Journal of Information Processing Vol.22 No.2 142–151 (Apr. 2014)

[DOI: 10.2197/ipsjjip.22.142]

Regular Paper

Practical DFA Strategy for AES Under Limited-access Conditions

Kazuo Sakiyama[1,a) Yang Li[1,b) Shigeto Gomisawa[1] Yu-ichi Hayashi[2] Mitsugu Iwamoto[1] Naofumi Homma[2] Takafumi Aoki[2] Kazuo Ohta[1]

Received: May 13, 2013, Accepted: November 1, 2013

Abstract: Secret data in embedded devices can be revealed by injecting computational faults using the fault analysis attacks. The fault analysis researches on a cryptographic implementation by far first assumed a certain fault model, and then discussed the key recovery method under some assumptions. We note that a new remote-fault injection method has emerged, which is threatening in practice. Due to its limited accessibility to cryptographic devices, the remote-fault injection, however, can only inject uncertain faults. In this surroundings, this paper gives a general strategy of the remote-fault attack on the AES block cipher with a data set of faulty ciphertexts generated by uncertain faults. Our method effectively utilizes all the information from various kinds of faults, which is more realistic than previous researches. As a result, we show that it can provide a decent success probability of key identification even when only a few intended faults are available among 32 millions fault injections.

Keywords: cryptography, advance encryption standard, differential fault analysis, intentional electromagnetic interference, uncertain faults.

1. Introduction

Nowadays, cryptographic functions are widely applied in daily embedded devices such as smart cards and smart phones. The security of these devices depends on the secret information such as an encryption/decryption key stored inside. It is well known that such sensitive information is leaked from erroneous cryptographic calculations that happen when the embedded devices is under intentionally added physical disturbance such as an illegal clock/power supply. These intentionally injected faults can be used by fault attacks for revealing the secret information. Until now, fault tolerance is the main research subject when considering the computational fault of the embedded devices. However, for cryptographic embedded devices, fault-based attacks that are significantly powerful in extracting the secret information need to be considered as well. An updated survey of fault injection attacks on cryptographic devices can be found in Ref. [1]. Differential Fault Analysis (DFA), which was proposed by Biham and Sharmir in 1996 [2], is one of the most discussed fault-based attacks. As shown in Refs. [3], [4], [5], [6], [7], if injected faults fit an attacker's intended fault model, only a few fault injections are enough to identify the secret key in the case of AES (Advanced Encryption Standard) [8].

The conventional fault-injection methods include the invasive type such as the optical emission and the laser shot, and the non-invasive type such as under-powering and over-clocking inputs. The invasive fault injections require damaging the chip by removing the package of the cryptographic module, while the non-invasive ones do not damage the device but require access to the power supply or the clock supply line connected to the target device. In contrast, in 2011, Hayashi et al. demonstrated a series of fault-injection techniques based on ElectroMagnetic (EM) coupling [9], [10], [11], which enables us to perform a fault injection remotely without any contact with a target device. Using this kind of fault injections, attackers can observe faulty outputs without leaving any hard evidence of the attacks. This remote-fault injection is believed to be very threatening in a practical attack scenario, where the attacker has a limited access to the target device.

In general, it is difficult to achieve a 100% accuracy of having the intended faults because of the environmental noise and the limitations of equipments, etc. Needless to say, remote-fault injections are much harder to control and result in being different from the attacker's expectation. Therefore, the key recovery of the DFA attacks in reality should be discussed with a noisy data set that includes many unexpected and useless data caused by unintended faults. Especially, when the remote-fault injections can hardly control the fault-injection timing and intensity, the data set will contain only a few useful data mixed with thousands or even millions of unexpected ones. In this paper, we discuss the key recovery strategy when the injected faults are uncertain. Note that we do not improve the existing DFA attacks under their assumed fault models, however, we rather combine them and extend their results considering that uncertain faults are injected. It is worth mentioning that the unintended faults are treated as noise and they can be any type of faults. Although some of these unintended faults can be useful for the key recovery if their injections are

[1] The University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[2] Tohoku University, Sendai, Miyagi 980–8575, Japan
[a] sakiyama@uec.ac.jp
[b] liyang@uec.ac.jp

certain. However, they are not helpful for the key recovery in the setting of uncertain faults due to too much computational cost. The requirement for a successful key recovery for the proposed algorithms is that several intended faults are injected and no false keys appears.

This paper also assumes the attackers do not know the value of the inputs, so that an exhaustive search using a plaintext-ciphertext pair is not accessible. This "chosen but unknown plaintext" setting has the same significance with the "uncertain faults" that challenges the successful key recovery for the attackers with less advantage. Also when the key confirmation using a plaintext-ciphertext pair is not available, the DFA key recovery can be failed with a false key. Accordingly, the possibility of the emergence of a false key needs to be discussed.

In this paper, firstly, we discuss a strategy when performing the remote-fault injection. Secondly, we propose a general algorithm for the DFA key recovery considering the probability of mistaking the false key for the correct one. Thirdly, we use AES as a case study to discuss the detailed strategy, and finally we demonstrate the attack result based on the practical data obtained from experiments with the EM-based remote-fault injections. We reveal that the combination of one 1-byte fault at the 8th round and two 1-byte faults at the 9th round are of great use to recover the secret key under the uncertain faults. One of the evaluation results shows that the secret key can be identified with a probability of 99.2% even when we have 32 millions of wrong data and a minimum of expected data. Based on a practical fault injection scenario, this work verifies the improvement of reducing the required data by applying the proposed key recovery method for uncertain faults.

The remainder of this paper is organized as follows. In Section 2, we have a discussion about the general strategy for remote-fault injections. In Section 3, we study a key identification procedure and propose the general key recovery algorithms considering both uncertain and detectable faults. Section 4 focuses on the discussion about the attack target of AES-128 to explore the best strategy for the key identification and the probability of the false key. In Section 5, we demonstrate the attack result using some practical data, and Section 6 concludes the paper.

2. General Strategy for Remote-Fault Injections

In this paper, we assume that the target device encrypts the attackers' chosen plaintexts but the values are unknown, and the corresponding ciphertexts are public. Namely, as a minimal condition, the attackers are able to inject faults to the target device remotely and have access to the faulty ciphertexts. Without any trigger signal available, the faults are injected at a random timing and intensity, and hence the information about the injected fault is principally unknown. In this section, we divide the strategy for the remote-fault injections into three parts to discuss.

