintext, ciphertext, and power or EM traces by collision attack, and recover the secret key of AES. As a result, he can decrypt all the data transmitted between this IoT device and other servers, although the data have been encrypted. Figure 2 shows this procedure. The similar physical attacks can also be mounted on the edge computing devices.

Masking is one of the most efficient countermeasures against side-channel attacks. To study the universality of proposed collision attack, we consider three common protected AES by masking.

A. RANDOM MASKING OF AES COMBINED WITH SHUFFLING S-BOXES

In order to protect cryptographic algorithms against side-channel attacks, two main countermeasures, masking and shuffling, are employed and taken as the general solutions. Masking is able to randomize the intermediate values processed in the cryptographic device by means of concealing every intermediate value with different random values. In this way, the power consumption generated by executing processes of the algorithm is independent of the intermediate values which may contain sensitive information. Together with masking, shuffling is another popular method to improve the security of block ciphers, which can spread sensitive information over different variables at different times and hide the dependencies between the intermediate values and power leakages.

A masked-and-shuffled AES algorithm implementation is described in Algorithm 1, whose countermeasures are similar to DPA contest v4.2 [20]. The only difference is the way they select masks. The low entropy masks are employed in DPA contest v4.2 while Algorithm 1 uses uniformly distributed random masks. In Algorithm 1, every intermediate value in the entire process of encryption is masked in bytes by an XOR operation with a random mask. Assuming that all the masks are randomized, there are 32 random values used in each round for the input and output of 16 S-boxes (320 in all), which are denoted as $mask_{r,w}$

Algorithm 1 Masked AES Implementation With Shuffling

```
Input: 16-byte plaintext $P$; 16-byte RoundKey,
       denoted as $Roundkey_i$, $i \in [0, 10]$; 8-bit random
       masks, denoted as $mask_{r,w}$, round number
       $r \in [0, 9]$, $w \in [0, 31]$; $mask_{r,0} \cdots mask_{r,15}$ and
       $mask_{r,16} \cdots mask_{r,31}$ are the input masks and
       output masks of S-boxes in round $r$ respectively.
Output: 16-byte Ciphertext.
1  $X = P$;
2  $Roundkey_0 = Roundkey_0 \oplus mask_{0,0\sim15}$;
3  for $r \in [0, 8]$ do
4      $X = X \oplus Roundkey_r$;
5      for $k \in F_{shuffle}([0, 15])$ do
6          $X_k = MaskedSubBytes(X_k)$ /*getting from
           lookup-table*/ ;
7      end
8      $X = ShiftRows(X)$;
9      $X = MixColumns(X)$ ;
10     $MaskCompensation =$
       $MixColumns(ShiftRows(mask_{r,16\sim31}))$ ;
11     $MaskCompensation =$
       $MaskCompensation \oplus mask_{r+1,0\sim15}$;
12     $X = X \oplus MaskCompensation$;
13 end
14 $X = X \oplus Roundkey_9$;
15 for $k \in F_{shuffle}([0, 15])$ do
16     $X_k = MaskedSubBytes(X_k)$;
17 end
18 $X = ShiftRows(X)$;
19 $MaskCompensation = ShiftRows(mask_{9,16\sim31})$;
20 $X = X \oplus Roundkey_{10}$;
21 return $X \oplus MaskCompensation$;
```

with $r \in [0, 9]$, $w \in [0, 31]$. Here, r is the round number, and $w \in [0, 15]$ and $w \in [16, 31]$ represent the indexes of the input and output masks of 16 S-boxes respectively. In the procedure presented in Algorithm 1, the lookup-table of MaskedSubBytes in step 6 is generated in advance by (1).

$$MaskedSubBytes(X_k) = SubBytes(X_k \oplus mask_{r,k}) \oplus mask_{r,k+16}, \quad k \in [0, 15] \tag{1}$$

Furthermore, shuffling is applied to the SubBytes layer to randomize the conducting order of the SubBytes operations of 16 S-boxes by means of a bijective function that is denoted as $F_{shuffle} : [0, 15] \rightarrow [0, 15]$. In extreme cases, the order of S-boxes are shuffled among 16! possibilities according to random permutation.

B. RANDOM MASKING OF AES COMBINED WITH SHUFFLING S-BOXES AND MIXCOLUMNS

Besides the countermeasures used in Algorithm 1 as well as DPA contest v4.2, the shuffling strategy is not only for

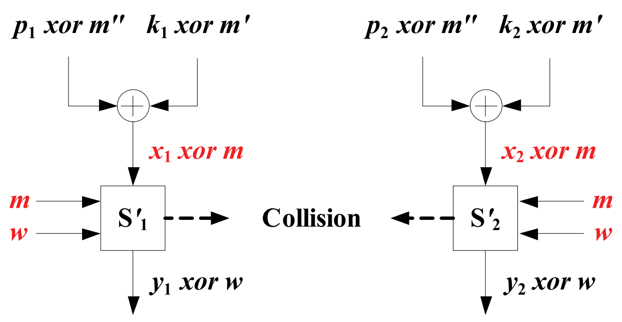


FIGURE 3. Re-used masking scheme of AES.

SubBytes, but it can be applied to MixColumns operations, which will also be taken into consideration to check the validity of our methods in the following pages.

C. RE-USED MASKING SCHEME OF AES

The leakage from re-used masks is considered, which is taken use of to achieve corresponding collisions in [13], which is described in Fig. 3, and our methods against these leakages are also discussed briefly.

III. COLLISION ATTACK ON LINEAR LAYERS OF AES

In this section, a new type of collision attack is proposed, which can retrieve the key of symmetric ciphers with first-order resistant masking scheme by targeting the linear layers. Firstly we introduce the general principle of our mask-based collision attack, then present a specific example of our approach to further illustrate our idea in detail, which takes the MixColumns operation as its attack point. Experiments are carried out to confirm its veracity. In the end, some discussions are made on the countermeasure of this attack and a simple solution is presented.

A. MASK-BASED COLLISION ATTACK ON LINEAR LAYERS OF AES

For block ciphers, masking techniques are used to conceal the relation between intermediate values and power traces, where the mask compensation is necessary to make sure that the old mask is removed and a new one is added. Since the masks are randomly selected and no enough memory can be provided to store all the precomputed results of the linear operations with all the random masks, the on-the-fly computation of mask compensation is unavoidable.

The general way of mask compensation is described in steps 10 and 11 of Algorithm 1. To the best of our knowledge, there is no other approach of removing the mask of the former round adopted on the review of the literature in existence.

In this approach, the linear operation of mask compensation is just the same as the way of processing masked data, denoted as $LinearLayer$. It is naturally drawn $LinearLayer(X) = LinearLayer(x \oplus m)$ when the masked data is processed, where X is the masked intermediate variable with a uniformly distributed random mask denoted as m and x is the sensitive actual value. It can be easily noticed that the mask compensation procedure $M = LinearLayer(m)$ is exactly the same as $LinearLayer(X)$ when $x = 0$. In other words, the procedure processing the mask collides with the procedure processing the masked data when x equals 0. As a result, the key can be retraced backwards from the collision results. Here, we name this collision **Linear Layer Collision Attack** (**LLCA**). Remarkably, our LLCA can be carried out in any specific subparts of the linear layers effectively. In next subsection, an example of our approach is presented by taking MixColumns transformation as the collided operation.

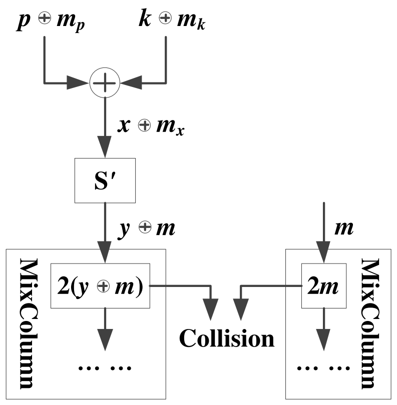


FIGURE 4. Collision attack on MixColumns.

B. COLLISION ATTACK ON MIXCOLUMNS

Collision attack on MixColumns is depicted in Figure 4. The MixColumns transformation performs a matrix multiplication which can be written as:

$$\begin{pmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} & h_{03} \\ h_{10} & h_{11} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} & h_{23} \\ h_{30} & h_{31} & h_{32} & h_{33} \end{pmatrix} \times \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \times \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \tag{2}$$

where a_{ij} indicates the input data of MixColumns layer in row i and column j, b_{ij} is the output data, and h_{ij} is the element of the fixed transformation matrix. Here a_{ij} is already masked with a random value.

After dealing with the masked data, the same operation is executed during the process of the mask compensation. As a result, the collision happens when the following formula is satisfied.

$$h_{ij} \cdot a_{jk} = h_{ij} \cdot m_{jk} \tag{3}$$

where m_{jk} is the input element of the MixColumns transformation of the mask compensation.

