PAPER *Special Section on Cryptography and Information Security*

Mixture-Based 5-Round Physical Attack against AES: Attack Proposal and Noise Evaluation

Go TAKAMI[†a)], ***Nonmember***, **Takeshi SUGAWARA**[†], **Kazuo SAKIYAMA**[†], ***and*** **Yang LI**[†b)], ***Members***

SUMMARY Physical attacks against cryptographic devices and their countermeasures have been studied for over a decade. Physical attacks on block-cipher algorithms usually target a few rounds near the input or the output of cryptographic algorithms. Therefore, in order to reduce the implementation cost or increase the performance, countermeasures tend to be applied to the rounds that can be targeted by physical attacks. For example, for AES, the conventional physical attacks have practical complexity when the target leakage is as deep as 4 rounds. In general, the deeper rounds are targeted, the greater the cost required for attackers. In this paper, we focus on the physical attack that uses the leakage as deep as 5 rounds. Specifically, we consider the recently proposed 5-round mixture differential cryptanalysis, which is not physical attack, into the physical attack scenarios, and propose the corresponding physical attack. The proposed attack can break AES-128 with data complexity and time complexity of $2^{25.31}$. As a result, it is clear that the rounds as deep as 5 must be protected for AES. Furthermore, we evaluated the proposed attack when the information extracted from side-channel leakage contains noise. In the means of theoretical analysis and simulated attacks, the relationship between the accuracy of information leakage and the complexity of the attack is evaluated.

key words: *AES, physical attack, cryptanalysis, side-channel attack*

1. Introduction

Over a decade, non-invasive physical attacks such as side-channel attacks and fault attacks become a security threat against cryptographic devices and their countermeasures. For example, Correlation Power Attack (CPA) is an efficient side-channel attack technique [1]. Also, Differential Fault Analysis (DFA) is one of the well-known examples of fault attacks [2], [3]. In these physical attacks, attackers usually target a few rounds near the public data of the cryptographic algorithms, i.e. the plaintext and the ciphertext. The information obtained for the intermediate values near the public data can be connected to the secret key with fewer computations. Therefore, partial calculation and fast evaluation of key guesses can be achieved.

In order to protect cryptographic devices from these physical attacks, applying countermeasures to all rounds of ciphers is usually desired. However, in practice, it might be too costly since the countermeasures introduce large overhead and performance decrease. In order to achieve a trade-off between the security and the efficiency, certain implementations would apply countermeasures to only related rounds. Therefore, it is important to confirm the possibility of physical attacks that can target deep rounds of cryptographic algorithms. Once these deep-round attacks are shown to be practical in both attack scenarios and the key recovery complexity, it becomes clear that the related rounds should be protected.

Take AES as an example, the first three and the last three rounds are often the targets considering the known state-of-the-art physical attacks. A few attacks that can target the 4th round and the 6th round are also proposed. The attack discussed in [4] that targets the 4th round requires data complexity of 2^{20}, and time complexity of 2^{27}. In [5], the proposed fault attack targets the 6th round and requires 10 fault injections and time complexity of 2^{40}. To the best of our knowledge, there is no physical attack that has less than 2^{32} complexity targeting the 5th round. In [6], MultiSet Collision attack has been proposed that requires 2^{32} measurements and time complexity of $2^{44.5}$. Such high-cost physical attacks were mainly of theoretical interest because the complexity is too high.

However, recently the complexity of mixture-based cryptanalysis for 5-round AES has been improved to $2^{22.25}$ [7]. Inspired by the 5-round cryptanalysis, this paper focuses on the feasibility of 5-round physical attack against AES. The contributions of this paper are summarized as follows:

- We extend the 5-round mixture-based cryptanalysis to physical attacks. Two attack scenarios are discussed. One is side-channel attack and the other one is fault attack. By comparing the attack assumptions and the complexity, we show that the proposed mixture-based 5-round side-channel attack is superior to the previous deep-round attacks. Compared to the 5-round attack on AES shown in [6], the new attack reduced data complexity from 2^{32} to $2^{25.31}$, and time complexity from $2^{44.5}$ to $2^{24.26}$.
- Since noise always exists in side-channel attacks, the accuracy of obtained data in side-channel attack could not be perfect as that in cryptanalysis. Therefore, we conduct a noise evaluation for the proposed mixture-based side-channel attack. Specifically, we propose a model to describe the data accuracy, and evaluate its relations with the key recovery success rate and the number of false keys.

The rest of this paper is organized as follows. In Sect. 2, we define the notation used in this paper and review several

Manuscript received March 15, 2021.
Manuscript revised July 21, 2021.
Manuscript publicized September 30, 2021.

[†]The authors are with the Department of Informatics, The University of Electro-Communications, Tokyo, 182-8585 Japan.
a) E-mail: takami@uec.ac.jp
b) E-mail: liyang@uec.ac.jp

DOI: 10.1587/transfun.2021CIP0016

deep-round physical attacks. Section 3 reviews the 5-round cryptanalysis that is the attack basis underlying our proposed attacks. In Sect. 4, the proposed physical attacks are explained and compared with previous deep-round attacks. In Sect. 5, we show the noise evaluation of the proposed attacks using theoretical analysis and attack simulations. In Sect. 6, we compare the proposed attack with several variations of classic side-channel attacks. Section 7 concludes this paper.

2. Physical Attacks That Target Deep Rounds of AES

In this section, we first briefly introduce AES and the notations used in this paper. Then, we briefly review a few previous physical attacks that target a deep round of AES such as the 4th or the 5th round.

2.1 AES and Notations

AES is a cryptographic algorithm that encrypts 128-bit plaintext using a secret key of 128, 192, or 256 bits [8]. In this paper, we focus on AES-128 whose key length is 128 bits. AES-128 takes the plaintext and the round key as input and encrypts the plaintext by repeating the round calculation 10 times. In the process of encryption/decryption, the state of the intermediate values are treated as a 4-row 4-column matrix of bytes. In round calculation, four functions are performed once in sequence as follows.

- SubBytes(SB) - Substitute each byte according to a S-box table.
- ShiftRows(SR) - Shift the bytes in the i-th row by i bytes to the left.
- MixColumns(MC) - Multiplication of each column by a constant 4×4 matrix over the field $GF(2^8)$.
- AddRoundKey(ARK) - Calculates the exclusive OR of the intermediate value and the round key.

In this work, we denote the input value to the i-th round as x_i. The intermediate value after SubBytes and ShiftRows in the i-th round is denoted as x_i' and x_i'', respectively. The plaintext is expressed as x_{-1}. The encryption key is k_{-1}, therefore $x_1 = x_{-1} \oplus k_{-1}$. The j-th byte of the intermediate value x_i is expressed as $x_{i,j}$. The multiple l_1-th, l_2-th, ..., l_n-th bytes of the intermediate value x_i is expressed as $x_{i,\{l_1,l_2,\ldots,l_n\}}$. To indicate the entire 32 bits of the j-th column, we use $x_{i,\mathrm{Col}(j)}$. In addition, when referring to bytes in multiple columns, such as the 1st, 2nd, and 3rd columns, we use $x_{i,\mathrm{Col}(1,2,3)}$. We denote the 4 bytes that are j-th column shifted by ShiftRows as $x_{i,SR(\mathrm{Col}(j))}$. Similarly, if we want to focus on the 4 bytes of the j-th column that are shifted by inverse ShiftRows, we use $x_{i,SR^{-1}(\mathrm{Col}(j))}$. Moreover, when we focus on the byte before AddRoundKey, we use $ARK^{-1}(x_i)$. This notation can also be used to represent multiple bytes.