2.1 Fault Injection Timing

If the attackers can make sure that each faulty ciphertext corresponds to a transient fault that disturbs intermediate values only at a certain round of AES, it will be easier for them to recover the secret key comfpared to the multiple-round faults. Each round of AES has 4 types of operations as SubBytes, ShiftRows, MixColumns and AddRoundKey. SubBytes is the only non-linear operations among them, while the injected fault will spread and affect each byte of the AES state after 2 rounds of calculations. As the target of the DFA attack, AES is very weak under certain types of injected faults, with which the full key recovery complexity becomes practical with only 2 intended fault injections. Therefore, the attackers will set the interval of fault injections longer than the duration of a one-time encryption operation. Moreover, assuming that each round of operation takes the same amount of calculation time, which is true in most cases, faults are assumed to be injected with the same probability for each round.

2.2 Fault Injection Intensity

The attackers will control the fault-injection intensity in order to have a low fault-injection rate, which is known to be preferable in the key recovery [3], [4]. More precisely, a small number of affected bytes is diffused rapidly through a cryptographic algorithm, e.g., two rounds for AES, so that they could recover all the secret key by performing differential analysis with small guessing entropy of the secret key. In the case of 128-bit AES (AES-128), when a fault is injected at the 9th or 10th round, the difference between fault-free and faulty ciphertexts generated from the same plaintext is not fully active as long as the fault injection does not affect multiple bytes at once. In other words, by checking the ciphertext difference, the attackers know that a specific fault, e.g., one-byte fault, is injected to the 9th or 10th round. Such detectable-fault injections enable us to determine the necessary fault model in the key recovery process. The key-extracting algorithms for undetectable and detectable faults will be discussed in Section 3 in detail.

2.3 Number of DFA Experiments

The more the attackers inject faults, the more likely they succeed in having intended faults that can be used for recovering the secret key. In the IEMI-based remote-fault injection by Hayashi et al. [9], they focused on injecting the most efficient fault, i.e., twice of 1-byte faults at the 8th round of AES [3]. In this case, it is difficult to decide how many experiments should be executed since the 8th-round fault is unable to be detected from the ciphertext difference.

Here, based on the assumption that all AES rounds are affected by faults with the same probabilities, the attackers can predict when to stop the experiments by observing the number of the 9th- or 10th-round fault injections that are detectable. For example, a reasonable stopping timing in attacking AES is when the two faulty ciphertexts corresponding to the 9th-round fault are collected. However, this method does not always provide an expected data set, which leads to failure of identifying the secret key.

Alternatively, the attackers can wait for multiple 1-byte faults at the 9th round that are detectable. However, we need at least 8 times of 1-byte faults at the 9th round, which requires more fault injections compared with the case of collecting two 1-byte faults at the 8th round. This strategy is inefficient and sometimes infeasible since the attackers have a limited access to the device and can often collect a minimum data set in a practical scenario.

In this situation, we construct a general strategy for the attackers to utilize a given data set and maximize the probability of identifying the true key.

3. Preliminary Study of Key Identification from Uncertain Faults

In a practical DFA experiment, we generally fail to inject an intended fault, i.e., an *uncertain fault* is injected to a device. In this case, we often result in performing DFA based on a wrong fault model, which leads to obtaining wrong key candidates. As a result, we may mistake wrong key(s) for a correct one. Such wrong keys are called *false keys* in this paper. Intuitively, the probability of taking a false key becomes higher as the uncertainty of faults is increased. In this section, we discuss the probability of retrieving false keys under uncertain faults in detail.

3.1 Notations and Abbreviations

The notations and abbreviations used in this paper are summarized as follows.

MC_y^x :	one of four MixColumns functions, labeled with $y(= 0, 1, 2, 3)$, at the x-th round of AES
k, k^{10} :	a secret key and 128-bit 10th-round key of AES
k_y^{10} :	a 4-byte 10th-round key of AES operated with the output of MC_y^9
m :	the amount of information leakage
p :	the number of total uncertain fault injections used for DFA experiments
q :	the number of intended fault injections used for DFA experiments
K, K_s :	key space, e.g., $K = 2^{128}$ and $K_s = 2^{32}$ for 128-bit AES
F_{xByR} :	fault model when an x-byte fault is injected at the y-th round of AES
$\tilde{F}_{xByR}$:	the same fault model as F_{xByR} but the attackers are unsure whether or not an x-byte fault is injected at the y-th round of AES
A, B, C, D :	sets used for identifying the true key

3.2 Identification of True Key from Undetectable-Uncertain Faults

We deal with uncertain faults, where intended and unintended faults are randomly injected for each experiment. More precisely, we have no clue as to whether the experimentally obtained key candidate sets include the true key or not. With undetectable-uncertain faults, we have two types of key candidate sets. One includes the true key and the other does not. They are obtained from DFA respectively based on a correct and wrong fault models[*1]. Algorithm 1 shows a general algorithm that can be used for extracting the true key from uncertain faults.

Intuition tells us that Algorithm 1 works well when the intended faults are injected at a high success rate and the size of key candidates, $|B_i|$, is small enough. This is because the condition of step 2 in Algorithm 1 is hardly taken for a false key. However, in some cases such that $|B_i|$ is not small enough for each DFA experiment, Algorithm 1 may output a false key. In the following, we discuss the conditions that Algorithm 1 outputs k correctly.

Algorithm 1 Algorithm for extracting the secret key from undetectable-uncertain faults

```
Require: the empty sets $A, B_0, B_1, \ldots,$ and $B_n$.
Ensure: the secret key $k$.
1: obtain sets of key candidates, $B_0$ and $B_1$, with two DFA experiments
   based on a fault model, $\tilde{F}$, and set $A = B_0$ and $i = 1$.
2: while $A \cap B_i = \emptyset$ do
3:     update $A$ as $A \Leftarrow A \cup B_i$, and increment $i$ $(i = i + 1)$.
4:     obtain $B_i$ with a DFA experiment based on $\tilde{F}$.
5: end while
6: update $A$ as $A \Leftarrow A \cap B_i$.
7: return the element of $A \ni k$.
```