In order to illustrate our method clearly, we take the first output byte b_{00} as an example, which can be stated as

$$b_{00} = 02 \cdot a_{00} \oplus 03 \cdot a_{10} \oplus 01 \cdot a_{20} \oplus 01 \cdot a_{30} \tag{4}$$

Since software operations take byte (8-bits) as the operating unit, each single multiplication can be distinguished from other execution processes. Therefore, we can just consider the first multiplication of b_{00}, corresponding to step 9 of Algorithm 1, which is given by

$$\begin{aligned} 02 \cdot a_{00} &= 02 \cdot MaskedSubBytes(p_{00} \oplus k_{00} \oplus mask_{0,0}) \\ &= 02 \cdot (SubBytes((p_{00} \oplus k_{00} \oplus mask_{0,0}) \oplus mask_{0,0}) \\ &\quad \oplus mask_{0,16}) \\ &= 02 \cdot (SubBytes(p_{00} \oplus k_{00}) \oplus mask_{0,16}) \end{aligned} \tag{5}$$

where p_{00} is the first byte of input plaintext P, k_{00} is the first byte of $RoundKey_0$, $mask_{0,0}$ and $mask_{0,16}$ are the 8-bit input and output mask of the first S-box at the first round respectively (same as the notation in Algorithm 1).

According to step 10 of Algorithm 1, the first byte of the mask compensation after ShiftRows and MixColumns can be written as

$$\begin{aligned} &MaskCompensation_{00} \\ &\quad = 02 \cdot mask_{0,16} \oplus 03 \cdot mask_{0,21} \oplus 01 \\ &\qquad \cdot mask_{0,26} \oplus 01 \cdot mask_{0,31} \end{aligned} \tag{6}$$

Similarly, the first multiplication of $MaskCompensation_{00}$ is as follows.

$$02 \cdot mask_{0,16} \tag{7}$$

From (5) and (7), it can be inferred that these two operations collide if the following condition satisfies.

$$SubBytes(p_{00} \oplus k_{00}) = 0 \tag{8}$$

Hence, if the collision happens, k_{00} can deduced by

$$k_{00} = SubBytes^{-1}(0) \oplus p_{00}. \tag{9}$$

The whole MixColumns transformation consists of 64 multiplications, and each is related to one byte of round key independently, so we can select certain multiplication operations as attack targets. The difference between certain multiplication operations and $02 \cdot a_{00}$ is whether they need to get through the ShiftRows. Nevertheless, their corresponding mask compensations also go through the ShiftRows layer, which have the same effect on both operations. As a result, whether getting through the ShiftRows does not change collision results, and we can also establish the relation similar to (8) by tracing the executing trace of the ShiftRows backwards. The same conclusion can be drawn and the entire round key can be obtained by applying our method to 16 independent multiplications.

In general, collision attacks aim at pitfall of the re-use of masks, and previous first-order attacks can not break the protection on the correct software implementation with uniformly distributed random masks. However, LLCA can successfully attack first-order masking schemes no matter masks are re-used or not, and the on-the-fly mask compensation strategy of linear layers is no longer secure.

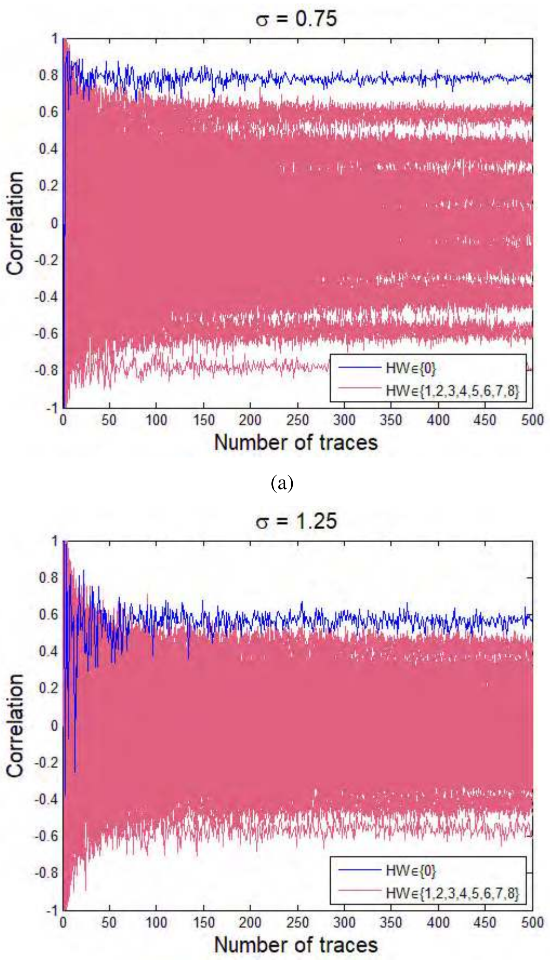


FIGURE 5. The simulation results of LLCA against Algorithm 1 over the number of traces under the Gaussian noise with different standard deviation, (a) $\sigma = 0.75$ and (b) $\sigma = 1.25$ respectively.

C. EXPERIMENTS AND RESULTS

1) SIMULATION EXPERIMENTS

In simulation experiments, we suppose that the power consumption is the Hamming weight of the intermediate data involved in the computation plus a Gaussian noise with standard deviation σ. Figure 5 shows the attack results of LLCA on Algorithm 1, from which we can see that LLCA can distinguish the Hamming weight 0 (blue curve) of S-box output from other Hamming weights (red curves) using different numbers of traces under Gaussian noise with $\sigma = 0.75$ and $\sigma = 1.25$ respectively. The results indicate LLCA can complete the attack successfully, which proves its effectiveness. However, although both (a) and (b) can distinguish the curve of Hamming weight 0 from others, it requires more traces

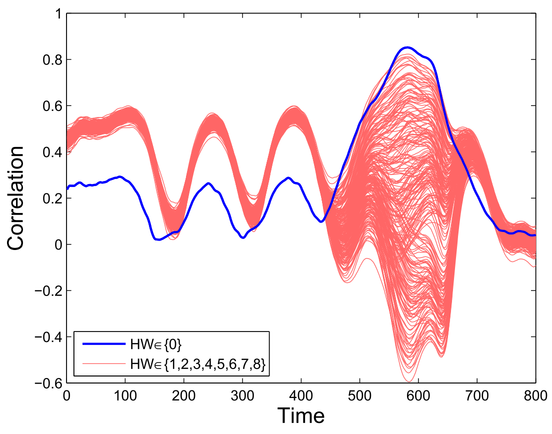


FIGURE 6. **Practical result of LLCA on Algorithm 1 using 1500 traces.**

as the Gaussian noise goes higher to achieve the success of collision.

2) PRACTICAL EXPERIMENTS IN EDGE COMPUTING

To emulate a scene of edge computing, we implement AES algorithm as C codes on AT89S52 processor of MathMagic side-channel analyzer. With sampling rate 1GSa/s, the power consumption can be acquired accurately during the encryption by PicoScope 3403D Oscilloscope. Our C codes can be easily programmed, and executed through ISO 7816 interface. Algorithm 1 is developed on this platform.

In our experiment, the MixColumns transformation on the first input byte is taken as the attack target, and the result of LLCA using 1500 traces is shown in Figure 6. Because the smart card we use is a general card that has significant noise, the correlation coefficient curve of the Hamming weight 0 is just a little higher than others. Even so, it can be distinguished and leads to a successful attack.

D. COUNTERMEASURE AND FURTHER DISCUSSIONS

Besides masking schemes, shuffling is used as a common countermeasure as well. Obviously, shuffling towards SubBytes can not resist against our method because LLCA attacks linear transformations instead of S-boxes. However, shuffling applied to the linear layers can defend against our attack to some extent, which makes the localization of two colliding positions in power traces difficult.

Similarly, we take $02 \cdot a_{00}$ as the example to analyze the effectiveness of our method when MixColumns transformation is shuffled. After randomizing the sequence of the processing of the columns, there are 4 positions where the power consumption of $02 \cdot a_{00}$ in power traces may appear, while $02 \cdot mask_{0,16}$ may appear in 4 positions with the same probability too. It is obvious that the correlation of the collision attack decreases to about 1/16 of the original value if not applying any extra tactics. In order to increase the effectiveness of collision attack, we integrate the two segments in power traces where $02 \cdot a_{00}$ and $02 \cdot mask_{0,16}$ may appear respectively. In our case, we respectively sum up the power consumption of all the 4 positions in these two segments, which reduces the correlation to a quarter under certain conditions according to Observation 1 and makes the attack success rate increase compared to 1/16.

Observation 1: We denote the power consumption at instants $t_1, t_2, \ldots, t_l$ by the random variables $P_1, P_2, \ldots, P_l$, and denote the power consumption at instants $t'_1, t'_2, \ldots, t'_l$ by the random variables $H_1, H_2, \ldots, H_l$. Furthermore, four assumptions are made as follows without loss of generality:

1) the power consumption $P_1, P_2, \ldots, P_l$ are independent of each other,

2) $P_2, P_3, \ldots, P_l$ are independent of $H_1, H_2, \ldots, H_l$,

3) the power consumption $H_1, H_2, \ldots, H_l$ are independent of each other,

4) $H_2, H_3, \ldots, H_l$ are independent of $P_1, P_2, \ldots, P_l$.