When predicting a key, we consider a quartet of plaintexts and the corresponding quartet of intermediate values. The plaintext quartet is denoted as $(x_{-1}, y_{-1}, z_{-1}, w_{-1})$. The quartet of the intermediate values that are the input to the i-th round is written as (x_i, y_i, z_i, w_i). The value after applying SubBytes to any value α is expressed as $SB(\alpha)$.

2.2 Deep-Round Side-Channel Analysis

In [6], Biryukov et al. showed Impossible Collision Attack and Multi-Set Collision Attack, which target the inner round of masked AES up to 4 rounds. The attack that targets 4th round is based on Impossible Collision Attack. This attack narrows down the wrong key candidates by filtering the ones that cause impossible collisions. Consider a differential pattern in which 4 bytes are active plaintext and the difference propagates as $4 \rightarrow 1 \rightarrow 4 \rightarrow 16$ bytes each time the round process is completed. In this case, at the time of input in the 4th round, the difference propagates over the entire 16 bytes and no collision will occur. However, pairs that do not follow this differential pattern may accidentally collide at any byte. If attackers can detect such byte collisions at the input in the 4th round, the conditions for attacking will be met because there can be no collisions in a right pair. In order to implement this attack, attackers need to feed $2^{19} - 2^{20}$ plaintexts on the target device, and observe the same number of measurements. Also, in order to detect impossible collisions, it is necessary to compare the intermediate pairs 2^{27} times. In this attack, attackers need to detect byte collisions of intermediate values pairs.

The 5-round side-channel attack is based on the Multi-Set Collision Attack [6]. In AES, when a certain plaintext set is used, there is a characteristic that only 3 bytes of collision occur in one column in the input value of the 4th round, which is also a distinguisher of 3-round AES. To apply this as a side-channel attack in the 5th round, first attackers need to feed 2^{32} plaintexts in which $SR^{-1}(\mathrm{Col}(0))$ bytes are active, and observe its measurements in the 5th round. Then, attackers make plaintext sets according to the 0th byte value of measurements. Next, attackers need to predict the subkey at the $SR^{-1}(\mathrm{Col}(0))$ bytes. For each key guess, attackers need to check whether the plaintext set satisfies the property. To restore all subkeys, attackers need 2^{32} measurements and $2^{44.5}$ time. This attack is based on a collision side-channel model which assumes identifying the value of one byte, and collisions of one column of the input of the 5th round. This attack targets relatively deep rounds but has the issue of high complexity.

In [4], Shivam et al. showed the SITM (See-In-The-Middle) attack, which targets deeper rounds of AES and several lightweight ciphers. For AES-128, their attack can go deep up to the 4th round. The main idea of SITM is to select a plaintext pair whose differential pattern converges through a round calculation. In order to detect such a differential pattern, attackers observe the side-channel information in the middle round. Finally, if such a plaintext pair is found, attackers can exhaust the possible values to deduce the set of potential key candidates, which is often much smaller than the entire key space. Targeting the 4th round of AES-128, SITM attackers need to exploit a plaintext pair whose differential pattern converges and diffuses as shown in Fig. 1. Active bytes are in red, blue, or green, where blue bytes in

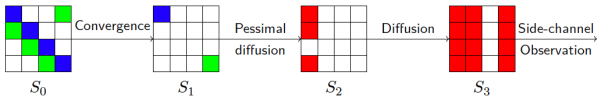


Fig. 1 AES differential pattern for 4-round attack [4].

plaintexts converge to a single blue byte in S_1, similarly for the green bytes. The number of required chosen plaintexts for SITM attack is $2^{27.5}$, memory space to store key candidates is 2^{12}, and time complexity is $O(2^{26.5})$. In SITM attack, side-channel information is used to identify the target difference pattern as shown in Fig. 1.

2.3 Deep-Round Fault Analysis

On the Meet-in-the-Middle fault analysis [5], attackers realize a fault injection on one byte between MixColumns in the 6th round and MixColumns in the 7th round on AES-128. The fault is totally diffused at the whole 10th round. This analysis requires 10 pairs of correct and faulty ciphertexts. If attackers know exactly which byte is faulted, the complexity of the attack is around 2^{40} in time and memory. The overall attack consists of expressing the fault path from the ciphertext to the beginning of the 9th round in the backward direction, and in the forward direction from the fault injection to the beginning of the 9th round. For 10 pairs of correct and faulty ciphertexts, attackers predict 5 bytes key candidates in 2^{40} and decrypt until the beginning of the 9th round. If the difference between the pair is 0 at 0th and 1st bytes, the key candidate is correct. By using the Meet-in-the-Middle attack, attackers can recover the key using 2^{40} memory and time.

Impossible Differential Fault Analysis is based on the fact that it is impossible to have a zero-difference just before MixColumns operation in the 9th round, if one byte fault is injected between MixColumns in the 6th round and MixColumns in the 7th round. Attackers analyze column by column. At first, attackers guess all possible 4 key bytes of the last round subkey. Attackers can filter wrong key candidates using many pairs of the correct and faulty ciphertexts, decrypting each of them with guessed keys. Then attackers repeat four times the no difference computation algorithm for each column before MixColumns operation in the 9th round. If only less than 2^{10} keys quadruples are left, the research can be complemented by exhaustive search. Hence, the time complexity is 2^{40}. If attackers can insert fixed byte fault, 45 correct and faulty ciphertexts pairs are needed.

3. 5-Round Cryptanalysis of AES

3.1 The 4-Round Distinguisher Proposed by Grassi

This section describes the 4-round AES distinguisher by Grassi [9], which is the basis of the 5-round cryptanalysis [7].

Here, we first give a definition of mixture.

Definition. Let the intermediate value pair $(x_i,\ y_i)$ be the input value for the i-th round, such that $x_{i,\mathrm{Col}(1,2,3)} = y_{i,\mathrm{Col}(1,2,3)}$. Similarly, consider the intermediate value pair (z_i, w_i). Then, (z_i, w_i) is a mixture of (x_i, y_i), if for each $j \in \{0,1,2,3\}$, the intermediate value pair $(z_{i,j}, w_{i,j})$ is equal to $(x_{i,j}, y_{i,j})$ or $(y_{i,j}, x_{i,j})$. When (z_i, w_i) is the mixture of (x_i, y_i), (x_i, y_i, z_i, w_i) is a mixture quadruple. Mixture quadruples can be created 7 ways for one intermediate value pair (x_i, y_i).