We discuss the probability of having no false key(s), i.e., the case that step 2 in Algorithm 1 is taken by coincidence. Suppose that we keep on failing to inject an intended fault for p times of DFA experiments based on a wrong fault model. We do not have false keys as far as step 2 is not taken in Algorithm 1. In this case, we have

$$A = \bigcup_{i=0}^{p} B_i, \tag{1}$$

where $k \notin B_i$ $(i = 0, 1, \ldots, p)$. If the elements of $|B_i|$ are assumed to be randomly chosen from the whole key space[*2], the probability of satisfying Eq. (1) can be expressed as

$$\begin{aligned}\Pr\left[A = \bigcup_{i=0}^{p} B_i\right] &= \prod_{i=1}^{p} \Pr\left[B_i \cap \bigcup_{j=0}^{i-1} B_j = \emptyset\right] \\ &= \prod_{i=1}^{p} \prod_{j=1}^{b} \left(1 - \frac{bi}{K - j + 1}\right),\end{aligned} \tag{2}$$

where the assumption, $|B_i| = b$, holds for $i = 0, 1, \ldots, p$ for simplicity. Note that, as far as using the same fault model, we have almost the same number of key candidates for each DFA experiment in a practical case. According to Eq. (2), if we take b small enough such that $b \ll K$, we have

$$\Pr\left[A = \bigcup_{i=0}^{p} B_i\right] \approx \prod_{i=1}^{p} \left(1 - \frac{bi}{K}\right)^{b}. \tag{3}$$

Next, we consider the case that the true key, k, can be found in several sets of B_i. Suppose that Algorithm 1 outputs the true key on the p-th execution of the DFA experiment. In this case, the probability of having no false keys by executing Algorithm 1 is regarded the same as Eq. (3)[*3]. Therefore, as far as the condition of $b \ll K$ is satisfied, Algorithm 1 outputs the true key correctly with a probability given by Eq. (3). In other words, Eqs. (2) and (3) are for calculating the probability that every element from all the key candidate sets is unique.

[*1] Note that both models are the same when we expect a specific fault injection.

[*2] This is a natural assumption in a DFA experiment considering the property of a cryptographic algorithm such as AES.

[*3] The cardinality of the sets containing the true key is also assumed to be b.

Algorithm 2 Algorithm for extracting the secret key from uncertain faults utilizing detectable faults

```
Require: the empty sets $C, D_0, D_1, \ldots,$ and $D_n$.
Ensure: the secret key $k$.
 1: obtain a set of key candidates, $C$, with one or several DFA experiments
    based on a fault model, $F_u$ (i.e., $k \in C$,) and set $c = |C|$ and $i = 0$.
 2: while $|C| = c$ do
 3:   obtain a set of the key candidates, $D_i$, with a DFA experiment based
      on a different fault model, $\tilde{F}_v$.
 4:   if $C \cap D_i \neq \emptyset$ then
 5:     update $C$ as $C \Leftarrow C \cap D_i$.
 6:   else
 7:     hold $C$ and discard $D_i$, and increment $i$ $(i = i + 1)$.
 8:   end if
 9: end while
10: return the element of $C$.
```

3.3 Identification of True Key from Uncertain Faults Utilizing Detectable Faults

In contrast to the case that all the faults are undetectable and uncertain, we sometimes can judge whether or not the intended fault is injected from the ciphertext such as the 1-byte fault at the 9th round, F_{1B9R}, with high probability [*4]. Algorithm 2 shows a general procedure that can be used for extracting the true key, which makes the best use of detectable faults.

Similar to Algorithm 1, Algorithm 2 can extract the true key when the intended faults are injected at a high success rate and the sizes of key candidates, $|C|$ and $|D_i|$, are small enough. It is worth mentioning that Algorithm 2 requires much less memory than Algorithm 1 because we can check whether $\tilde{F}_v$ is the intended one or not by following the step 4 in Algorithm 2. In other words, the undetectable $\tilde{F}_v$ becomes detectable thanks to C obtained based on F_u, which makes the key recovery efficient.

If the detectable faults are used for reducing a part of the key, C contains a partial-key candidates. Therefore, the key space dealt with in Algorithm 2, denoted as K_s, is normally smaller than the whole key space, K. More specifically, Algorithm 2 can also be understood as an algorithm extracting the 4-byte 10th-round key, i.e., k_0^{10}, from the detectable F_{1B9R} and undetectable-uncertain 1-byte faults at the 8th round, $\tilde{F}_{1B8R}$, where intended and unintended faults are indistinguishably mixed.

Accordingly, we can derive the probability of having no false keys, i.e., step 4 in Algorithm 2 is not taken by coincidence, during the p times of DFA experiments as

$$\begin{aligned}\Pr\Big[|C| = c\Big] &= \prod_{i=1}^{p}\prod_{j=1}^{d}\left(1 - \frac{c}{K_s - j + 1}\right)\\ &\approx \left(1 - \frac{c}{K_s}\right)^{dp}\\ &\approx \exp\left(-\frac{cd}{K_s}p\right), \qquad (4)\end{aligned}$$

where we assume that $|D_i| = d$ holds for $i = 0, 1, \ldots, p$ and $c, d \ll K_s$.

[*4] F_{2B9R}, F_{3B9R}, and F_{4B9R} may be misjudged as F_{1B9R}. However, we ignore such cases since the probabilities are assumed to be much lower than that of F_{1B9R}. Even if we misjudge the fault, we can eliminate the resultant false keys as will be discussed in Section 4.

Suppose that Algorithm 2 outputs the true key on the p-th execution of the DFA experiment. In this case, as far as the condition of $c, d \ll K_s$ is satisfied, Algorithm 2 outputs the true key correctly with a probability given by Eq. (4). Equation (4) is similar to Eqs. (2) and (3). The difference is that c replaces bi in Eq. (4) since the key candidate set obtained from the detectable fault is used.

3.4 Practicability of Algorithms 1 and 2

However, in the case that the number of key candidates, b, c, or d, is not small enough for each DFA experiment, Algorithms 1 and 2 may output a set including false keys. When multiple elements appear as the result of Algorithm 1 or 2, at least one of them is a false key. Then the attacker can continue the search for the remaining data for the key candidate that appears the most times. After all the data are used, if still multiple key candidates are equally likely to be correct, then the data is not enough for key identification. However, we can restrict the key space to a very small set. In the setting of no key confirmation using a plaintext-ciphertext pair, the recovered key cannot be surely the correct one. In the following sections, we discuss the practicability of Algorithm 1 based on several well-known fault models on AES-128.

4. Fault Models for AES-128

4.1 One-byte Faults at the 8th Round of AES-128

For AES-128, the most efficient DFA was firstly proposed by Piret and Quisquater in 2003 [3], and several optimizations have been reported [5], [6]. Tunstall and Mukhopadyay proposed a DFA method whereby the key space can be reduced up to 2^{12}. It is assumed that an attacker is able to inject a one-byte fault at the 8th round, F_{1B8R}, and they analyze the differential of the correct and faulty outputs to retrieve the key candidates.