Based on the above assumptions, it holds for $i = 1, 2, \ldots, l$ and $j = 1, 2, \ldots, l$ that $E(P_i \cdot H_j) - E(P_i) \cdot E(H_j) = 0$ except for $i = j = 1$. The correlation between $\sum_{i=1}^{l} P_i$ and $\sum_{i=1}^{l} H_i$ can hence be calculated as follows:

$$\begin{aligned}
&\rho(\sum_{i=1}^{l} P_i, \sum_{i=1}^{l} H_i)\\
&\quad = \frac{E(\sum_{i=1}^{l} P_i \cdot \sum_{i=1}^{l} H_i) - E(\sum_{i=1}^{l} P_i) \cdot E(\sum_{i=1}^{l} H_i)}{\sqrt{Var(\sum_{i=1}^{l} P_i) \cdot Var(\sum_{i=1}^{l} H_i)}}\\
&\quad = \frac{E(P_1 \cdot H_1) - E(P_1) \cdot E(H_1)}{\sqrt{\sum_{i=1}^{l} Var(P_i) \cdot \sum_{i=1}^{l} Var(H_i)}}\\
&\quad = \frac{E(P_1 \cdot H_1) - E(P_1) \cdot E(H_1)}{\sqrt{Var(P_1) \cdot Var(H_1)} \cdot \sqrt{\frac{\sum_{i=1}^{l} Var(P_i) \cdot \sum_{i=1}^{l} Var(H_i)}{Var(P_1) \cdot Var(H_1)}}}\\
&\quad = \frac{\rho(P_1, H_1)}{\sqrt{\frac{\sum_{i=1}^{l} Var(P_i)}{Var(P_1)}}\sqrt{\frac{\sum_{i=1}^{l} Var(H_i)}{Var(H_1)}}} \qquad (10)
\end{aligned}$$

If the variances of the power consumption values $P_1, P_2, \ldots, P_l$ and $H_1, H_2, \ldots, H_l$ are equal respectively, the correlation between P_1 and H_1 can be lowered by l. That is to say $\rho(\sum_{i=1}^{l} P_i, \sum_{i=1}^{l} H_i) = \frac{\rho(P_1, H_1)}{l}$.

IV. SCALABLE COLLISION ATTACK ON LINEAR LAYERS OF AES

Although LLCA has good performance on a lot of block ciphers with general masking schemes, from the above discussions it can be seen that the shuffling techniques applied to linear transformations do have some negative effects on the success rate of our attack, and the attack may fail under some circumstances such as existence of big noise during algorithm

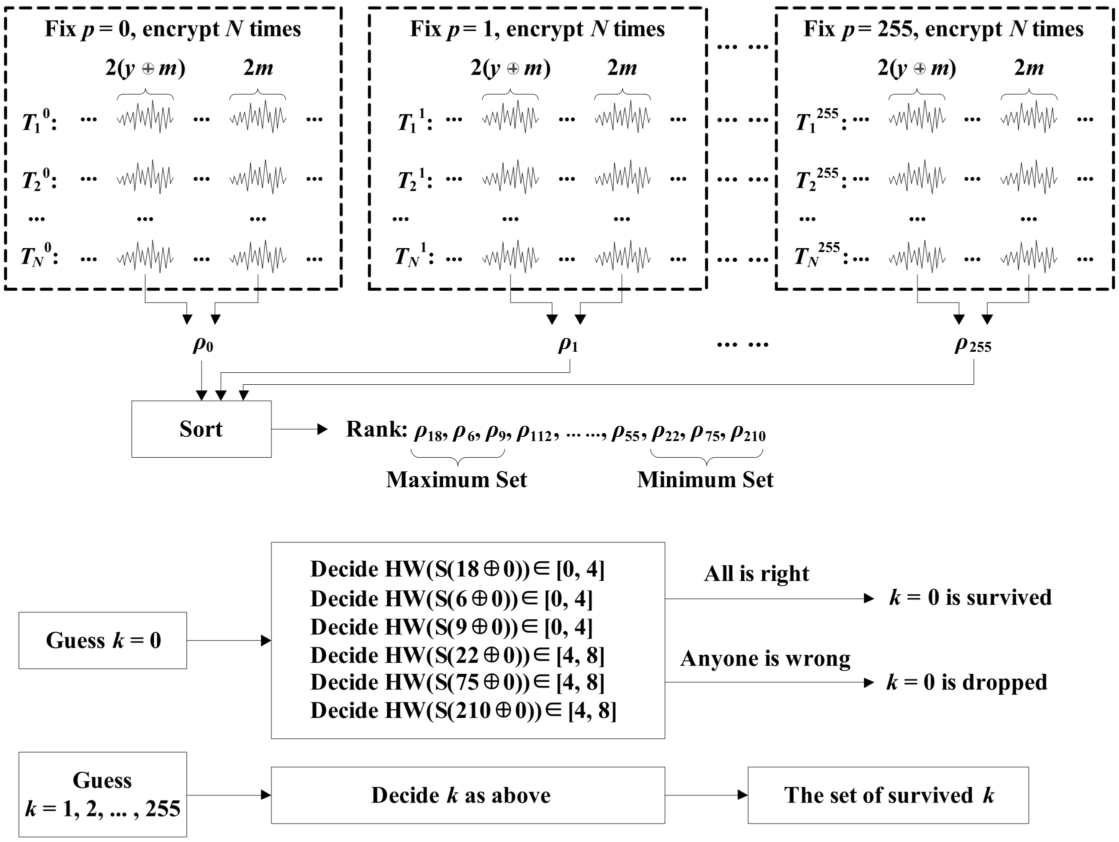

Fix p = 0, encrypt N times
2(y ⊕ m)
2m
Fix p = 1, encrypt N times
Fix p = 255, encrypt N times
ρ0
ρ1
ρ255
Sort
Rank: ρ18, ρ6, ρ9, ρ112, ……, ρ55, ρ22, ρ75, ρ210
Maximum Set
Minimum Set
Guess k = 0
Decide HW(S(18⊕0))∈[0, 4]
Decide HW(S(6⊕0))∈[0, 4]
Decide HW(S(9⊕0))∈[0, 4]
Decide HW(S(22⊕0))∈[4, 8]
Decide HW(S(75⊕0))∈[4, 8]
Decide HW(S(210⊕0))∈[4, 8]
All is right
k = 0 is survived
Anyone is wrong
k = 0 is dropped
Guess k = 1, 2, … , 255
Decide k as above
The set of survived k

FIGURE 7. The workflow of scalable collision attack on linear layers.

executions. Considering all these situations, we improve our LLCA and propose another way of attacking symmetric ciphers called **Scalable Collision Attack**.

A. BASIC IDEA OF SCALABLE COLLISION ATTACK

In this section, the basic idea of scalable collision attack is presented by taking the attack on linear layers as an example, which we call **scalable collision attack on linear layers** (**LLSCA**). Similar with the way of LLCA, we use the input of $02 \cdot a_{00}$ and the input of its corresponding mask compensation process $02 \cdot mask_{0,16}$ as the collision points, and some relations between $02 \cdot a_{00}$ and $02 \cdot mask_{0,16}$ are shown in Table 1. The first row lists all the possibilities of the Hamming weight of $a_{00} \oplus mask_{0,16}$, that is also the Hamming weight of $SubBytes(p_{00} \oplus k_{00})$ which can be derived from (11).

$$\begin{aligned} & a_{00} \oplus mask_{0,16} \\ & \quad = (SubBytes(p_{00} \oplus k_{00}) \oplus mask_{0,16}) \oplus mask_{0,16} \\ & \quad = SubBytes(p_{00} \oplus k_{00}) \end{aligned} \tag{11}$$

The second row ρ lists the correlation coefficients between $HW(a_{00})$ and $HW(mask_{0,16})$, which can be deduced from Observation 2. When p_{00} traverses all possibilities, the quantity distribution of p_{00} is stated in third row in reference to the Hamming weight of $SubBytes(p_{00} \oplus k_{00})$.

TABLE 1. The $\rho(HW(a_{00}), HW(mask_{0,16}))$ and quantity distribution of p_{00} for different Hamming weights of $SubBytes(p_{00} \oplus k_{00})$.

HW	0	1	2	3	4	5	6	7	8
ρ	1	0.75	0.5	0.25	0	-0.25	-0.5	-0.75	-1
$N(p_{00})$	1	8	28	56	70	56	28	8	1

TABLE 2. Correlation coefficients $\rho(HW(u), HW(v))$ for different Hamming weights of $u \oplus v$.

HW	0	1	2	3	4	5	6	7	8
ρ	1	0.75	0.5	0.25	0	-0.25	-0.5	-0.75	-1

Observation 2: We assume that u and v are 8-bit values, and use $HW(u)$ and $HW(v)$ to denote their Hamming weights. Table 2 lists their correlation coefficients for different values of $HW(u \oplus v)$.