Algorithm 1 Grassi's 5-round Cryptanalysis

```
1: Encrypt $2^{32}$ plaintexts whose $SR^{-1}(\mathrm{Col}(0))$ assumes all $2^{32}$ possible
   values and the rest of the bytes are constant.
2: Extract a ciphertext pair $(C_1, C_2) = (x_5, y_5)$ whose $SR(\mathrm{Col}(0))$ difference is 0.
3: for $k_{-1,SR^{-1},\mathrm{Col}(0)}$ do
4:     Partially encrypt the corresponding plaintext pair $(x_{-1}, y_{-1})$ and get
       $(x_2, y_2)$.
5:     Let $(z_2, w_2)$ be the mixture of $(x_2, y_2)$ and calculate $(z_{-1}, w_{-1})$ to
       get $(C_3, C_4) = (z_5, w_5)$.
6:     if $z_{5,SR(\mathrm{Col}(0))} \oplus w_{5,SR(\mathrm{Col}(0))} \neq 0$ then
7:         discard the key guess $k_{-1,SR^{-1}(\mathrm{Col}(0))}$.
8:     end if
9: end for
```

For AES and the definition of mixture, we can have the following theorem.

Theorem. For the mixture quadruple (x_i, y_i, z_i, w_i), which is the input to the i-th round of AES, the corresponding intermediate values $(x_{i+2}, y_{i+2}, z_{i+2}, w_{i+2})$ sum up to zero. That is,

$$x_{i+2} \oplus y_{i+2} \oplus z_{i+2} \oplus w_{i+2} = 0. \tag{1}$$

Furthermore, if $x_{i+2,SR^{-1}(\mathrm{Col}(j))} \oplus y_{i+2,SR^{-1}(\mathrm{Col}(j))} = 0$ are satisfied for each $j \in \{0,1,2,3\}$, the corresponding quartet $(x''_{i+3}, y''_{i+3}, z''_{i+3}, w''_{i+3})$ (i.e. input values to MixColumns at $i+3$ round) satisfies

$$x''_{i+3,SR(\mathrm{Col}(j))} \oplus y''_{i+3,SR(\mathrm{Col}(j))} \tag{2}$$
$$= z''_{i+3,SR(\mathrm{Col}(j))} \oplus w''_{i+3,SR(\mathrm{Col}(j))} = 0. \tag{3}$$

The proofs of Eqs. (1), (2), and (3) can be checked from [9]. Grassi proposed a theoretical attack against 5-round AES using this distinguisher [9]. The attack algorithm is shown in Alg. 1.

3.2 Improved 5-Round Cryptanalysis

In this section, we explain cryptanalysis[7] against 5-round AES improved by Bar-On et al. The 5-round cryptanalysis improvement proposed by Bar-on et al. can be divided into four main steps.

1. Data complexity reduction.
   The cryptanalysis proposed by Grassi uses a data volume of 2^{32}. In reality, even if the amount of data is reduced to 2^{24} and the attack is performed, the key can be restored with high probability. However, reducing the amount of data in a straightforward manner increases the time complexity, so the next step is taken.

2. Streamline key prediction.
   If the quartet of intermediate values (x_2, y_2, z_2, w_2) is a mixture, it satisfies $x_2 \oplus y_2 \oplus z_2 \oplus w_2 = 0$, and at the same time it satisfies $x''_1 \oplus y''_1 \oplus z''_1 \oplus w''_1 = 0$. From this, it is possible to perform key predictions more efficiently.
3. Use of pre-calculated table.
   Save time complexity further by pre-storing the required calculation results instead of performing the calculation for all quartets.
4. Improvement of selected plaintext and cost reduction.
   So far, they have considered plaintext sets where SR^{-1}(Col(0)) bytes are active and the other bytes are constant. Restoring $k_{-1,\{5,10,15\}}$ using a plaintext set whose $x_{-1,\{5,10,15\}}$ are active is more efficient than recovering a 4-byte key using this 4-byte active plaintext set.

3.3 Trade-Off Between Data Complexity and Memory

Up to now, Bar-On et al. have considered searching for a ciphertext pair in which the difference of SR(Col(0)) is 0, and proceed with the attack who has a data complexity of $2^{22.25}$, a time complexity of $2^{22.5}$, and a memory complexity of 2^{20}. This level of complexity is very attractive. This is because the complexity of cryptanalysis for 5 round AES has not been less than 2^{32} for about 20 years. Compared to that, this is a significant improvement. However, by searching for ciphertext pairs with a difference of SR(Col(1,2,3)) is 0 and using them in an attack similarly, it is possible to increase the number of ciphertext pairs with a difference of 0 and increase the probability of successful attacks. Therefore, the data complexity used for attacks can be further reduced. However, since the number of stored ciphertext pairs increases, the memory used also increases.

4. Application to Physical Attack

So far, we have explained mixture-based 5-round cryptanalysis by Bar-On et al. In this section, we discuss two approaches of converting this mixture-based cryptanalysis into the physical attacks. The first approach applies the cryptanalysis in the first 5 rounds as side-channel attack, and the other one applies the cryptanalysis in the last 5 rounds as fault attack.

Note that not all cryptanalysis can be converted to physical attacks. For example, the best cryptanalysis on 5-round AES by far is the retracing boomerang attack presented by Dunkelman [10], in which the attack complexity is $2^{16.5}$. However, this attack is difficult to be used in a physical attack scenario. Since this cryptanalysis is based on an adaptive chosen plaintext and ciphertext model, so if attackers try to use this cryptanalysis in a physical attack scenario, they need both side-channel analysis and fault injection, which is difficult.

4.1 Mixture-Based 5-Round Side-Channel Attack

In this section, we show an outline of the mixture-based 5-round side-channel attack including the capabilities required for attackers. Firstly, we try to apply 5-round cryptanalysis to the former 5-round of 10-round AES. x'_5 is regarded as the ciphertext in cryptanalysis. In this case, the characteristic $2^{22.25}$ plaintext required for this attack can be entered as it is, but extracting the pair whose difference of SR(Col(0)) is 0 in the output of the 5th round SubBytes is generally difficult. Therefore, we can consider the method of physical attack. Specifically, prepare a table T_c of size $(2^8)^4$ that can store elements of 2^{32}. Next, in the SubBytes of the 5th round, trace the 4-byte value of SR(Col(0)) from the power consumption, store 4-byte value as an index, and the corresponding plaintext as the value in T_c. If there are elements with duplicate indexes during T_c, they can be regarded as a pair with a difference of 0.