Under the fault model of F_{1B8R}, it is reported that the leaked information, m, can be derived as

$$m = 128 - \log_2\Big((2^8 - 1) \times 16\Big) \approx 116.006,$$

in the case of AES-128 from the information-theoretic point of view [7]. Therefore, we conclude that the method by Tunstall and Mukhopadyay is optimal [*5]. Suppose that the attacker can inject the intended fault, F_{1B8R}, without failure and obtain two sets of the key candidates, B_0 and B_1 in Algorithm 1, whose cardinalities are both 2^{12}. The probability in Eq. (3) becomes almost 1 for $b = 2^{12}$, $p = 2$, and $K = 2^{128}$, which means that we hardly have false keys.

We extend this discussion to DFA with uncertain faults, which has not been mentioned explicitly in any previous work. Considering a practical case, where p is small enough compared to K, e.g., $p = 2^{20}$, the probability in Eq. (3) also becomes almost 1. Therefore, uncertain faults aiming at F_{1B8R} are regarded as powerful to identify the 128-bit secret key of AES-128. However, we should notice the fact that as p increases, the probability gradually decreases and a false key may occur in Algorithm 1. In this case, we may need more sets of key candidates from DFA with

[*5] This agrees well with the experimental results using an actual AES implementation.

the intended fault.

4.2 One-byte Faults at the 9th Round of AES-128

The one-byte fault model at the 9th round, F_{1B9R}, is also useful for the attackers.de The fault propagates through one of the MixColumns functions at the 9th round, and results in a 4-byte difference in the output, which means that the corresponding 4-byte key space can be reduced based on the model of F_{1B9R}. We denote the column of the MixColumns functions with the fault propagation at the 9th round as $\mathsf{MC}^9_\square$. According to the information-theoretic observation [7], the information leakage with regard to 4 bytes of the 10th-round key, $k^{10}_\square$, is derived as

$$m = 32 - \log_2\left((2^8 - 1) \times 4\right) \approx 22.006.$$

Therefore, the key space of $k^{10}_\square$ can be reduced up to 2^{10} approximately. Suppose that two faults of F_{1B9R} are injected as intended and they are propagated to the same $\mathsf{MC}^9_\square$. We have $b = 2^{10}$, $p = 2$, and $K = 2^{32}$ in Algorithm 1 and can use the relation of Eq. (3), and the probability becomes almost 1. Thus, we rarely have false keys in Algorithm 1.

Note that we do not consider the case for the uncertain fault aiming to inject F_{1B9R} because the attacker easily knows whether or not F_{1B9R} is actually injected from the difference between correct and faulty outputs. Again, as mentioned in Section 2.3, the key recovery based on only the detectable F_{1B9R} is possible but considered to be ineffective since the available information from F_{1B8R} is not effectively used.

4.3 One-byte Faults at the Eighth and Ninth Rounds of AES-128

This section shows that it is a good strategy to use the combination of undetectable-uncertain F_{1B8R} (i.e., $\tilde{F}_{1B8R}$) and detectable F_{1B9R} to identify the key. As the first step, the faults of F_{1B9R} are used with $\tilde{F}_{1B8R}$ to reduce the key space of $k^{10}_\square$. As the second step, for the reduced key space, we apply the fault model of F_{1B8R} to the rest of DFA experiments as discussed in Section 4.1. This analysis method is especially useful when the intended F_{1B8R} faults are rarely injected.

Assuming that we succeed in injecting the intended F_{1B8R} only once through DFA experiments, in the following, we discuss how many faults of F_{1B9R} are necessary for identifying the key.

4.3.1 Case of $F_{1B8R} \times 1 + F_{1B9R} \times 1$ Under Certain Fault Injections

For instance, we have one F_{1B9R} and one F_{1B8R} injected to AES-128 as intended in two DFA experiments. The 4 bytes of the 10th-round key, e.g., k^{10}_1, XOR-ed with the output of a specific MixColumns function, e.g., MC^9_1 as shown in **Fig. 1**, can be identified with a probability of almost 1 as mentioned in Section 4.3. Accordingly, we can specify all the differential values of the output of MC^9_1 using the identified 4-byte 10th-round key, k^{10}_1. Hence, the 1-byte differential value in the output of MC^8_0 is exactly determined. Moreover, as we know that only 1 byte is active in the input of MC^8_0, we have only four possible differential values in the inputs of MC^8_0 considering the differential property of the MixColumns function. This observation enables us to obtain the information leakage as

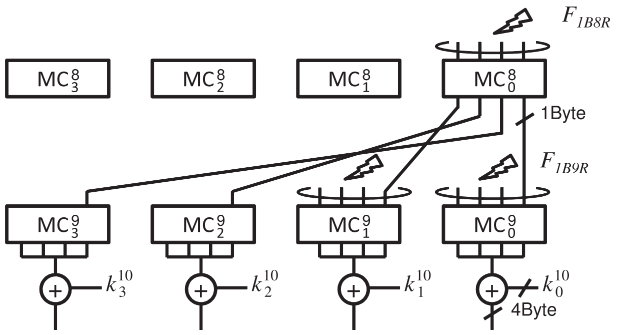


Fig. 1 A simplified AES structure oriented to the differential property used in this paper.

$$m = 128 - \log_2 4 = 126.$$

Therefore, we cannot identify the 128-bit key and three false keys would be obtained theoretically. Note that the above observation holds for the case that more than one fault of F_{1B9R} is injected to the input of the same MC^9_1.

4.3.2 Case of $F_{1B8R} \times 1 + F_{1B9R} \times 2$ Under Certain Fault Injections

If we have one fault for F_{1B8R} and (more than) two faults for F_{1B9R}, which are related to different MixColumns functions, e.g., MC^9_0 and MC^9_1 in Fig. 1, injected to AES-128 without failure, we can identify the key. In order to understand this, we consider two sets of key candidates, each of which is obtained from one F_{1B9R} and one F_{1B8R} as mentioned above. Note that each set of the key candidates has three 128-bit false keys depending on the location of the differential values in the inputs of MC^8_0. The probability that the false keys collide for two sets is negligible, so that the key can be identified.

4.3.3 Case of $\tilde{F}_{1B8R} \times p + F_{1B9R} \times \alpha$ Under Uncertain Fault Injections ($\alpha = 1, 2$)

However, in the case that uncertain faults happen while aiming to inject F_{1B8R}, the probability of having no false keys becomes higher as q/p increases.