Still, we calculate all the collision correlation coefficients for each 8-bit plaintext p_{00}, and sort them by correlation coefficient in descending order which can be denoted by the plaintext array $Pt = \{pt_0, pt_1, pt_2, \cdots, pt_{255}\}$. If there is no noise during algorithm executions, the Hamming weight array $HW(SubBytes(pt_i \oplus k_{00}))$ for $i \in [0, 255]$ is clear and consistent with Table 1. However, noise inevitably exists in practice, and some plaintexts whose S-box outputs have big Hamming weights may correspond to higher collision correlation coefficients than the correct ones due to existence of random noise. Therefore we propose the LLSCA that can deal with those implementations with low signal-to-noise ratio, whose principle is depicted in Figure 7 and described in detail as follows:

- The Hamming weight of $SubBytes(pt_0 \oplus k_{00})$ is most likely 0, and in this case k_{00} can be derived from $SubBytes^{-1}(0) \oplus pt_0$, which is just our proposed method in the above section. However, it may not still hold because of noise, and the following results can be further obtained.
- For pt_0, pt_1 and pt_2, their corresponding Hamming weights of S-box outputs are most likely to be 0 or 1. In this case the 8-bit candidate keys can be recovered by solving the following equations, and the success rate is improved compared to the above one.

$$\begin{cases} HW(SubBytes(pt_0 \oplus k_{00})) \in \{0, 1\} \\ HW(SubBytes(pt_1 \oplus k_{00})) \in \{0, 1\} \\ HW(SubBytes(pt_2 \oplus k_{00})) \in \{0, 1\} \end{cases} \tag{12}$$

- By that analogy, choosing different plaintexts and different numbers of equations, the 8-bit candidate keys can be recovered by solving any one of the following equations.

$$\begin{cases} HW(SubBytes(pt_0 \oplus k_{00})) \in \{0, 1, 2\} \\ HW(SubBytes(pt_1 \oplus k_{00})) \in \{0, 1, 2\} \\ \cdots \\ HW(SubBytes(pt_9 \oplus k_{00})) \in \{0, 1, 2\} \end{cases} \tag{13}$$

$$\begin{cases} HW(SubBytes(pt_0 \oplus k_{00})) \in \{0, 1, 2, 3\} \\ HW(SubBytes(pt_1 \oplus k_{00})) \in \{0, 1, 2, 3\} \\ \cdots \\ HW(SubBytes(pt_{13} \oplus k_{00})) \in \{0, 1, 2, 3\} \end{cases} \tag{14}$$

$$\begin{cases} HW(SubBytes(pt_0 \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \\ HW(SubBytes(pt_1 \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \\ \cdots \\ HW(SubBytes(pt_{29} \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \end{cases} \tag{15}$$

- In the same way, the array Pt can be utilized in the ascending order, and the 8-bit candidate keys can be recovered by solving any one of the following equations.

$$\begin{cases} HW(SubBytes(pt_{255} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \\ HW(SubBytes(pt_{254} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \\ \cdots \\ HW(SubBytes(pt_{226} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \end{cases} \tag{16}$$

$$\begin{cases} HW(SubBytes(pt_{255} \oplus k_{00})) \in \{5, 6, 7, 8\} \\ HW(SubBytes(pt_{254} \oplus k_{00})) \in \{5, 6, 7, 8\} \\ \cdots \\ HW(SubBytes(pt_{243} \oplus k_{00})) \in \{5, 6, 7, 8\} \end{cases} \tag{17}$$

$$\begin{cases} HW(SubBytes(pt_{255} \oplus k_{00})) \in \{6, 7, 8\} \\ HW(SubBytes(pt_{254} \oplus k_{00})) \in \{6, 7, 8\} \\ \cdots \\ HW(SubBytes(pt_{247} \oplus k_{00})) \in \{6, 7, 8\} \end{cases} \tag{18}$$

$$\begin{cases} HW(SubBytes(pt_{255} \oplus k_{00})) \in \{7, 8\} \\ HW(SubBytes(pt_{254} \oplus k_{00})) \in \{7, 8\} \\ HW(SubBytes(pt_{253} \oplus k_{00})) \in \{7, 8\} \end{cases} \tag{19}$$

- The Hamming weight 8 can be used directly as well as the case of Hamming weight 0, and the 8-bit key can be given by $k_{00} = SubBytes^{-1}(255) \oplus pt_{255}$.

The number of each equation listed above is determined by experiments for the purpose of unique determination of the key with a high probability.

In order to increase the attack success rate further, in practice we use (20) to perform LLSCA. As in Table 1, theoretically, it is expected that the Hamming weights of S-box outputs of the first 163 plaintexts in array Pt should be 0, 1, 2, 3 or 4. Similarly, it is expected that the Hamming weights of S-box outputs of the last 163 plaintexts in array Pt should be 4, 5, 6, 7 or 8. Actually the above assumptions may be not true due to noise influence. Fortunately, according to (20), we only need to find the first 15 plaintexts whose Hamming weights of S-box outputs are 0, 1, 2, 3 or 4 and the last 15 plaintexts whose Hamming weights of S-box outputs are 4, 5, 6, 7 or 8, which are easy to accomplish in reality.

$$\begin{cases} HW(SubBytes(pt_0 \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \\ HW(SubBytes(pt_1 \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \\ \cdots \\ HW(SubBytes(pt_{14} \oplus k_{00})) \in \{0, 1, 2, 3, 4\} \\ HW(SubBytes(pt_{255} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \\ HW(SubBytes(pt_{254} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \\ \cdots \\ HW(SubBytes(pt_{241} \oplus k_{00})) \in \{4, 5, 6, 7, 8\} \end{cases} \tag{20}$$

Equation (20) can find the correct candidates of p_{00} with greater probability, which means it has greater success rate than other equations in the case of high noise. In addition, (20) generally can have a unique solution of 8-bit key, but it doesn't matter whether there are more candidates since we can obtain the final key by simple exhaustive search.

In fact, because the Hamming weights of Sbox outputs obey binomial distribution, there is no need to traverse all the 256 possibilities of p_{00} and we can simply randomly select 256 p_{00} whose distribution is approximately consistent with Table 1 naturally. Furthermore, it is even not necessarily need to acquire 256 plaintexts, but instead we can randomly pick D plaintexts to perform the collision.

As well as D, we can also change the equation number E in our method, because no matter how many equations are used, the equations we construct contain right information on algorithm key as long as the plaintexts in equations are selected correctly. Therefore the number of the equations in (20) can be flexibly set.

It can be concluded that scalable collision attack can achieve different effects through different numbers of plaintexts D and different numbers of equations E. In high noise circumstances, we employ more plaintexts and fewer equations to make more candidate keys remained to insure the recovery of the right key, while in low noise circumstances, we can reduce the number of plaintexts and increase the number of the equations to recover the unique key efficiently. LLSCA utilizes the information of the power traces more fully than LLCA, which enhances the anti-noise capability.

Algorithm 2 Scalable collision attack on masked MixColumns

```
Input: The number of selected equations $E$, the number of plaintexts $D$, and the encryption times $N$ for each plaintext.
Output: The candidate key set $\mathbb{K}$.
Randomly choose $D$ plaintexts $p_0, p_1, \ldots, p_{D-1}$;
Encrypt each of plaintexts $(p_i)_{0 \leqslant i \leqslant D-1}$ for $N$ times, and acquire $D \times N$ traces $(T_{i,j})_{0 \leqslant i \leqslant D-1, 0 \leqslant j \leqslant N-1}$;
for $1 \leqslant a \leqslant 16$ do
    for $0 \leqslant i \leqslant D-1$ do
        Acquire the sub-traces $(\tau_{i,j,t'})_{0 \leq j \leq N-1, t_0 \leq t' < t_0+l_0}$ corresponding to $\mathrm{Sbox}(p_{i,a} \oplus k_a) \oplus m'$ and acquire $(\tau_{i,j,t''})_{0 \leq j \leq N-1, t_1 \leq t'' < t_1+l_1}$ corresponding to $m'$;
        for $s_0 = 0$ to $l_1 - 1$ do
            for $s_1 = 0$ to $l_0 - 1$ do
                $t' = t_0 + s_1$;
                $t'' = t_1 + (s_0 + s_1 \pmod{l_1})$;
                $v(i, s_0 * l_0 + s_1) = \rho((\tau_{i,j,t'})_{0 \leq j \leq N-1, t'}, (\tau_{i,j,t''})_{0 \leq j \leq N-1, t''})$;
            end
        end
    end
    $(\bar{i}, \overline{s_0}, \overline{s_1}) = \arg\max_{i,s_0,s_1}(v_{i,s_0*l_0+s_1})$;
    Sort $(p_{i,a})_{0 \leqslant i \leqslant D-1}$ by $(v_{i,\overline{s_0}*l_0+\overline{s_1}})_{0 \leqslant i \leqslant D-1}$ in descending order, denoted as $(pt_i)_{0 \leqslant i \leqslant D-1}$;
    $\mathbb{K} = \emptyset, flag = 0$;
    for $k = 0$ to $255$ do
        for $i = 0$ to $E/2 - 1$ do
            if $HW(SubBytes(pt_i \oplus k)) \in \{0, 1, 2, 3, 4\}$ then
                $flag = 0$;
            else
                $flag = 1$;
                break;
            end
        end
        if $flag == 0$ then
            for $i = D - E/2$ to $D - 1$ do
                if $HW(SubBytes(pt_i \oplus k)) \in \{4, 5, 6, 7, 8\}$ then
                    $flag = 0$;
                else
                    $flag = 1$;
                    break;
                end
            end
        end
        if $flag == 0$ then
            $\mathbb{K} = \mathbb{K} \cup k$;
        end
    end
    $\mathbb{K}$ are the candidate keys of $k_a$;
end
```

TABLE 3. The correlation coefficients ρ and quantity distribution of p_{00} for different Hamming weights of $SubBytes(p_{00} \oplus k_{00})$ on masked-and-shuffled implementation with additively shuffling MixColumns after integration of power traces.