Algorithm 2 Attack Flow of Mixture-based Side-channel Attack

```
1:  Initialize table T.
2:  for $a < b < c$ do
3:      Store $\hat{k}$ which satisfy $SB(\hat{k}) \oplus SB(a \oplus \hat{k}) \oplus SB(b \oplus \hat{k}) \oplus SB(c \oplus \hat{k}) = 0$ to $T[a, b, c]$.
4:  end for
5:  Initialize table $T_c$.
6:  for each $2^{22.25}$ plaintexts do
7:      Encrypt a plaintext whose $x_{-1,\{5,10,15\}}$ are active and other bytes are constant.
8:      Trace a 4-byte value of $SR$(Col(0)).
9:      Store a 4-byte value as an index and the corresponding plaintext as a value in $T_c$.
10: end for
11: for pairs of $T_c$ elements with duplicate indexes do
12:     Let the corresponding plaintexts quartet $(x_{-1}, y_{-1}, z_{-1}, w_{-1})$.
13:     Let $(a, b, c)$ be the sorted values calculated $(y_{-1,5} \oplus x_{-1,5}, z_{-1,5} \oplus x_{-1,5}, w_{-1,5} \oplus x_{-1,5})$.
14:     for $\hat{k} = T[a, b, c]$ do
15:         Store $k_{-1,5} = \hat{k} \oplus x_{-1,5}$ to a list $L_5$.
16:     end for
17:     Repeat (13) - (16) for 10th, and 15th bytes.
18:     for All key guesses $(k_{-1,5} \in L_5, k_{-1,10} \in L_{10}, k_{-1,15} \in L_{15})$ do
19:         Calculate ARK, SB, SR, MC of $(x_{-1}, y_{-1}, z_{-1}, w_{-1})$ and obtain $(x_2, y_2, z_2, w_2)$.
20:         if $(x_2, y_2, z_2, w_2)$ is a mixture quartet then
21:             Remain the key candidate.
22:         end if
23:     end for
24: end for
25: Output all remain key candidates $k_{-1,5}, k_{-1,10}, k_{-1,15}$.
```

Therefore, the attack flow is shown in Alg. 2. If attackers can confirm whether or not the difference of SR(Col(0)) in the intermediate value pair is 0, the requirements for this attack are met. For example, such attackers can measure the power consumption of the 5th round SubBytes and can identify the zero difference of the intermediate value pair. Specifically, after encrypting $2^{22.25}$ plaintexts, $(((2^{22.25})^2)/2) \cdot 2^{-32} = 2^{11.5}$ intermediate value pairs for which the difference of SR(Col(0)) is 0 are required to be extracted.

Figure 2 and Alg. 2 show the method of checking the difference of only SR(Col(0)). However, by checking all columns, it is possible to increase the probability of suc-

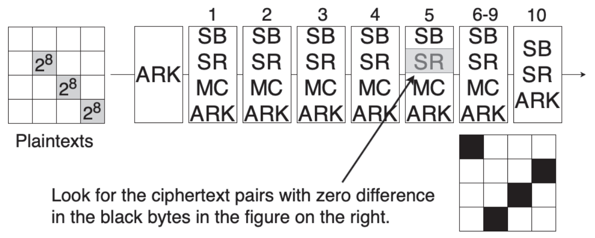

Plaintexts
2^8
2^8
2^8
ARK
1
2
3
4
5
6-9
10
SB
SR
MC
ARK
Look for the ciphertext pairs with zero difference in the black bytes in the figure on the right.

Fig. 2 Schematic of the attack on the first five rounds.

cessful key recovery and reduce the data complexity of the attack.

4.1.1 Attack Complexity

So far, we have considered an attack that restores the secret key of 3 bytes by inputting $2^{22.25}$ plaintexts whose $x_{-1,\{5,10,15\}}$ bytes are active. In this section, we consider the complexity required to fully recover the key for AES-128. Here, the time complexity is calculated based on single encryption of AES-128. When targeting key bytes of $k_{-1,\{5,10,15\}}$, the plaintexts whose $x_{-1,\{5,10,15\}}$ bytes are active are used. Similarly, in order to recover key bytes of $k_{-1,\{3,9,14\}}$, the plaintexts whose $x_{-1,\{3,9,14\}}$ bytes are active should be used. The same attacks are repeated for SR^{-1}(Col(2,3)). Then, attackers can recover $k_{-1,\{0,5,10\}}$ key bytes using the same approach. Up to this point, 13 key bytes have been recovered. Finally, attackers can conduct a brute force recovery for the remaining 3 keys bytes for a full key recovery.

For the 3 bytes key recovery using $2^{22.25}$ plaintexts, the data complexity is $2^{22.25}$. In this case, memory needs $2 \times 2^{32} = 2^{33}$. This is because when searching for an intermediate value pair with a difference of 0, the intermediate value pair is stored in the T_c for each 4-byte value. Two T_c tables are needed because some elements may duplicate. As for the time complexity, first attackers need to encrypt $2^{22.25}$ in the first step of the attack. Next, we consider the time complexity required for the key recovery after encrypting plaintexts. The total number of all possible intermediate value pairs extracted in the 5th round is $(2^{22.25})^2/2 = 2^{43.5}$. Also, the probability that the difference of SR(Col(0)) is zero for any intermediate value pair is 2^{-32}. Therefore, the number of pairs with zero difference is $2^{43.5} \cdot 2^{-32} = 2^{11.5}$, and the number of quartets is $(2^{11.5})^2/2 = 2^{22}$. In the step of predicting the key from quartets, they are only encrypted one round, so the time in this step is $2^{22}/10 \approx 2^{18.68}$. From the above, the time required for 3-byte key recovery is $2^{22.25}/2 + 2^{18.68} \approx 2^{21.47}$.

Attackers can repeat the above attack 5 times to recover the 13 key bytes. When restoring the key of the remaining 3 bytes, brute force is performed, then data complexity required for this is 2^{24}, and time complexity is 2^{24}. From the above, in order to recover the full-byte key, the required data is $5 \cdot 2^{22.25} \approx 2^{24.57}$, memory is $5 \cdot 2^{33} \approx 2^{35.32}$, and time is $5 \cdot 2^{21.67} + 2^{24} = 2^{24.26}$.

Table 1 Comparison with previous works.

Method	Data	Memory	AES-128 cost	Attack depth
SITM [4]	$2^{27.5}$	2^{12}	$O(2^{26.5})$	4
IMP [6]	$2^{19} - 2^{20}$		2^{27}	4
MULT [6]	2^{32}		$2^{44.5}$	5
Our work	$2^{25.31}$	$2^{35.32}$	$2^{24.26}$	5

4.1.2 Comparison with Previous Works

In this section, we compare our attack with the previous side-channel attacks that target deep rounds of AES. Table 1 shows a comparison of complexity required for our side-channel attack with previous studies.

Compared to these previous works, the proposed attack is better at data and time complexity, even though we are targeting a deeper round. In side-channel attacks, it's worth keeping data complexity down. The data complexity is large compared to the 4-round Impossible Collision Attack, but it is difficult to make a direct comparison because our attack is aimed at a deeper round than this attack. Our data complexity is small when compared to the 4-round SITM and the 5-round MultiSet Collision Attack.

Regarding time complexity, it's better than an attack targeting 4 rounds. Concerning the complexity of memory, our attack is larger than that of previous works, but memory space can be satisfied easier than data and time, and $2^{35.32}$ memory complexity is still realistic.

4.2 Mixture-Based 5-Round Fault Attack

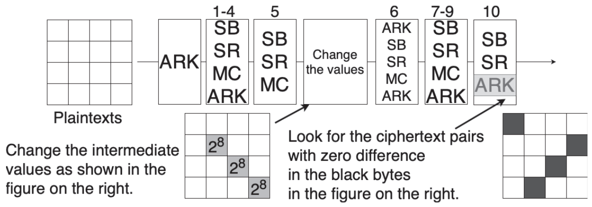
In this section, we show an outline of the mixture-based 5-round fault attack including the capabilities required for attackers.