In order to explain this, we consider an example case that one F_{1B8R} is realized in 2^{11} fault injections and one F_{1B9R} is obtained. As discussed previously, we first use the F_{1B9R} to find out the correct F_{1B8R} from 2^{11} times of experiments. Note that the correct F_{1B8R} can identify 4 bytes of $k^{10}_\square$ with the F_{1B9R} while the false F_{1B8R} usually cannot. Using Eq. (4) for $c = d = 2^{10}$, $p = 2^{11}$, and $K_s = 2^{32}$, the probability of identifying the correct F_{1B8R} is derived as

$$\exp\left(-\frac{cd}{K_s}p\right) = \exp\left(-\frac{2^{10}\cdot 2^{10}}{2^{32}}\cdot 2^{11}\right) \approx 0.607.$$

As p increases, the probability becomes lower. Accordingly, we have several false keys left as 4 bytes of $k^{10}_\square$ with the case for DFA experiments with $\tilde{F}_{1B8R} \times p + F_{1B9R} \times 1$ for a large p.

Furthermore, in order to derive the expected value of false keys, denoted as w, we consider the probability that the number of false keys is i. Assuming that at most one false key is obtained in a single DFA experiment, the probability can be expressed as

$$\binom{p}{i}\left(\binom{d}{1}\frac{c}{K_s}\left(1-\frac{c}{K_s}\right)^{d-1}\right)^i\left(\left(1-\frac{c}{K_s}\right)^d\right)^{p-i}$$

$$= \binom{p}{i}\left(\frac{cd}{K_s}\right)^i\left(1-\frac{c}{K_s}\right)^{dp-i}$$
$$= \binom{p}{i}\left(\frac{cd}{K_s-c}\right)^i\left(1-\frac{c}{K_s}\right)^{dp}.$$

Therefore, the expected value is derived as

$$w \approx \sum_{i=1}^{p}\left\{i\binom{p}{i}\left(\frac{cd}{K_s-c}\right)^i\left(1-\frac{c}{K_s}\right)^{dp}\right\}$$
$$= p\left(1-\frac{c}{K_s}\right)^{dp}\sum_{i=1}^{p}\left\{\binom{p-1}{i-1}\left(\frac{cd}{K_s-c}\right)^i\right\}$$
$$= p\left(1-\frac{c}{K_s}\right)^{dp}\left(\frac{cd}{K_s-c}\right)\left(1+\frac{cd}{K_s-c}\right)^{p-1}$$
$$\approx \frac{cd}{K_s}p\cdot\exp\left(-\frac{cdp}{K_s}\right)\cdot\exp\left(\frac{cd(p-1)}{K_s}\right)$$
$$= \frac{cd}{K_s}p\cdot\exp\left(-\frac{cd}{K_s}\right)$$
$$\approx \frac{cd}{K_s}p. \tag{5}$$

Hence, we see that w is proportional to p when $c, d \ll K_s$.

For the case of $\tilde{F}_{1B8R}\times p + F_{1B9R}\times 2$, suppose that two 4-byte 10th-round keys, e.g., k_0^{10} and k_1^{10}, are obtained by executing Algorithm 2 twice [*6]. Specifically, we execute Algorithm 2 using a data set with one F_{1B9R} at MC_0^9 and $\tilde{F}_{1B8R}\times p$ to obtain k_0^{10}. And then, we execute Algorithm 2 again based on the same data set including one F_{1B9R} at MC_1^9 and $\tilde{F}_{1B8R}\times p$ for retrieving k_1^{10}. Once we have k_0^{10} and k_1^{10}, we see the validity of those 4-byte key candidates by using the property of the MixColumns function at the 8th round, i.e., MC_0^8, to see the consistency among the key candidates. More precisely, we check whether or not pairs of the key candidates for k_0^{10} and k_1^{10} are generated from the same DFA experiments based on $\tilde{F}_{1B8R}$, and the differential values meet the relationship of MC_0^8.

Suppose that we obtain w false keys, as derived in Eq. (5), for both k_0^{10} and k_1^{10}, in addition to the true key. Then, we consider the probability that $j(\le w)$ pairs of the false keys for k_0^{10} and k_1^{10} satisfy the above-mentioned consistency. We notice the following facts in deriving the probability.

(i) We assume that false keys for k_0^{10} and k_1^{10} are independently determined corresponding to w (out of p) DFA experiments based on $\tilde{F}_{1B8R}$. We have the probability that at least one pair of the key candidates for k_0^{10} and k_1^{10} are generated from the same DFA experiments is $Q_j = \binom{w}{j}\binom{p-w}{w-j}/\binom{p}{w}$.

(ii) Moreover, the probability that the differential values meet the relationship of MC_0^8 is about $2^8/2^{16}$ or 2^{-8} since there are 2^8-1 and $(2^8-1)^2$ differential values respectively for the input and output of MC_0^8.

As a result, we have the probability that there are no false keys and identify the key as

$$\left(1-\sum_{j=1}^{w}Q_j\right)+\left(\sum_{j=1}^{w}Q_j(1-2^{-8})^j\right)$$

[*6] Note that we have several key candidates for k_0^{10} and k_1^{10} according to the results summarized in Table 1, and the key candidates are not narrowed down as for k_2^{10} and k_3^{10}.

Table 1 Probability of identifying the correct k^{10} for various p in the case for $\tilde{F}_{1B8R}\times p + F_{1B9R}\times 2$ ($c = d = 2^{10}$ and $K_s = 2^{32}$).

p	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}	2^{25}
w	256	512	1024	2048	4096	8192
Pr.	1.000	1.000	1.000	0.998	0.996	0.992

$$= 1-\sum_{j=1}^{w}Q_j\left(1-(1-2^{-8})^j\right). \tag{6}$$

Table 1 summarizes the probability of identifying k^{10} for various p, which is derived by Eq. (6). We know that even under uncertain faults, we can identify the true key at a high probability by utilizing two F_{1B9R} related to different $\mathsf{MC}_\square^9$. As a result, considering the case that 32 millions or 2^{25} DFA experiments, which is regarded as near to a limitation of reality, the success rate of identifying the key is 99.2%.

In this section, for a specific attack target of 128-bit AES, we discussed the key recovery under different fault models. In summary, Sections 4.3.1 and 4.3.2 discuss the attacks using certain faults, while Section 4.3.3 is about uncertain faults. In Section 4.3.3, the fault of F_{1B9R} is treated as detectable uncertain faults and the discussion is related to the Algorithm 2 in Section 3. It is worth mentioning that F_{1B9R} is not widely used in previous fault attacks. On the other hand, Algorithm 1 that only uses F_{1B8R} is often used since it is the most straightforward key identification method when ignoring the usage of F_{1B9R}.

5. DFA Experiments with Remote-Fault Injections

5.1 Intentional Electromagnetic Interference on AES Hardware

We employ the same configuration for the fault-injection experiments as in Refs. [9], [10], [11].