HW	0	1	2	3	4	5	6	7	8
ρ	0.25	0.1875	0.125	0.0625	0	-0.0625	-0.125	-0.1875	-0.25
$N(p_{00})$	1	8	28	56	70	56	28	8	1

The detailed algorithm flow is shown in Algorithm 2, in which scalable collision attack on masked MixColumns to recover 16-byte key is given. In the simulation and practical experiments in the followings, we verify our algorithm when $D = 256$ and $E = 30$.

B. SCALABLE COLLISION ATTACK ON SHUFFLED LINEAR LAYERS OF AES

LLSCA can also be applied to the implementation with shuffling linear layers in the similar way as section III-D. The two processes of $02 \cdot a_{00}$ and $02 \cdot mask_{0,16}$ randomly spread over 4 different locations respectively, and we add the 4 power traces of $02 \cdot a_{00}$, $02 \cdot a_{01}$, $02 \cdot a_{02}$, $02 \cdot a_{03}$ and the 4 power traces of $02 \cdot mask_{0,16}$, $02 \cdot mask_{0,17}$, $02 \cdot mask_{0,18}$, $02 \cdot mask_{0,19}$ respectively to obtain two integrated power traces. Then Table 3 can be obtained, where HW still denotes the Hamming weight of the $SubBytes(p_{00} \oplus k_{00})$ and ρ denotes the correlation coefficient between $HW(a_{00}) + HW(a_{01}) + HW(a_{02}) + HW(a_{03})$ and $HW(mask_{0,16}) + HW(mask_{0,17}) + HW(mask_{0,18}) + HW(mask_{0,19})$.

Compared with Table 1, values of ρ in Table 3 change in equal proportion, which is the only difference between Table 3 and Table 1. Therefore, (20) also satisfies and can be applied to perform the scalable collision attack even on shuffled linear layers. Remarkably, LLSCA is good at the case of low signal-to-noise ratio, which makes LLSCA even better on shuffled implementation than LLCA and enhances the attack success rate greatly.

C. COLLISION DETECTION ALGORITHM

The collision detection method that we use is based on the collision-correlation method proposed by Clavier *et al.* [13] and it is simply described in Figure 8. A set of N power traces is captured when N encryptions of message M are executed, denoted as $(T^n)_{0 \leq n \leq N-1}$. Power trace segments Θ_0 and Θ_1 collide with each other in reality, but they can not be found directly just by observation. Hence, instead of finding the locations of Θ_0 and Θ_1, we choose two power consumption segments Φ_0 and Φ_1 to detect the collision based on the following principles.

- $\Phi_0 = (T^n_{t'})_{0 \leq n \leq N-1, t_0 \leq t' < t_0+l_0}$ is the first segment, the set of curves starting at time t_0 and including l_0 points. It should be much more likely to contain the colliding target point, which means

$$\Phi_0 \cap \Theta_0 \neq \varnothing. \tag{21}$$

Algorithm 3 Collision Detection Function

```
Input: the first curve segment
    $\Phi_0 = (T^n_{t'})_{0\le n\le N-1, t_0\le t'<t_0+l_0}$, the second curve
    segment $\Phi_1 = (T^n_{t''})_{0\le n\le N-1, t_1\le t''<t_1+l_1}$.
Output: correlation curve $\tau(t)$, $0 \le t \le l_0 * l_1 - 1$.
1 for $i = 0$ to $l_1 - 1$ do
2     for $j = 0$ to $l_0 - 1$ do
3         $t' = t_0 + j$;
4         $t'' = t_1 + (i + j \pmod{l_1})$;
5         $\tau(i * l_0 + j) =$
          $\rho((T^n_{t'})_{0\le n\le N-1,t'}, (T^n_{t''})_{0\le n\le N-1,t''})$;
6     end
7 end
8 return $\tau$
```

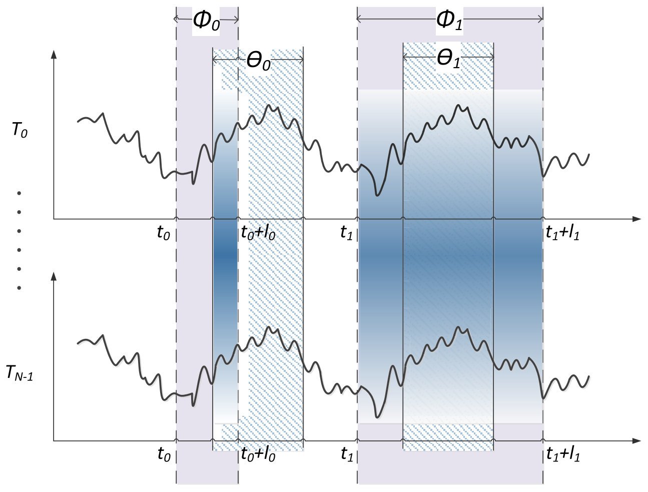


FIGURE 8. Collision detection algorithm.

- The other segment covers a relatively large range in the traces, which starts at t_1 and includes l_1 points, denoted as $\Phi_1 = (T^n_{t''})_{0\le n\le N-1, t_1\le t''<t_1+l_1}$. It should satisfy

$$\Theta_1 \subset \Phi_1. \tag{22}$$

If these two principles are satisfied, the method in Algorithm 3 is adopted to calculate the correlation curve τ between the curve segments Φ_0 and Φ_1. In LLCA, we make p_{00} traverse all the 256 values and get 256 correlation curves Γ_{256} consequently. Then we find out the curve τ with the relatively highest correlation over certain consecutive period of time in Γ_{256}, denoted as $\overline{\tau}$. If its corresponding p_{00} makes (8) satisfied, collision detection succeeds. In LLSCA, the method of collision detection is similar to that used in LLCA. After seeking out $\overline{\tau}$ in Γ_{256}, we pick one instant with the highest value in $\overline{\tau}$, sort the 256 correlation curves at this instant, and recover the candidate key by using the top 15 and the last 15 correlation curves.

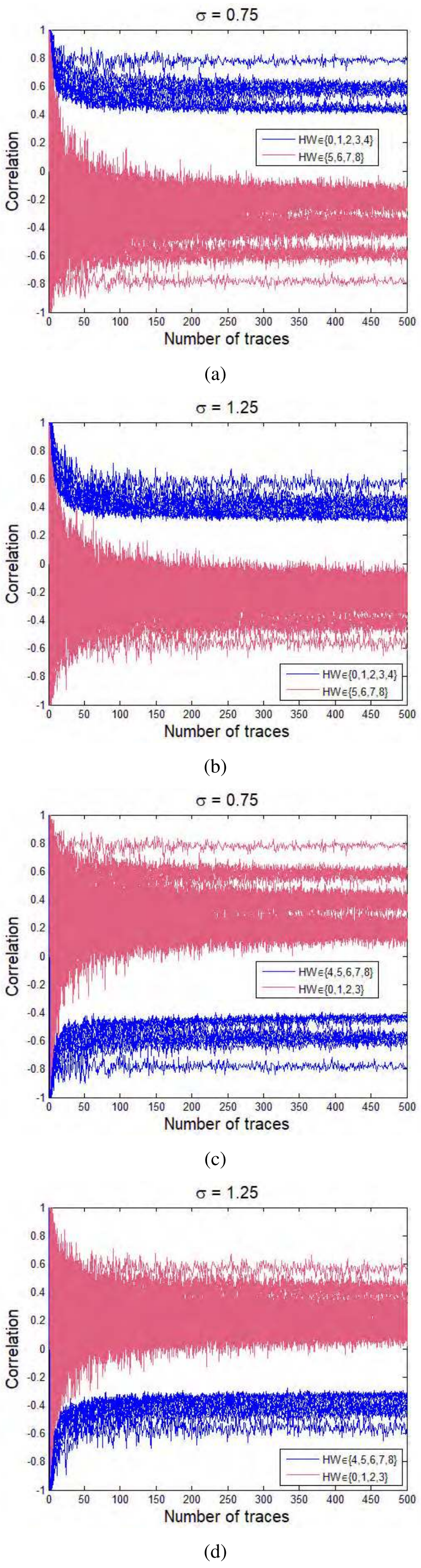


FIGURE 9. Simulation results of LLSCA against Algorithm 1 over the number of traces under different Gaussian noise.