Attackers need to change the intermediate value after MixColumns in the 5th round so that only the $ARK^{-1}(x_{6,\{5,10,15\}})$ bytes are active, and obtain $2^{22.25}$ ciphertexts. By applying our 5-round cryptanalysis, it is theoretically possible to recover the subkey as shown in Fig. 3. The attack flow differs from Alg. 2 in the following ways:

- Step 7: Input arbitrary plain text.
- Step 8: After calculating the 5th round, change the values of the $x_{-1,\{5,10,15\}}$ bytes to active and the other bytes to constant.
- Step 9: Trace a 4-byte value of SR(Col(0)) of ciphertext, and store it as an index and the corresponding plaintext as a value in T_c.

If attackers can change the $x_{-1,\{5,10,15\}}$ bytes after MixColumns to be active and all the other bytes to be constant, the condition that the analysis can be applied is set.

For that purpose, attackers can take a physical means to change the value by provoking faults in bit level. It is necessary to change $2^{22.25}$ intermediate values as described above. Similar to the Sect. 4.1, by checking more columns, it is possible to reduce the data complexity. In order to recover



1-4
5
6
7-9
10
ARK
SB SR MC ARK
SB SR MC
Change the values
ARK SB SR MC ARK
SB SR MC ARK
SB SR ARK
Plaintexts
Change the intermediate values as shown in the figure on the right.
2^8
2^8
2^8
Look for the ciphertext pairs with zero difference in the black bytes in the figure on the right.

Fig. 3 Schematic of the attack on the latter five rounds.

Table 2 Comparison with previous works.

Method	# of faults	AES-128 cost	Attack depth
MITM [5]	10	2^{40}	4
IMP [5]	45	2^{40}	4
Our work	$2^{24.57}$	$2^{24.26}$	5

full key-byte, attackers need to repeat the above steps five times and conduct brute force.

4.2.1 Attack Complexity

We have considered an attack that restores the private key of 3 bytes by injecting faults and making $x_{-1,\{5,10,15\}}$ bytes active. In this section, we consider the complexity required to fully recover the key, like that of the former 5-round. The attack flow is similar as in Sect. 4.1, except for the injection of faults.

First, we review the complexity required for 3-byte key recovery. Attackers need $2^{22.25}$ faults. Similar to the attack on the former rounds, 2^{33} memory is required, and time complexity is $2^{21.47}$.

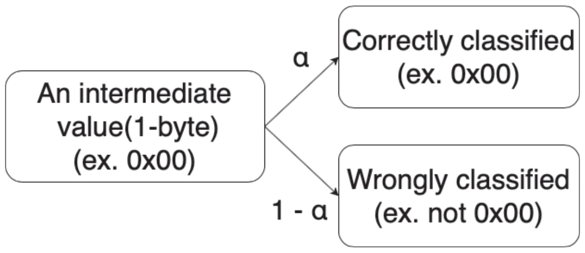
Next, we review the complexity for full-byte key recovery. The required memory and time complexity are the same as Sect. 4.1.1. Therefore, the required memory is $2^{35.32}$, and time complexity is $2^{24.26}$. In addition, the required number of faults is $5 \cdot 2^{22.25} \approx 2^{24.57}$.

4.2.2 Comparison with Previous Works

In this section, we compare our attack model with previous fault attacks which target deeper rounds. In [5], they showed Meet-in-the-Middle and Impossible Differential Fault Analysis. If attackers can insert fixed byte fault, 45 correct and faulty ciphertexts pairs are needed. Our attacks are inferior in the number of faults required when compared to these previous studies. However, as far as we know, we have never seen a fault analysis that puts a fault in the 5th round. So this attack is mainly of theoretical interest because it requires too many fault injections and an impractical fault model.

5. Impact of Noise for Mixture-Based 5-Round SCA

We showed that Mixture-Based 5-Round SCA is better than previous works. In the following, we focus on the method shown in Sect. 4.1 from the reality of attacks. Assuming a situation in which the difference of SR(Col(0,1,2,3)) is identified from the power consumption, noise is included in it, and the attack is less likely to succeed compared to the ideal state. In this section, we model the effect of noise generated in the power consumption on the difference judgement and evaluate the possibility of successful attacks from the theoretical and experimental aspects. Also, we simulated the output false keys and considered their effect on the attack.



An intermediate value(1-byte) (ex. 0x00)
α
1 - α
Correctly classified (ex. 0x00)
Wrongly classified (ex. not 0x00)

Fig. 4 Intermediate values classification model.

5.1 Proposal of Difference Judgement Probabilistic Model

Attackers classify the intermediate values for each value and regard the duplicate elements as intermediate value pairs with a difference of 0. Here, the probability that an arbitrary 1-byte intermediate value is correctly classified from the consumption is defined as α. For example, this is the probability that a trace is classified as 0x00 when the value at the 0th byte is 0x00. Then, the probability that the zero difference of a pair of bytes is 2^{-8}. Therefore, when we focus on 1 byte of a pair of intermediate values, the probability that this pair of bytes has a zero difference also their corresponding power traces are both correctly classified into the correct values is $\alpha \cdot \alpha \cdot 2^{-8} = \alpha^2 \cdot 2^{-8}$. The value of α becomes smaller as noise gets in and the values cannot be classified correctly from the traces.

Attackers advance this attack by setting the intermediate value pair in the upper right of Fig. 4. That is, the duplicated pairs of the values classified from the traces, as the intermediate value pair with zero difference. The following problems can be predicted from the difference judgement probabilistic model for the intermediate value pairs used in the attack.

- In the ideal state, the value of α in Fig. 4 is 1. However, in actual physical attacks, the value of α becomes smaller than 1 due to the influence of noise, and the amount of intermediate value pairs with a difference of 0 extracted by attackers may be reduced. From Sect. 3.1, the probability that a theoretical attack of 5 rounds will succeed depends on the amount of intermediate value pairs whose difference is 0, so it can be expected that the probability of a successful attack will be lower.
- In an actual physical attack, the value of α becomes smaller than 1 due to the influence of noise, and there is a possibility that the difference in the intermediate value pairs extracted by attackers are not 0. The condition of the key output in the 5-round cryptanalysis was that

it consisted of ciphertext pairs with a difference of 0, and that there were one or more mixture quadruples. It can be expected that false keys will be output if an intermediate value pair with a non-difference of 0 is used in the attack and the mixture property is satisfied when the quadruplet is created.

5.2 Theoretical Evaluation

In Sect. 5.1, we qualitatively showed two types of effects of noise on attacks. In this section, we evaluate theoretically the relationship between the probability α and the attack success probability, and the number of output false keys. In the following, the physical attack using the 5-round cryptanalysis is performed once, and the case where one or more 3-byte correct keys are output is defined as a successful attack.

5.2.1 Success Rate

We consider the attack is success if the correct key exists in the attack result. The probability of success attack is the success rate. As the condition that the correct key is inside the attack result, there should be at least one mixture quadruple whose 4-byte differences specified by the intermediate value pairs are 0.