Figure 2 shows a block diagram of the Intentional ElectroMagnetic Interference (IEMI) fault-injection setup, where the cryptographic module is mounted on a common device, i.e., a PCB board, equipped with a twisted-pair power cable. In general, IEMI is an overt threat that usually causes permanent damage when applied to electronic devices [12]. However, in this experiment, we cause transient faults in electrical devices without damaging their operation and hardware based on above previous studies. The main purpose in the implementation of this IEMI technique is to invert one or several bits in the intermediate result, which can be recovered after a reset or at the end of the operation. Here, the entire function of the Integrated Circuit (IC) is expected to be unchanged during such a fault injection. This section presents an experiment in which the above-mentioned IEMI-based remote-fault injection is applied to an actual cryptographic IC. We first describe the setup of the experiment and injection results.

As shown in Fig. 2, we employ a standard evaluation board SASEBO-G as the test device [13]. The side-channel attack standard evaluation board (SASEBO) is developed for evaluating the cryptographic implementations against physical attacks including the fault-based attacks. SASEBO-G is an FPGA-based type of SASEBO boards, which has several ports that make the power consumption measurement more accessible. In our experiment,

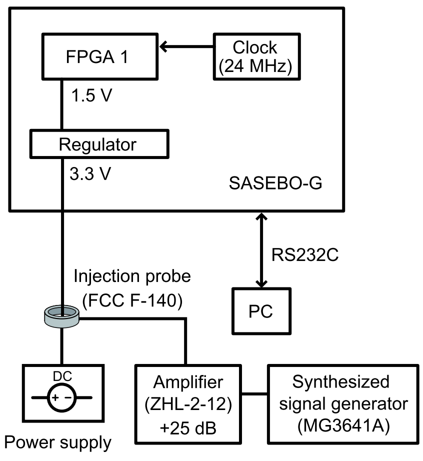

FPGA 1
Clock (24 MHz)
1.5 V
Regulator
3.3 V
SASEBO-G
RS232C
Injection probe (FCC F-140)
PC
DC
Power supply
Amplifier (ZHL-2-12) +25 dB
Synthesized signal generator (MG3641A)

Fig. 2 Experimental setup.

we measure the power consumption of the FPGA from a connector named J8 to observe the effects of fault injection via the power consumption variation. An AES circuit supporting 128-bit key length is implemented in FPGA1. The circuit uses a loop architecture, where one round is processed every clock cycle. As a result, a single encryption operation takes 10 clock cycles for the ten cryptographic rounds and one additional clock cycle for data I/O. The clock frequency and the supply voltage on SASEBO-G are 24 MHz and 3.3 V, respectively. The secret key is a reference value (0x2b7e151628aed2a6abf7158809cf4f3c), as given in the algorithm specification. The fault injections are performed for 340,000 different plaintexts generated at random, and the faulty outputs, i.e., ciphertexts, are stored in a PC. We calculate the true outputs separately and determine what kinds of faults have occurred in the module by comparing the corresponding true output with the faulty one.

Sinusoidal waves are generated by a signal generator (MG3641A), after which they are amplified by using an amplifier (ZHL-2-12.) Finally, the sinusoidal waves are introduced via an injection probe (FCC F-140) into a power cable attached to SASEBO-G. The injection probe is located 60-cm away from SASEBO-G. In this experiment, we observe the VDD/GND fluctuations of the FPGA1 by using an oscilloscope (MSO6104A) in order to determine how the introduced sinusoidal wave affects the original wave of the voltage drop. Note that the measurement point is located on the GND line of FPGA1.

In the experiments, we employed sinusoidal waves with a frequency between 170 and 200 MHz, increased at steps of 1 MHz, against SASEBO-G running AES operations continuously. This is because the transfer functions for transferring from the injection probe to the points near FPGA1 have the lowest decrease rate. We find that we can generate faults with high probability based on the previous experimental results [9], [10], [11]. Note that in this case it is also possible to induce faults by introducing an impulse wave composed of a wide range of frequencies, e.g., a wave generated by an ESD gun, with a much higher voltage, e.g., around 10 – 1,000 V. However, such high-voltage signals would damage some of the components inside the device, and therefore such an IEMI attack would not pose a severe threat from the viewpoint of information leakage.

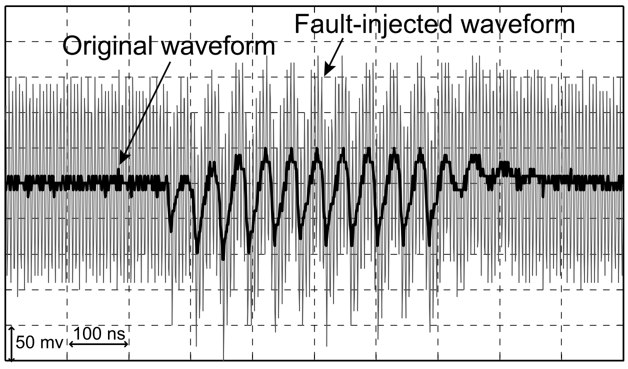


Fig. 3 Voltage waveforms observed during AES operation.

Table 2 Occurrence frequency of F_{1B8R} and F_{1B9R} with IEMI-based remote-fault injections.

F_{1B8R}	F_{1B9R}
0.007% (1/13,640)	0.073% (10/13,640)

Figure 3 shows voltage fluctuations between VDD and GND during AES operations with and without any fault injection, which is observed at a power consumption measurement connector J8. The encryption process starts at around 250 ns and finishes after 11 clock cycles. There are 11 peaks in each of the waveforms. Compared with the original waveform, the faulty waveform fluctuated widely in the range of approximately 400 mV. In this experiment, faulty outputs were observed as a result of injected voltages in the interval between 132 and 135 dBμV. As shown in Fig. 3, the AES power consumption waveform with no fault injection (black line) is much more stable compared to the waveform under fault injection (grey lines). The electromagnetic interference varies the electric potential of VDD/GND and generates temporary faults. Such faults would be similar to those obtained by under powering. The results show that faults can be injected by using sinusoidal waves of several volts, which can be generated by a combination of off-the-shelf equipments, under the experimental conditions of injecting faults from a distance of 60 cm from the device. On the other hand, we did not observe any faults to the cryptographic module when we employed sinusoidal waves in other frequency bands because the other sinusoidal waves first caused damage to other modules.

5.2 Results of DFA with IEMI-Based Remote-Fault Injections

Based on the experiments shown in Section 5.1, **Table 2** shows the occurrence frequency of faults for F_{1B8R} and F_{1B9R}. Different from the theoretical assumption we made, the occurrence frequency of the faults varies for each round. This is because the IEMI-based remote-fault injection is not stable and the size of each data set is not big enough.