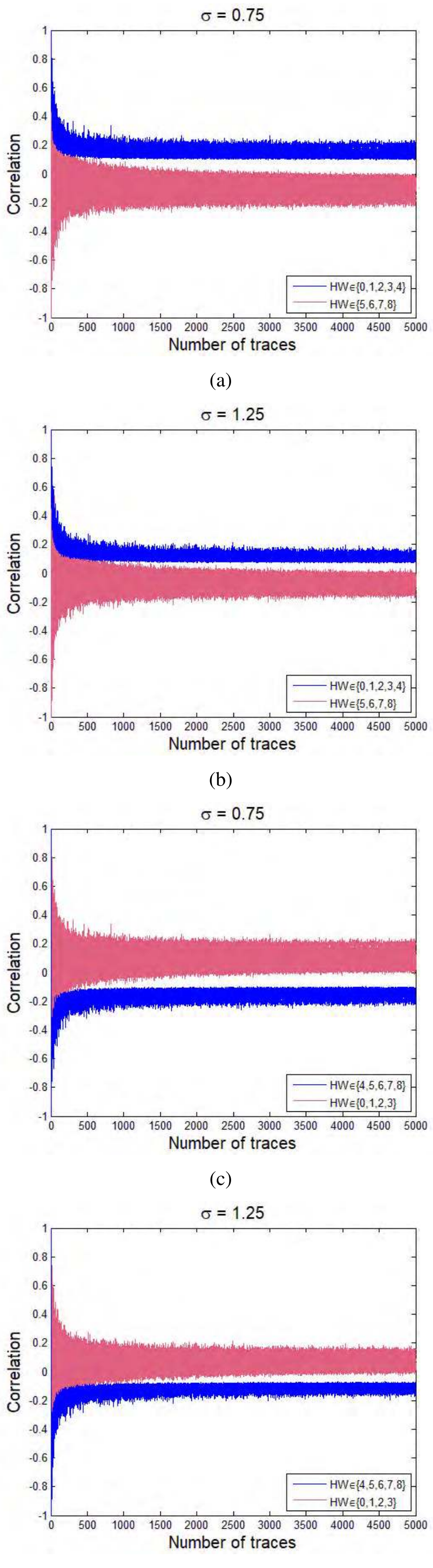


FIGURE 10. Simulation results of LLSCA against Algorithm 1 with shuffled MixColumns over the number of traces under different Gaussian noise.

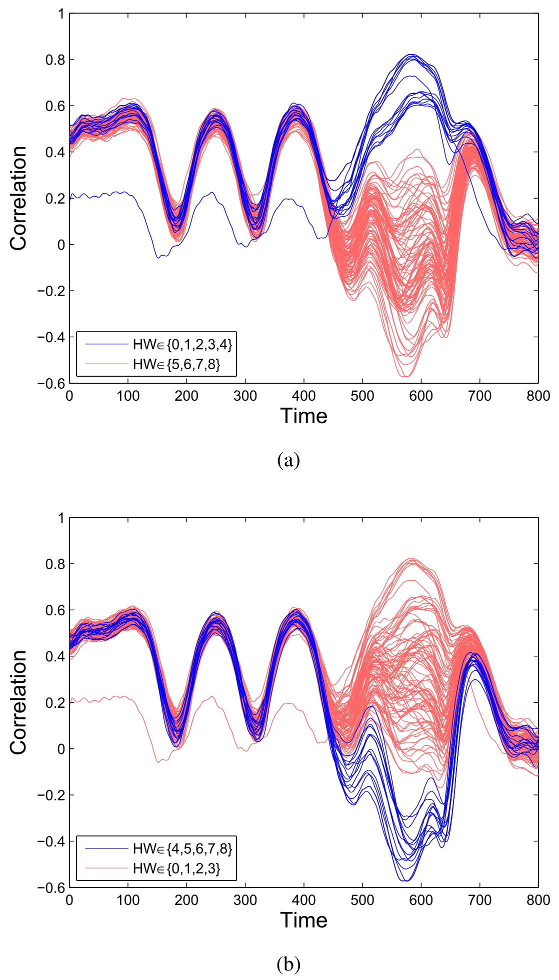
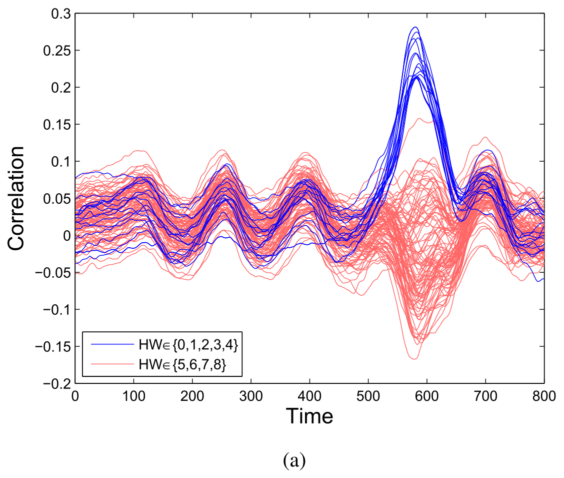


FIGURE 11. The practical results of LLSCA on Algorithm 1 using 600 traces.

D. EXPERIMENTS AND RESULTS

1) SIMULATION EXPERIMENTS

As previously mentioned, it requires two conditions for LLSCA to achieve the collision when using (20). One is that the top 15 correlation coefficient curves whose corresponding Hamming weights of S-box outputs belong to {0, 1, 2, 3, 4} (blue curves) can be distinguished from those curves with Hamming weights belonging to {5, 6, 7, 8} (red curves), as shown in (a) and (b) in Figure 9. The other is that the last 15 correlation coefficient curves whose corresponding Hamming weights of S-box outputs belong to {4, 5, 6, 7, 8} (blue curves) can be distinguished from those curves with Hamming weights belonging to {0, 1, 2, 3} (red curves), shown in (c) and (d). From (20) we can draw the conclusion that LLSCA is valid when both of the two partitions are accomplished correctly. (a) and (c) show that only 14 traces are needed to achieve the attack when $\sigma = 0.75$, while 40 traces would be proper for $\sigma = 1.25$ as shown in (b) and (d). However, compared the result of LLCA, LLSCA requires fewer traces than LLCA and thus LLSCA is more efficient especially in the high noise environment.



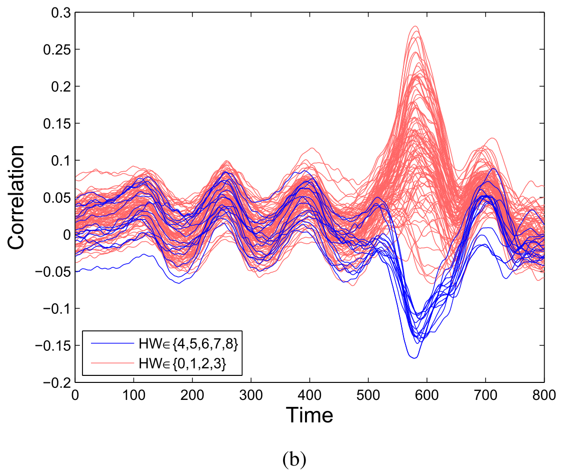
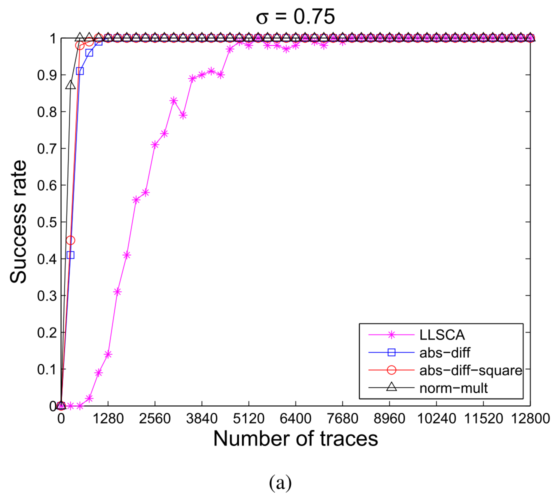


FIGURE 12. The practical results of LLSCA on Algorithm 1 with shuffled MixColumns using 1500 traces.



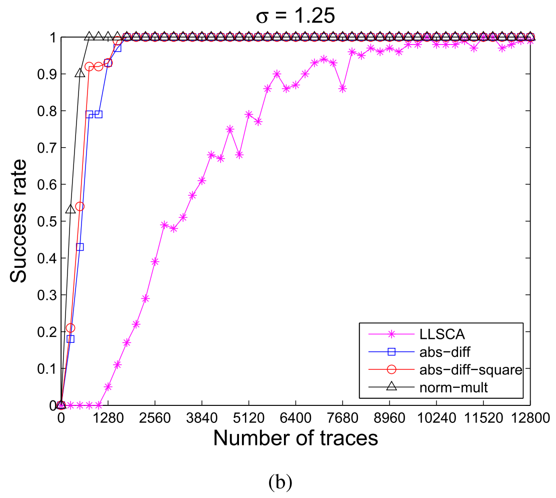


FIGURE 13. Performance comparison between the worst-case LLSCA using (20) and classical second-order CPA under the Gaussian noise with different standard deviation, (a) $\sigma = 0.75$, (b) $\sigma = 1.25$.