From an arbitrary quartet $(x_{-1}, y_{-1}, z_{-1}, w_{-1})$, the correct key will be output if the following two conditions are satisfied. First, (x_2, y_2, z_2, w_2) must be a mixture. This corresponds to Eq. (1). Second, the difference of any two values from $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))}, z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ must be 0. This corresponds to Eqs. (2) and (3).

Hereafter, we derive the success rate in the following way. First, we calculate the number of quartets that can satisfy the mixture property. Let the number of plaintexts that attackers feed be 2^d. As calculated in Sect. 4.1 of [7], the number of quartets is 2^{4d-3}. Also, the probability that an arbitrary quartet satisfies the mixture property is $3 \cdot 2^{-55}$. Therefore, the number of quartets that can satisfy the mixture property is $3 \cdot 2^{-55} \cdot 2^{4d-3} = 3 \cdot 2^{4d-58}$.

Second, we seek the probability of zero difference for a mixture quadruple. Specifically, $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))})$ and $(z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ should have zero difference, which corresponds to Eqs. (2) and (3).

For a quartet (x_5, y_5, z_5, w_5), the probability that $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))})$ has zero difference is 2^{-32}. If the difference of $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))})$ is 0, the rest pair $(z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ also has zero difference. This can be confirmed from Eq. (1) as shown in Sect. 3.1. Therefore, the probability that the sum of $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))}, z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ has zero difference is 2^{-32}.

Next, we consider the scenario when noise exists when the attackers look for intermediate value pairs that satisfy Eqs. (2) and (3). According to the accuracy α as defined in Sect. 5.1, the probability that the 2 bytes of $(x_{5,0}, y_{5,0})$ are correctly classified is α^2. Also, the probability that the $4 \times 2 =$ 8 bytes of $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))})$ are correctly classified is α^8. Therefore, the probability that the $4 \times 4 = 16$ bytes of $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))})$, $(z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ are correctly classified is α^{16}.

From the above, the probability that the difference of $(x_{5,SR(\mathrm{Col}(0))}, y_{5,SR(\mathrm{Col}(0))}, z_{5,SR(\mathrm{Col}(0))}, w_{5,SR(\mathrm{Col}(0))})$ is 0 and attackers can correctly identify the intermediate values is $2^{-32} \cdot \alpha^{16}$.

By far, the success rate can be calculated as the complementary event that the differences in $SR(\mathrm{Col}(0))$ among all the mixture quadruples all have non-zero differences. That is,

$$1 - (1 - 2^{-32} \cdot \alpha^{16})^{3 \cdot 2^{4d-58}}. \tag{4}$$

In addition, when the pairs of intermediate values for which the difference is 0 are examined not only for $SR(\mathrm{Col}(0))$ but also for $SR(\mathrm{Col}(1,2,3))$, the intermediate value pairs are obtained four times, so the success rate becomes

$$1 - (1 - 2^{-32} \cdot \alpha^{16})^{4 \cdot 3 \cdot 2^{4d-58}}. \tag{5}$$

5.3 Attack Simulations

We implemented a 5-round physical attack simulation considering the probability α in the differential judgement probabilistic model, and evaluated the success rate.

5.3.1 Simulation Setup

Algorithm 2 is implemented using C language to simulate the key recovery attack. In the simulations, we only consider the attack that examines all 4 columns with 0 difference. This is because the number of quartets used for the attack increases and the probability of successful attacks improves when many columns with 0 difference are examined, even if the complexity of data is the same.

For each specific α, we repeat the key recovery trials 2400 times. For each key recovery trial, we use $2^{22.25}$ randomly generated plaintexts. The main focus of the experiments is to verify the success rate and the number of false keys under different α.

As for the simulation machine, we use a PC that has the Intel(R) Xeon(R) CPU E5-2687W working at 3.00 GHz and 256 GB RAM.

5.3.2 Success Rate

As for the success rate in our attack, it means the probability of recovering at least one correct key. The result of simulated success rate and the theoretical success rate is shown in Table 3 as well as Fig. 5. The theoretical success rate is extracted using Eq. (5). In Table 3, α means the accuracy of byte-value identification, successes means the number that one or more correct keys are output out of 2400 key recovery trials. The former is calculated from simulation results, and the latter is calculated from Eq. (5). Figure 5 shows a graph

Table 3 Success rate with 3 active bytes.

α	Successes	Success rate of simulation	Success rate of theory
1	2396	0.998	0.998
0.95	2183	0.910	0.929
0.90	1509	0.629	0.671
0.85	806	0.340	0.360
0.80	323	0.135	0.155
0.75	132	0.055	0.058
0.70	40	0.023	0.020
0.60	5	0.002	0.001
0.50	1	0.000	0.000
0.40	0	0	0

Table 4 Requirements to keep the success rate of above 0.9 with three active bytes.

α	# of x_{-1}(3-byte recovery)	# of x_{-1}(full-byte recovery)
1	$2^{21.91}$	$2^{24.23}$
0.9	$2^{22.53}$	$2^{24.85}$
0.8	$2^{23.20}$	$2^{25.52}$
0.7	$2^{23.97}$	$2^{26.29}$
0.6	$2^{24.86}$	$2^{27.18}$
0.5	$2^{25.91}$	$2^{28.23}$
0.4	$2^{27.16}$	$2^{29.48}$
0.3	$2^{28.86}$	$2^{31.18}$
0.2	$2^{31.20}$	$2^{33.52}$
0.1	$2^{35.20}$	$2^{37.52}$

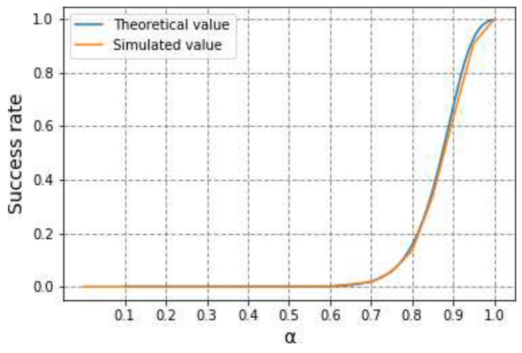


Fig. 5 Success rate with 3 active bytes.

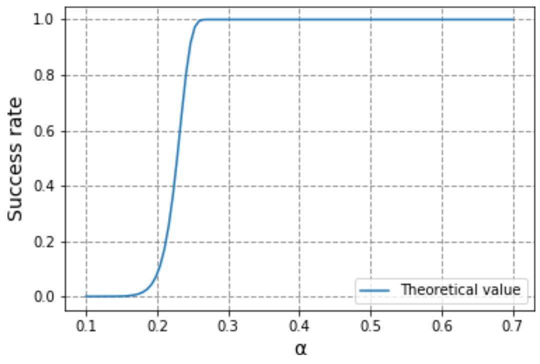


Fig. 6 Success rate with 4 active bytes.

of the probability of outputting one or more correct keys, corresponding to α, when attacked once with a plaintext of $2^{22.25}$. The light blue line is the theoretical values calculated from Eq. (5) and the orange line is the simulated values.

From the similarity between the theoretical value and the simulation result, we can confirm the validity of the theoretical analysis of the success rate.

