# Coupon Collector's Problem for Fault Analysis against AES — High Tolerance for Noisy Fault Injections

Yu Sasaki[1], Yang Li[2], Hikaru Sakamoto[2] and Kazuo Sakiyama[2]

[1] NTT Secure Platform Laboratories
sasaki.yu@lab.ntt.co.jp
[2] The University of Electro-Communications
{liyang,sakiyama}@uec.ac.jp

**Abstract.** In this paper, we propose a new technique for SQUARE Differential Fault Analysis (DFA) against AES that can recover a secret key even with a large number of noisy fault injections, while the previous approaches of the SQUARE DFA cannot work with noise. This makes the attack more realistic because assuming the 100% accuracy of obtaining intended fault injections is usually impossible. Our success lies in the discovery of a new mechanism of identifying the right key guess by exploiting the coupon collector's problem and its variant. Our attack parameterizes the number of noisy fault injections. If the number of noisy faults is set to 0, the analysis becomes exactly the same as the previous SQUARE DFAs. Then, our attack can work even with a large number of noisy faults. Thus our work can be viewed as a generalization of the previous SQUARE DFAs with respect to the number of tolerable noisy fault injections.

**Keywords:** AES, Fault analysis, DFA, Noisy fault model, SQUARE DFA, Coupon collector's problem

## 1 Introduction

Fault analysis is one of the major approaches of the side-channel analysis. In particular, Differential Fault Analysis (DFA) has been applied to very wide range of ciphers. The suitable fault injection rounds for DFA (DFA round) are almost uniquely determined depending on the structure of the target cipher. These DFA rounds are the first place to be protected for the DFA countermeasures. A common approach is to take redundancy in hardware cost or time to trade the ability of detecting faulty calculations [1–3]. When the faults are injected before the DFA rounds, the complexity of the key recovery process usually becomes impractical in a straightforward DFA application.

For the AES-128 encryption [9], many DFA results [4–8] imply that the DFA rounds are 8 and 9. Even after the countermeasure is proposed so that rounds 8 and 9 are protected, researchers have proposed practical DFA variants based

**Table 1.** Comparison of SQUARE DFA.

| Approach | $\alpha$ | #Tolerable noise | Time | #Fault Injections |
|---|---|---|---|---|
| [10] | 256 | 0 | $2^{36.68}$ | 255 |
| [12] | 44 | 0 | $2^{34.14}$ | 43 |
| Ours | 256 | 1610 | $2^{44.54}$ | 59712 |
| | 128 | 49 | $2^{41.15}$ | 5664 |
| | 64 | 13 | $2^{39.94}$ | 2464 |

on the fault injected at round 7 such as SQUARE DFA [10], impossible DFA [10, 11], and Meet-in-the-Middle (MitM) DFA [11].

The SQUARE DFA exploits a property of the AES algorithm, *i.e.*, if the attacker collects 256 ciphertexts where one byte at the beginning of round 7 takes all 256 distinct values and other bytes are fixed to a constant, in a few rounds after, all bytes take exactly 256 distinct values with probability 1. The illustration of the property is available in Fig. 1. Then, the attacker guesses the last round key and performs a partial inverse computations for 256 ciphertexts. The correctness of the key guess is verified by checking whether or not 256 distinct values are observed. The attack requires to obtain 255 distinct fault values on a target byte with exactly 255 fault injections. Hence, the attack cannot accept the noise, and [10] assumed the bit-fault model.

Kim improved the SQUARE DFA based on the observation that 256 distinct values are not necessary to recover the key [12]. The attacker collects only $\alpha$ distinct values at the beginning of round 7, and thus the number of fault injections is reduced and the attack model is relaxed to the random-byte fault model. The attack requires to obtain $\alpha - 1$ distinct fault values on a target byte with exactly $\alpha - 1$ fault injections. Hence, the attack cannot accept the noise.

In practice, fault injections cannot assure the 100% accuracy of the intended fault injections. As a result, it is necessary to introduce the *noisy fault model.*, where random faulty ciphertexts corresponding to unintended fault injections are mixed in the data to be analyzed for recovering a key.