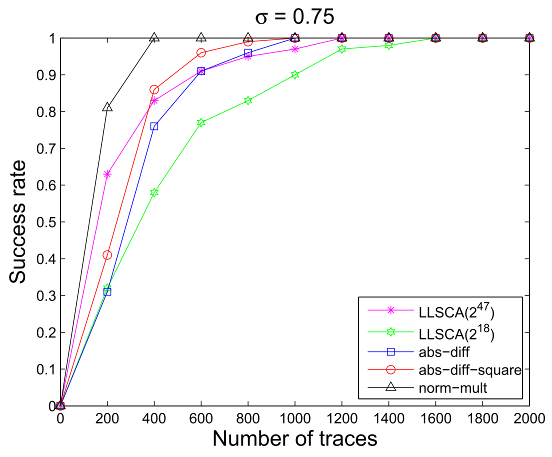
For the implementation with shuffled MixColumns, 5000 traces are used to verify our methods and the corresponding attack results are shown in Figure 10. In this case, LLCA can not accomplish the attack because of the low signal-to-noise rate, while LLSCA can still succeed and distinguish clearly under the same circumstances, which further proves the validity of scalable collision attack in high noise environment.

2) PRACTICAL EXPERIMENTS IN EDGE COMPUTING

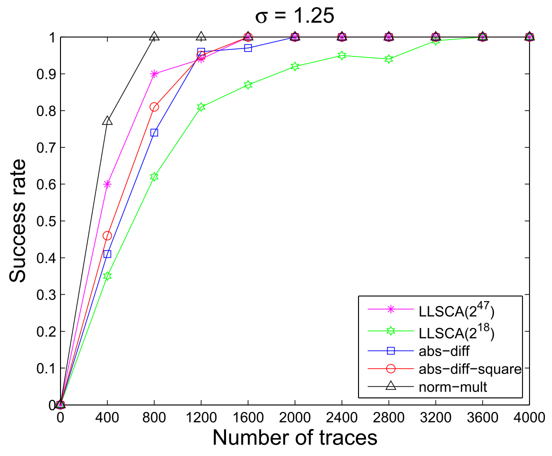
As above, we implement AES algorithm as C codes on AT89S52 processor, and acquire the power traces by oscilloscope. The advantage of LLSCA shows up obviously in the practical experiments, and its attack results are shown in Figure 11 and Figure 12 when attacking Algorithm 1 using 600 traces and Algorithm 1 with shuffled MixColumns using 1500 traces respectively. It can be seen in Figure 11 that when the collision occurs, the top 15 correlation coefficient curves whose corresponding Hamming weights of S-box outputs belong to $\{0, 1, 2, 3, 4\}$ (blue curves in (a)) are much higher than those curves with Hamming weights belonging to $\{5, 6, 7, 8\}$ (red curves in (a)), and the last 15 correlation coefficient curves whose corresponding Hamming weights of S-box outputs belong to $\{4, 5, 6, 7, 8\}$ (blue curves in (b)) are much lower than those curves with Hamming weights belonging to $\{0, 1, 2, 3\}$ (red curves in (b)). As a result, (20) can be constructed, by solving which the 8-bit key can be obtained. Sometimes, there may be more than one 8-bit candidate key obtained from the equations, and we can recover the final key by means of the exhaustive search after getting all the candidates for all the 16 8-bit keys. The same goes for the attack against the implementation with shuffled MixColumns, which is shown in Figure 12. Compared to Figure 11, the correlation at the moment of collision is much lower, but it is distinguishable amongst others.

3) COMPARISON WITH SECOND-ORDER CPA ON LINEAR LAYERS OF AES

Three typical preprocessing functions for second-order CPA are considered, including the absolute difference function, the squared absolute difference function and the normalized product function.



(a)



(b)

FIGURE 14. Performance comparison between LLSCA after relaxing the conditions and classical second-order CPA under the Gaussian noise with different standard deviation, (a) $\sigma = 0.75$, (b) $\sigma = 1.25$.

Figure 13 shows the comparison between the worst case of our collision attacks and classical second-order CPA. In the worst case, collision attacks use $256*N$ traces for all 256 possibilities of p_{00} while N traces are needed for second-order CPA.

Remarkably, the number of the equations our scalable collision attack uses is variable and the complexity of the attack is adjustable. Therefore, when LLSCA runs, we can flexibly set the variable parameters instead of using (20) directly. To evaluate the influence on different parameter settings, 20 plaintexts are randomly selected and scalable collision attack is given by using two different number of equations: 8 and 12. Their corresponding average complexity of off-line search is less than 2^{47} and 2^{18} respectively to recover 16-byte key. Figure 14 shows the comparison between the LLSCA after relaxing its conditions and classical second-order CPA, from which we can clearly see that the number of traces needed by scalable collision attack approximates that of second-order CPA when the complexity of the off-line search is 2^{18}, while scalable collision attack can reach the equal level of performance compared to second-order CPA if the search complexity can be broadened to 2^{47}.

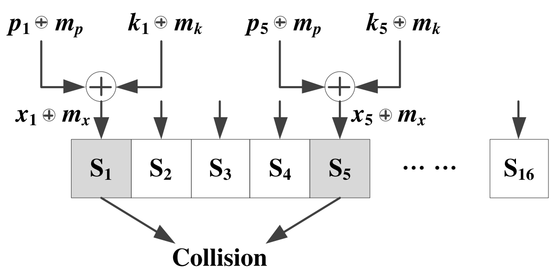


FIGURE 15. The application scenario of scalable collision attack on masked S-boxes.

In addition, collision attack has its own crucial advantages: it does not require the details of algorithm implementation and the intermediate values during the attack process and the only knowledge it needs is that two targeted operations are implemented in the same way. Besides, collision attack still has some special applications. For example, the SCARE (Side-Channel Analysis Reverse Engineering) method [21].

V. SCALABLE COLLISION ATTACK ON MASKED S-BOXES OF AES

Scalable collision attack can be applied not only to linear layers but also to S-boxes. In fact, nearly all collision attacks can be improved by scalable collision attack and the corresponding attack success rates can be increased significantly. In this section, scalable collision attack is used to attack the masked S-boxes whose masks are re-used (Figure 15) based on the collision attack in [13].

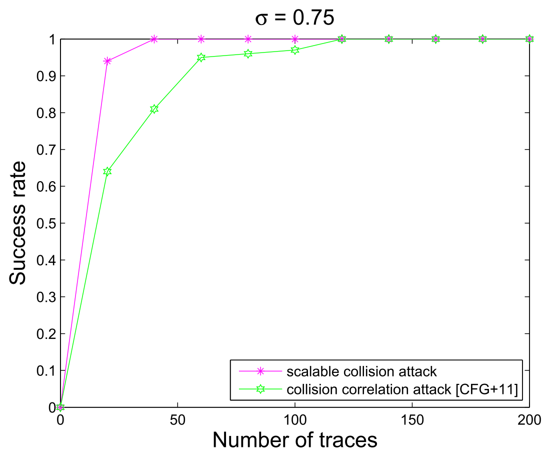
A. BRIEF DESCRIPTION OF SCALABLE COLLISION METHOD ON MASKED S-BOXES

We randomly select 256 plaintexts, and encrypt each of them for N times, and acquire the power traces corresponding to $SubBytes(p_a \oplus k_a)$ and $SubBytes(p_b \oplus k_b)$ respectively. By using collision detection method, the corresponding 30 p_a and p_b are obtained and (23) is constructed to recover k_a and k_b.

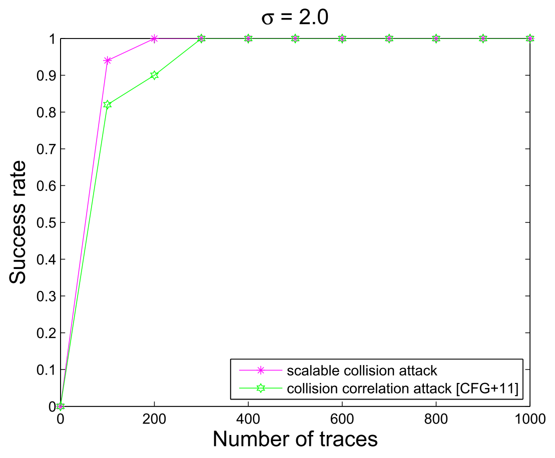
$$\begin{cases} HW(SubBytes(p_a^0 \oplus k_a) \oplus SubBytes(p_b^0 \oplus k_b)) \\ \qquad \in \{0, 1, 2, 3, 4\} \\ HW(SubBytes(p_a^1 \oplus k_a) \oplus SubBytes(p_b^1 \oplus k_b)) \\ \qquad \in \{0, 1, 2, 3, 4\} \\ \ldots \\ HW(SubBytes(p_a^{14} \oplus k_a) \oplus SubBytes(p_b^{14} \oplus k_b)) \\ \qquad \in \{0, 1, 2, 3, 4\} \\ HW(SubBytes(p_a^{255} \oplus k_a) \oplus SubBytes(p_b^{255} \oplus k_b)) \\ \qquad \in \{4, 5, 6, 7, 8\} \\ HW(SubBytes(p_a^{254} \oplus k_a) \oplus SubBytes(p_b^{254} \oplus k_b)) \\ \qquad \in \{4, 5, 6, 7, 8\} \\ \ldots \\ HW(SubBytes(p_a^{241} \oplus k_a) \oplus SubBytes(p_b^{241} \oplus k_b)) \\ \qquad \in \{4, 5, 6, 7, 8\} \end{cases} \tag{23}$$

TABLE 4. Typical collision attack methods and their attack capabilities.