From Fig. 5, we can see that the attack success probability is stable when α is larger than 0.95, but it decreases exponentially if it is smaller than 0.95. As explained by Eq. (5), in order to increase the success rate, attackers can use more plaintexts. For example, to keep the success rate above 0.9, the accuracy of byte-value identification α and the required size of plaintext is shown in Table 4. This table shows the number of plaintexts required for each value of α to achieve a 0.9 probability of outputting at least one correct key in a single key recovery. The column of number of x_{-1}(3-byte recovery) means the number of required plaintext number when attackers recover the key of 24-bit under accuracy α, and the column of number of x_{-1}(full-byte recovery) is the number of plaintexts when recovering full-bit key. From Table 4, we can see that if α is small, the success rate can be kept by increasing the number of plaintexts.

Since only 3 bytes are active in the improved mixture-based attack when attack 3-byte of key, when α is larger than 0.7, we can expect a success rate of 90% when using all the active plaintexts. If we extend the active byte into 4 bytes, then there is still a possibility to recover the key with the use of more plaintexts. In this case, the probability that the intermediate values (x_2, y_2, z_2, w_2) form a mixture quadruples decreases from $3 \cdot 2^{-55}$ to $3 \cdot 2^{-63}$ according to Sect. 4.1 in [7]. Based on this fact, the exponent of Eq. (5) becomes $4 \cdot 3 \cdot 2^{4d-66}$. Figure 6 shows a graph of the probability of outputting one or more correct keys, corresponding to α, when attacked once with a plaintext of 2^{32} whose 4 bytes of SR^{-1}(Col(0)) are active. From Fig. 6, we can see that if α is less than 0.25, the probability of success decreases, and the attack becomes more difficult. Since data complexity cannot be larger than 2^{32}, α should be more than 0.25 at least. From the above, we can see that if attackers demand a high success rate, α should be more than 0.7 for the proposed attack procedure, and under the condition that data complexity can be increased to the maximum, α should be more than 0.25.

The accuracy of the classification of the intermediate value could be difficult to be high for many implementations on general platforms. However, along with the research progress of deep-learning based side-channel attacks, the classification for implementations without countermeasures is likely to become more accurate. In such attacks, the intermediate values are classified by the power consumption and the pre-trained model. In certain scenarios, the intermediate values are not classified by the Hamming Weight (HW) or the Hamming Distance (HD), but by their values. For example, a byte of intermediate value is classified into one of 256 classes. This type of attack exactly fits the attack scenario discussed in this paper. The accuracy of such classification is corresponding to α defined in this paper.

In [11], Kim et al. used DPA contest v4 dataset [12], which provides the masked AES power consumption and

Table 5 Number of false keys.

α	# of false keys for 2400 trials	Expected # of false key in 1 trial
1	49	0.0204
0.9	50	0.0208
0.8	57	0.0238
0.7	49	0.0204
0.6	39	0.0163
0.5	41	0.0171
0.4	40	0.0167
0.3	43	0.0179
0.2	61	0.0254
0.1	47	0.0196
0	51	0.0213

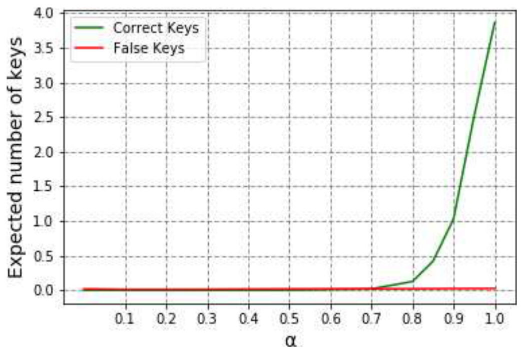


Fig. 7 The expected values of keys.

the secret keys are fixed. The mask is known and thus they turn the implementation into an unprotected scenario. Kim et al. performed 10 attack experiments on four data sets including DPA contest v4 for each of the number of traces used for training, the number of epochs, and the noise added. The amount of noise added to the learning trace is used to prevent over-fitting. In their result of Sect. 5.3 in [11], the accuracy of the predicted intermediate value is as high as 0.95 when the number of learning traces is 80000, the noise level is 0.25, and the number of epochs is 100 or more. This result shows that the proposed attack has certain practical significance although it costs much larger complexity than normal side-channel attacks.

5.3.3 Number of False Keys

In the key recovery process, it is possible that a key satisfies the mixture property for the calculated quartet is not the correct key. We call these keys the false keys. Hereafter, we will evaluate false keys and show that they appear rarely in the key recovery. Furthermore, the number of false keys is not affected α, since the false keys are derived from the non-mixture quartets whose total number is not affected by α. The higher α is, the more mixture quadruples can be used to recover the key, but the number of non-mixture quadruples is too large to be affected by α. The number of false keys using the simulations that examine 4 columns is shown in Table 5. Same with the previous simulations, we repeat the key recovery trials for 2400 times. In Table 5, the second column means the total number of false keys that were output during the 2400 iterations of the key recovery, and the third column means the expected number of false keys when attackers try this attack once.

Figure 7 shows the comparison of expected values of correct and false keys. The green graph is the expected value of correct keys, and the red graph is the expected value of false keys. As shown in Fig. 7, it can be seen that the expected values of false keys are much smaller than the expected value of correct keys. Therefore, even if more than one type of key candidate is output, the number is limited. Therefore, it is possible to fully search for the correctness of each key candidate and find the correct key with a realistic amount of comparison. For example, when the attack is repeated 2,400 times with the accuracy $\alpha = 0.5$, one correct key and 50 false keys are expected to be output from Table 5. Since the number of false keys is sufficiently small, it is possible to perform a full search with a practical amount of computation to identify the correct key. From the above, it can be concluded that false keys have little impact on the proposed attack.

6. Comparison with SCA Variations

In this section, we compare the proposed mixture-based side-channel attack with several related SCA variations such as blind SCA [14]–[16], algebraic SCA [17]–[19], soft analytic SCA [20], [21], and cube side-channel attacks[22] for the difference in the attack requirements and the attack characteristics.

Classic side-channel attacks such as DPA [13] and CPA [1] usually target the beginning or the finish of the cryptographic algorithm. The public information such as plaintext or ciphertext is used in the key recovery together with the leakage from a near intermediate value. Using the noise-free public information simplifies the key recovery. A near intermediate value keeps the involved cryptographic calculation simple to allow easy divide-and-conquer. For each leakage trace of cryptographic calculation, only the part related to the targeted cryptographic calculation can be used to recover the key. Generally, classic SCA have a good noise robustness but a high data complexity. Several newly proposed side-channel attacks put forward different ideas to improve key recovery efficiency or feasibility.

For example, blind SCA does not use the public information. Instead, blind SCA uses side-channel leakage from multiple intermediate values, e.g. a and $SB(a \oplus k)$, to recover the key. Blind SCA does not care about the specific location of the detailed leakage but is more sensitive to the noise of leakage. Blind SCA also keep the calculation between two intermediate values to be simple to achieve a low key recovery complexity under the noisy side-channel information. In comparison, the mixture-based SCA uses noise-free public information. However, the related cryptographic calculation is much more for the mixture-based SCA. Thus, the mixture-based SCA requires a higher complexity in the key recovery.