As can be seen from the data set, two faults of F_{1B8R} are not available so that we fail to identify the key as far as using the previous strategy [9]. Instead, we thought of using the detectable faults, F_{1B9R}. However, we may not be able to identify the true key since the detectable faults, F_{1B9R}, may not be related to all the different $\mathsf{MC}^{9}_{\square}$ as discussed in Section 2.3, which is, in fact, the case of the data set obtained from the experiments. Therefore,

Table 3 Comparison between three attack approaches to identify the true key.

Fault Model	Identification Alg.	Mem. [Bytes]	Time [$T_{F_{1B9R}}$]	Key Identification
$\tilde{F}_{1B8R} \times 2$	Algorithm 1	$2^{16} \cdot p$	$5p$	Fail
$F_{1B9R} \times z$	-	2^{12}	z	Fail
$\tilde{F}_{1B8R} \times p + F_{1B9R} \times 2$	Algorithm 2	2^{17}	$p + w$	Success

For the judgement of the key identification, we use the experimental data in Table 2 and $z = 10$.
For the comparison of the memory and time cost, we use the theoretical estimations.

the only strategy to identify the true key in this case is to use the fault model, $\tilde{F}_{1B8R} \times p + F_{1B9R} \times 2$.

5.2.1 Comparison for Key-Identification Experiments

The key recovery does not use the detectable faults of F_{1B9R}, which is similar to Algorithm 1. While we actively use the F_{1B9R} in our key recovery process which makes our key recovery process similar to Algorithm 2. Three key recovery processes are compared in this section.

First of all, the key recovery using F_{1B9R} has a higher success rate for the identification of the true key than the one that only uses F_{1B8R}. For example for the data set we mentioned that only one fault of F_{1B8R} is available, the secret key cannot be identified using Algorithm 1.

For the key-identification process based on only faults of F_{1B8R}, each faulty ciphertext should be treated as it is corresponding to the fault of F_{1B8R}, therefore all the corresponding key spaces have to be stored as can be seen in Algorithm 1. Thus, the memory required to store all the key space is estimated to be about $2^{12} \cdot p \cdot 16$ or $2^{16} \cdot p$. For the time complexity, we consider each F_{1B8R} is treated as 4 times of F_{1B9R} at the final AES round and 1 time of F_{1B9R} at the 9th AES round. The total process time is about $z \cdot T_{F_{1B9R}}$ since only the z faulty ciphertexts need to be used.

As for the key-identification process based on only faults of F_{1B9R}, which is detectable. Therefore, we need the memory only to store all the partial-key candidates, $k^{10}_{\square}$, which is estimated as $2^{10} \cdot 4$ or 2^{12}, approximately. For the time complexity, we consider each F_{1B8R} is treated as 4 times of F_{1B9R} at the final AES round and 1 time of F_{1B9R} at the 9th AES round. Denote the time of dealing with a F_{1B9R} as $T_{F_{1B9R}}$, the total process time is about $5p \cdot T_{F_{1B9R}}$ since all the p faulty ciphertexts have to be treated as F_{1B8R}. This approach requires significantly less memory and time costs, however, it heavily depends on the property of the fault injections.

On the other hand, for the approach using both F_{1B8R} and F_{1B9R}, which uses the detectable F_{1B9R} to verify the undetectable $\tilde{F}_{1B8R}$, only the key space for the verified F_{1B8R} needs to be stored. As a result, we can largely reduce the memory cost for storing the key spaces. Considering only 2 faults of F_{1B9R} and 1 fault of F_{1B8R} are required for the key identification, the size of the stored key space is about $2^{10} \cdot 2 \cdot 4 + 2^{12} \cdot 16 \approx 2^{17}$. For the processing time, first, key candidates of two different $k^{10}_{\square}$ are obtained in $2 \cdot T_{F_{1B9R}}$. Secondly, we need about $p \cdot T_{F_{1B9R}}$ to reduce the key candidates for a $k^{10}_{\square}$ and we have w key candidates. At the same time, we can specify w faulty ciphertexts that satisfy the fault model of F_{1B8R} out of p faulty ciphertexts. For the w faulty ciphertexts, we check the consistency as discussed in Section 4.3.3 [*7], which requires $w \cdot T_{F_{1B9R}}$. Lastly, we perform DFA based on F_{1B8R} to identify the whole key, which requires $5 \cdot T_{F_{1B9R}}$. In total, we have $(2 + p + w + 5) \cdot T_{F_{1B9R}} \approx (p + w) \cdot T_{F_{1B9R}}$ for the time cost.

As shown in **Table 3**, we compare the memory and time complexity for these two key recovery processes as well as the results of the key identification. For AES-128, the most useful fault injection is F_{1B8R}, which has been discussed a lot, while F_{1B9R} has not been done enough since it is not very effective for the key recovery. Therefore, even under the more practical scenario where only uncertain faults are available, Algorithm 1 tends to be used by considering F_{1B8R}, which leads to a straightforward result from previous researches. However, we note that F_{1B9R} is detectable and can help significantly in the key identification process, and hence Algorithm 2 is proposed here. Comparing the results using Algorithms 1 and 2, we know that Algorithm 2 is better in key recovery for the data set obtained from a practical experiment (see Table 3). The reason behind this is that the detectable fault of F_{1B9R} is actively used and the useful information is not wasted for the key recovery.

6. Conclusions

This paper discussed the practical strategy for the key recovery on AES implementation with uncertain faults. For the uncertain faults, the injected faults are not necessarily as intended, and therefore the previous DFA researches cannot be applied directly. In this paper, we proposed a methodology for the key recovery process considering the uncertain faults can be divided as undetectable ones and the others. We discussed the probability of having false keys in detail against various scenarios of fault occurrences. The proposed key recovery algorithms and probability calculation are verified using the data from the IEMI-based remote-fault injections. Our work filled the gap between the practical injections and the theoretical DFA key recovery researches. We also demonstrated that the key can still be identified effectively at a high success rate when the injected faults are uncertain and noisy.

Acknowledgments This research was partially supported by Strategic International Cooperative Program (Joint Research Type), Japan Science and Technology Agency.