	Re-used masks [18]	Low entropy masks [20]	Typical random masks	Random masks & shuffling
LCA [10]	×	×	×	×
CCA [13]	√	×	×	×
NLCA [16]	√	×	×	×
NCA [17]	√	×	×	×
LLCA	√	√	√	×
scalable CA	√	√	√	√



(a)



(b)

FIGURE 16. Success rates of scalable collision attack on masked S-boxes under the Gaussian noise, (a) $\sigma = 0.75$, (b) $\sigma = 2$.

B. COMPARISON WITH COLLISION CORRELATION ATTACK

Figure 16 shows the evolution of the success rate for scalable collision attack and the collision attack in [13] under different noise levels. It is obvious that the performance of scalable collision attack is better than that of collision correlation attack no matter σ equals 0.75 or 2. As a matter of fact, scalable collision attack is the most efficient collision attack so far, compared to all the known collision attacks.

VI. CONCLUSION

In this paper we propose two novel collision attack on block ciphers: one is the collision attack on linear layers, the other is the scalable collision attack. Focusing on the edge computing scene, we conduct practical attack experiments to verify our proposal. Through analyzing the relation between the mask and the masked data, we find the leakage of linear layers and further propose the mask-based collision attack against software implementations of symmetric ciphers (LLCA) by utilizing this leakage. It is the first time that collision attack takes masked linear layers as the attack target. In general, this method is capable of breaking the first-order masking strategy that uses uniformly distributed random masks.

However, shuffling techniques enhance noise and do have some influences on LLCA. Therefore, we improve LLCA and propose scalable collision attack on linear layers (LLSCA) which not only inherits all the advantages of LLCA but also does well in the case of low signal-to-noise ratio. LLSCA can flexibly change the number of equations it constructs according to different noise levels and extract key-related information by fully use of power traces. As a result, when the variable parameters are properly set, LLSCA can reach equal level of performance to second-order CPA with acceptable off-line search. Furthermore, scalable collision attack can be applied to many kinds of application scenarios to improve the known collision attack, and increase the success rate greatly.

At last, Table 4 summarises several typical collision attacks and their attack capabilities which also suggest possible protections. The typical collision attacks we select include linear collision attack (LCA), collision correlation attack (CCA), non-linear collision attack (NLCA), near collision attack (NCA), and two proposed collision attacks (LLCA and scalable CA).

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] Z. Liu, K.-K. R. Choo, and J. Großschädl, "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.

[3] 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.

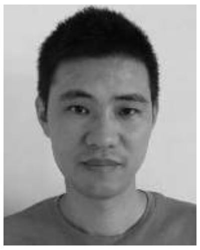
[4] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Adv. Cryptol. 19th Annu. Int. Cryptol. Conf. (CRYPTO)* (Lecture Notes in Computer Science), vol. 1666, M. J. Wiener, Ed., Santa Barbara, CA, USA: Springer, Aug. 1999, pp. 388–397.

[5] 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.
[6] 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, Apr./Jun. 2017.
[7] C. Chen, T. Wang, Y. Kou, X. Chen, and X. Li, "Improvement of trace-driven I-Cache timing attack on the RSA algorithm," *J. Syst. Softw.*, vol. 86, no. 1, pp. 100–107, 2013.
[8] K. Schramm, T. Wollinger, and C. Paar, "A new class of collision attacks and its application to DES," in *Proc. Fast Softw. Encryption, 10th Int. Workshop Fast Softw. Encryption* (Lecture Notes in Computer Science), vol. 2887, T. Johansson, Ed. Lund, Sweden: Springer, Feb. 2003, pp. 206–222.
[9] K. Schramm, G. Leander, P. Felke, and C. Paar, "A collision-attack on AES: Combining side channel- and differential-attack," in *Proc. 6th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)* (Lecture Notes in Computer Science), vol. 3156, M. Joye and J. J. Quisquater, Eds. Cambridge, MA, USA: Springer, Aug. 2004, pp. 163–175.
[10] A. Bogdanov, "Improved side-channel collision attacks on AES," in *Proc. 14th Int. Workshop Sel. Areas Cryptogr.* (Lecture Notes in Computer Science), vol. 4876, C. Adams, A. Miri, and M. Wiener, Eds. Ottawa, MTL, Canada: Springer, Aug. 2007, pp. 84–95.
[11] A. Bogdanov, "Multiple-differential side-channel collision attacks on AES," in *Proc. 10th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)* (Lecture Notes in Computer Science), vol. 5154, E. Oswald and P. Rohatgi, Eds. Washington, DC, USA: Springer, Aug. 2008, pp. 30–44.
[12] A. Moradi, O. Mischke, and T. Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Proc. 12th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)* (Lecture Notes in Computer Science), vol. 6225, S. Mangard and F. Standaert, Eds. Santa Barbara, CA, USA: Springer, Aug. 2010, pp. 125–139.
[13] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Improved collision-correlation power analysis on first order protected AES," in *Proc. 13th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 6917, B. Preneel and T. Takagi, Eds. Nara, Japan: Springer, Sep./Oct. 2011, pp. 49–62.
[14] A. Moradi, O. Mischke, C. Paar, Y. Li, K. Ohta, and K. Sakiyama, "On the power of fault sensitivity analysis and collision side-channel attacks in a combined setting," in *Proc. 13th Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 6917, B. Preneel and T. Takagi, Eds. Nara, Japan: Springer, Sep./Oct. 2011, pp. 292–311.
[15] T. Roche and V. Lomné, "Collision-correlation attack against some 1^{st}-order Boolean masking schemes in the context of secure devices," in *Proc. 4th Int. Workshop Constructive Side-Channel Anal. Secure Design (COSADE)* (Lecture Notes in Computer Science), vol. 7864, E. Prouff, Ed. Paris, France: Springer, Mar. 2013, pp. 114–136.
[16] X. Ye, C. Chen, and T. Eisenbarth, "Non-linear collision analysis," in *Proc. 10th Int. Workshop Radio Freq. Identificat., Secur. Privacy Issues (RFIDSec)* (Lecture Notes in Computer Science), vol. 8651, N. Saxena and A. Sadeghi, Eds. Oxford, U.K.: Springer, Jul. 2014, pp. 198–214.
[17] B. Ege, T. Eisenbarth, and L. Batina, "Near collision side channel attacks," in *Proc. 22nd Int. Conf. Sel. Areas Cryptogr.* (Lecture Notes in Computer Science), vol. 9566, O. Dunkelman and L. Keliher, Eds. Sackville, NB, Canada: Springer, Aug. 2015, pp. 277–292.
[18] C. Herbst, E. Oswald, and S. Mangard, "An AES smart card implementation resistant to power analysis attacks," in *Proc. 4th Int. Conf. Appl. Cryptogr. Netw. Secur. (ACNS)* (Lecture Notes in Computer Science), vol. 3989, J. Zhou, M. Yung, and F. Bao, Eds. Singapore: Springer, Jun. 2006, pp. 239–252.
[19] A. Moradi, S. Guilley, and A. Heuser, "Detecting hidden leakages," in *Proc. 12th Int. Conf. Appl. Cryptogr. Netw. Secur. (ACNS)* (Lecture Notes in Computer Science), vol. 8479, I. Boureanu, P. Owesarski, and S. Vaudenay, Eds. Lausanne, Switzerland: Springer, Jun. 2014, pp. 324–342.
[20] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm, "Analysis and improvements of the DPA contest v4 implementation," in *Proc. 4th Int. Conf. Secur., Privacy, Appl. Cryptogr. Eng. (SPACE)* (Lecture Notes in Computer Science), vol. 8804, R. S. Chakraborty, V. Matyas, and P. Schaumont, Eds. Pune, India: Springer, Oct. 2014, pp. 201–218.
[21] C. Clavier, Q. Isorez, and A. Wurcker, "Complete SCARE of aes-like block ciphers by chosen plaintext collision power analysis," in *Proc. 14th Int. Conf. Cryptol. India* (Lecture Notes in Computer Science), vol. 8250, G. Paul and S. Vaudenay, Eds. Mumbai, India: Springer, Dec. 2013, pp. 116–135.

YONGCHUAN NIU received the M.S. degree in information security from Shandong University, in 2013. He is currently with the Data Communication Science and Technology Research Institute, Beijing, China. His main research interests include side-channel analysis and cryptographic engineering.

JIAWEI ZHANG received the M.S. degree in control theory and control engineering from Peking University, in 2012. She is currently with the Data Communication Science and Technology Research Institute, Beijing, China. Her main research interests include side-channel analysis, machine learning, embedded systems, and biomimetic robotics.

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



CAISEN CHEN was born in 1983. He received the master's degree in computer software and theory, and the Ph.D. degree in network security technology from the Ordnance Engineering College, in 2009 and 2011, respectively. He is currently a Lecturer with the Academy of Army Armored Force, Beijing. His current research interests include information security and implementation attack analysis on cryptosystems. He also researches the security of mobile device.