Another type of attack is algebraic SCA (ASCA) and soft analytic SCA (SASCA). For ASCA and SASCA, the key recovery problem is described as a set of linear equations or a code problem, respectively. ASCA and SASCA are proposed to reduce the data complexity of the classic SCA by using more information in the complete trace to recover the key. Under certain conditions, both ASCA and SASCA can extract the key of an AES implementation from a single leakage trace, in an unknown plaintext/ciphertext scenario. On the other hand, cube SCA combines cube cryptanalysis and side-channel attack. The leakage in cube attack is assumed to be one bit of information about the intermediate state of the encryption after a few rounds. The leakage bit can be represented by low degree multivariate polynomials, therefore cube cryptanalysis techniques are used to recover the keys. ASCA, SASCA, and cube SCA tend to use more relations among the leakage from multiple intermediate values to reduce the number of required traces for the key recovery. For these attacks, the key recovery still depends on the leakage from the intermediate values near the public data. Different from them, the mixture-based SCA uses only the leakage in a deep round, which demonstrates the vulnerability even when the leakage near the public data is not available.

7. Conclusion

In this paper, we examined the feasibility of physical attack that only uses the leakage from the AES round as deep as 5. Based on the mixture-based cryptanalysis of 5-round AES, we proposed the corresponding physical attacks. We showed that the proposed mixture-based 5-round side-channel attack has a much smaller complexity compared to the previous attack. Also, the success rate and the number of false keys are analyzed with theoretical analysis and attack simulations. The results show that the attack is still possible even when the data contains a certain level of noise. As a result of this work, we show that the 5th round of AES-128 should be protected against side-channel attacks.

Acknowledgments

This work has been supported by JSPS KAKENHI Grant Number JP18H05289.

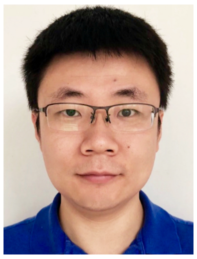
References

[1] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," CHES 2004, LNCS 3156, pp.16–29, 2004.
[2] C. Giraud, "DFA on AES," AES 2004, LNCS 3373, pp.27–41, 2005.
[3] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," CRYPTO 1997, LNCS 1294, pp.513–525, 1997.
[4] S. Bhasin, J. Breier, X. Hou, D. Jap, R. Poussier, and S.M. Sim, "SITM: See-in-the-middle side-channel assisted middle round differential cryptanalysis on SPN block ciphers," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol.2020, no.1, pp.95–122, 2020.
[5] P. Derbez, P. Fouque, and D. Leresteux, "Meet-in-the-middle and impossible differential fault analysis on AES," CHES 2011, LNCS 6917, pp.274–291, 2011.
[6] A. Biryukov and D. Khovratovich, "Two new techniques of side-channel cryptanalysis," CHES 2007, LNCS 4727, pp.195–208, 2007.
[7] A. Bar-On, O. Dunkelman, N. Keller, E. Ronen, and A. Shamir, "Improved key recovery attacks on reduced-round AES with practical data and memory complexities," CRYPTO 2018, LNCS 10992, pp.185–212, 2018.
[8] J. Daemen and V. Rijmen, "AES proposal: Rijndael," AES Algorithm Submission, 1999.
[9] L. Grassi, "Mixture differential cryptanalysis: A new approach to distinguishers and attacks on round-reduced AES," IACR Transactions on Symmetric Cryptology, vol.2018, no.2, pp.133–160, 2018.
[10] O. Dunkelman, N. Keller, E. Ronen, and A. Shamir, "The retracing boomerang attack," EUROCRYPT 2020, LNCS 12105, pp.280–309, 2020.
[11] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic, "Make some noise unleashing the power of convolutional neural networks for profiled side-channel analysis," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol.2019, no.3, pp.148–179, 2019.
[12] TELECOM ParisTech SEN research group, DPA Contest (4-th edition), 2013–2014. http://www.DPAcontest.org/v4/
[13] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," CRYPTO 1999, LNCS 1666, pp.388–397, 1999.
[14] C. Clavier and L. Reynaud, "Improved blind side-channel analysis by exploitation of joint distributions of leakages," CHES 2017, LNCS 10529, pp.24–44, 2017.
[15] Y. Linge, C. Dumas, and S. Lambert-Lacroix, "Using the joint distributions of a cryptographic function in side channel analysis," COSADE 2014, LNCS 8622, pp.199–213, 2014.
[16] C. Clavier, L. Reynaud, and A. Wurcker, "Quadrivariate improved blind side-channel analysis on boolean masked AES," COSADE 2018, LNCS 10815, pp.153–167, 2018.
[17] M. Renauld, F. Standaert, and N. Veyrat-Charvillon, "Algebraic side-channel attacks on the AES: Why time also matters in DPA," CHES 2009, LNCS 5747, pp.97–111, 2009.
[18] Y. Oren, M. Kirschbaum, T. Popp, and A. Wool, "Algebraic side-channel analysis in the presence of errors," CHES 2010, LNCS 6225, pp.428–442, 2010.
[19] M.S.E. Mohamed, S. Bulygin, M. Zohner, A. Heuser, M. Walter, and J. Buchmann, "Improved algebraic side-channel attack on AES," 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, pp.146–151, 2012.
[20] N. Veyrat-Charvillon, B. Gérard, and F. Standaert, "Soft analytical side-channel attacks," ASIACRYPT 2014, LNCS 8873, pp.282–296, 2014.
[21] V. Grosso and F. Standaert, "ASCA, SASCA and DPA with enumeration: Which one beats the other and when?," ASIACRYPT 2015, LNCS 9453, pp.291–312, 2015.
[22] I. Dinur and A. Shamir, "Side channel cube attacks on block ciphers," Cryptology ePrint Archive, Report 2009/127, 2009.

Go Takami was born in Iwate, Japan, in 1998. He received the B.E. degree from The University of Electro-Communications, Tokyo in 2020. He is currently a second-year master's student at The University of Electro-Communications, Tokyo. His research interest includes side-channel attack.

Takeshi Sugawara received Ph.D. from Tohoku University, Sendai in 2011. He joined Mitsubishi Electric Corporation in 2011 and involved in R&D of embedded systems security. He is currently an Associate Professor at The University of Electro-Communications, Tokyo, since 2017. His research interest involves hardware and embedded systems security, lightweight cryptography, and side-channel attack.

Kazuo Sakiyama received the B.E. and M.E. degrees from Osaka University in 1994 and 1996, respectively, the M.S. degree from UCLA in 2003, and the Ph.D. degree in electrical engineering from KU Leuven in 2007. From 1996 to 2004, he was with the Semiconductor and IC Division, Hitachi, Ltd. He has been Professor at The University of Electro-Communications, Tokyo since 2013. He is a member of IACR and IPSJ, and a senior member of IEEE.



Yang Li received Ph.D. from the University of Electro-Communications, Japan, in 2012. From 2013 to 2015, he worked as a research assistant professor at the University of Electro-Communications, Japan. From 2015 to 2018, he worked as an associated professor at Nanjing University of Aeronautics and Astronautics, China. Currently, he is an associated professor at the University of Electro-Communications, Japan. His main research interests include security evaluation for cryptographic hardware and IoT devices.