References

[1] Barenghi, A., Breveglieri, L., Koren, I. and Naccache, D.: Fault Injection Attacks on Cryptographic Devices: Theory, Practice, and Countermeasures, *Proc. IEEE*, Vol.100, No.11, pp.3056–3076 (2012).
[2] Biham, E. and Shamir, A.: *Differential Cryptanalysis of the Data Encryption Standard*, Springer (1993).
[3] Piret, G. and Quisquater, J.-J.: A differential fault attack technique against SPN Structures, with application to the AES and KHAZAD, *CHES*, pp.77–88 (2003).
[4] Moradi, A., Shalmani, M.T.M. and Salmasizadeh, M.: A generalized method of differential fault attack against AES, cryptosystem, *CHES*, pp.91–100 (2006).

[*7] We assumed that the true key can be identified by the consistency check (i) in Section 4.3.3 for simplicity)

[5] Tunstall, M. and Mukhopadhyay, D.: Differential fault analysis of the advanced encryption standard using a single fault, Cryptology ePrint Archive, Report 2009/575 (2009), available from ⟨http://eprint.iacr.org/⟩.

[6] Mukhopadhyay, D.: An improved fault based attack of the advanced encryption standard, *AFRICACRYPT*, pp.421–434 (2009).

[7] Sakiyama, K., Li, Y., Iwamoto, M. and Ohta, K.: Information-theoretic approach to optimal differential fault analysis, *IEEE Trans. Information Forensics and Security*, Vol.7, No.1, pp.109–120 (2012).

[8] National Institute of Standards and Technology, FIPS 197: Advanced Encryption Standard (Nov. 2001).

[9] Hayashi, Y., Gomisawa, S., Li, Y., Homma, N., Sakiyama, K., Aoki, T. and Ohta, K.: Intentional electromagnetic interference for fault analysis on AES block cipher IC, *Workshop on Electromagnetic Compatibility of Integrated Circuits* (*EMC COMPO*), pp.235–240 (2011).

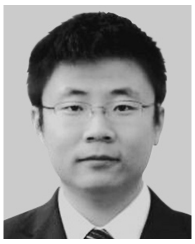
[10] Hayashi, Y., Homma, N., Sugawara, T., Mizuki, T., Aoki, T. and Sone, H.: Non-invasive EMI-based fault injection attack against cryptographic modules, *IEEE International Symposium on Electromagnetic Compatibility* (*EMC*), pp.763–767 (2011).

[11] Hayashi, Y., Homma, N., Sugawara, T., Mizuki, T., Aoki, T. and Sone, H.: Non-invasive trigger-free fault injection method based on intentional electromagnetic interference, *Non-Invasive Attack Testing Workshop* (*NIAT*) (2011).

[12] Radasky, W., Baum, C. and Wik, M.: Introduction to the special issue on high-power electromagnetics (HPEM) and intentional electromagnetic interference (IEMI), *IEEE Trans. Electromagnetic Compatibility*, Vol.46, No.3, pp.314–321 (2004).

[13] Research Center for Information Security (RCIS): Side-channel attack standard evaluation board (SASEBO), available from ⟨http://www.rcis.aist.go.jp/special/SASEBO/CryptoLSI-en.html⟩.

Kazuo Sakiyama received his B.E. and M.E. degrees from Osaka University, Japan, in 1994 and 1996, respectively, M.S. degree from the University of California, Los Angeles in 2003, and Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Belgium in 2007. From 1996 to 2004, he was with the Semiconductor and IC Division, Hitachi, Ltd. He is currently a Professor at The University of Electro-Communications, Tokyo. He is a member of IACR, IPSJ and IEEE.



Yang Li received his B.E. degree in electronic and information engineering from Harbin Engineering University, Harbin, in 2008, M.E. degree in information and communication engineering and Ph.D. degree in faculty of informatics and engineering from The University of Electro-Communications, Tokyo, in 2011 and 2012, respectively. He is currently a Project Assistant Professor in department of informatics at The University of Electro-Communications, Tokyo. His main research interest includes security evaluation and improvement for cryptographic hardware and embedded systems.

Shigeto Gomisawa was born in 1987. He received his B.E. degree in information and communication engineering from The University of Electro-Communications in 2010. He obtained a Master degree from Information Systems at The University of Electro-Communications.

Yu-ichi Hayashi received his B.E. degree in computer science and engineering from Aizu University, Aizuwakamatsu, Japan, in 2003, and M.S. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2005 and 2009, respectively. He is currently an Associate Professor at the Graduate School of Information Sciences, Tohoku University. His research interests include electromagnetic compatibility and information security. He is the Chair of Electromagnetic Information Leakage Subcommittee in IEEE Electromagnetic Compatibility Technical Committee 5. Dr. Hayashi received the Best Paper Award at Computer Security Symposium in 2009, the Young Researchers Award in the Institute of Electronics Information and Communication Engineers in 2010, the Excellent Presentation Award at the Institute of Electrical Engineers of Japan in 2011, and the Best Symposium Paper Award at 2013 IEEE EMC International Symposium on Electromagnetic Compatibility.

Mitsugu Iwamoto received his B.E., M.E., and Ph.D. degrees from The University of Tokyo, Tokyo, Japan, in 1999, 2001, and 2004, respectively. In 2004, he joined The University of Electro-Communications, where he is currently an Assistant Professor of the Center for Frontier Science and Engineering. His research interests include information theory and cryptography. He is a member of the IACR and the IEICE.

Naofumi Homma received his B.E. degree in information engineering, and M.S. and Ph.D. degrees in information science from Tohoku University, Sendai, Japan, in 1997, 1999, and 2001, respectively. He is currently an Associate Professor of the Graduate School of Information Sciences at Tohoku University. His research interests include computer arithmetic, high-performance VLSI computing, and cryptographic hardware. Dr. Homma received the IP Award at the 2005 LSI IP Design Award.

Takafumi Aoki received his B.E., M.E., and Ph.D. degrees in electronic engineering from Tohoku University, Sendai-shi, Japan, in 1988, 1990, and 1992, respectively. He is currently a Professor in the Graduate School of Information Sciences, Tohoku University. From 1997 to 1999, he also joined the PRESTO project, Japan Science and Technology Corp. (JST). His research interests include theoretical aspects of computation, VLSI computing structures for signal and image processing, multiple-valued logic, and bio molecular computing. He received various awards including the IEE Ambrose Fleming Premium Award in 1994, the IEICE Inose Award in 1997, and the IEE Mountbatten Premium Award in 1999.

Kazuo Ohta received his B.S., M.S., and Dr.S. degree from Waseda University, Tokyo, Japan, in 1977, 1979, and 1990 respectively. He has been a Professor at The University of Electro-Communications since 2001. He had been a researcher at NTT laboratories between 1979 and 2001. He is presently engaged in research on information security. He is a member of IEEE and IACR.