**Our Contributions.** We propose a new technique for the SQUARE DFA, which can recover the key even with a large number of noisy fault injections.

Assume that an attacker can inject an intended fault every two trials. First of all, the attacker obtains one ciphertext without injecting the fault. In order to collect $\alpha - 1$ intended faulty ciphertexts, the same number of noisy ciphertexts are obtained, and the total number of data to be analyzed is $n = 2(\alpha - 1) + 1$. Our attack still can find the key with such a high probability of obtaining noise at a small additional cost. See Table 1 for the comparison with previous work.

Our attack collects $\alpha$ distinct values at the beginning of round 7. However, the attacker also obtains $n - \alpha$ unintended ciphertexts due to the noise, and she does not know which are intended ones. For the right key guess, at least $\alpha$ distinct values appear after the partial decryption. If the probability of this

event is small enough, the key space is reduced. The probability of this event is estimated by solving the coupon collector's problem and its variant, Let $\beta$ be the size of each cell of the cipher ($\beta = 256$ for AES). The probability that a key guess is regarded as a right key candidate, *i.e.*, $\alpha$ distinct values appear in one byte after the partial decryption is equivalent to the probability where at least $\alpha$ out of $\beta$ coupons are collected with $n$ coupon drawing events.

## 2 Related Work

### 2.1 Specification of AES

AES-128 [9] is a 128-bit block cipher using a 128-bit key. It consists of 10 rounds. Subkeys $sk_i$ for round $i$ are generated from the original secret key $K$. At first, subkey $sk_0$ is XORed to the plaintext. Then a round function $AK \circ MC \circ SR \circ SB$ is iteratively applied, where $SB$ is an S-box transformation, $SR$ is a byte shift, $MC$ is a multiplication by an MDS matrix, and $AK$ is a subkey XOR. Note that $MC$ is omitted for the last round. We denote the initial state in round $x$ by $\#x^I$. Then, states immediately after each operation in round $x$ are denoted by $\#x^{SB}$, $\#x^{SR}$, $\#x^{MC}$, and $\#x^{AK}$. Obviously, $\#x^{AK}$ is identical with $\#(x+1)^I$. We denote 4-byte positions in column $j$ of state $\#S$ by $\#S[Col(j)]$.

### 2.2 Square DFA on AES

Phan and Yen showed that the key is recovered with the fault injected at the beginning of round 7 [10]. The attack uses the idea of the Square attack, which was firstly proposed by Daemen *et al.* for the Square cipher.

The attack collects 256 distinct values on a single byte at the beginning of round 7. The attacker firstly obtains a ciphertext. While the same plaintext is encrypted 255 times. she injects a fault on a target byte. The fault model by Phan and Yen [10] is summarized as follows.

**Fault model 1 (bit-fault and deterministic-fault)**
- *An attacker can inject the fault in any intended bit position, and moreover, several target bits can be flipped simultaneously.*
- *The fault injection to the target bits always succeeds.*

All in all, the attacker can obtain 255 distinct fault values with 255 injections without obtaining any noisy data. 256 distinct values for a single byte at $\#7^I$ result in 256 distinct values for all bytes at $\#9^{AK\prime}$ as shown in Fig. 1. Note that the order of the linear operations are exchanged in rounds 9 and 10.

After 256 ciphertexts following the data structure of Fig. 1 are obtained, the attacker recovers the converted last round key $sk'_{10}$, where $sk'_{10} = SR^{-1}(sk_{10})$. The key recovery phase is described in Fig. 2. At first, the inverse of $SR$ is applied to all ciphertexts. Then, the attacker guesses $sk'_{10}$ column by column. Let us focus our attention on column 0, which is colored in Fig. 2. $2^{32}$ value of $sk'_{10}[Col(0)]$ are exhaustively guessed. For each guess, the attacker decrypts $\alpha = 256$ converted ciphertexts $(C'_0, \ldots, C'_{255})$ up to $\#9^{AK'}[Col(0)]$. Then, the right key can be identified based on the following mechanism.

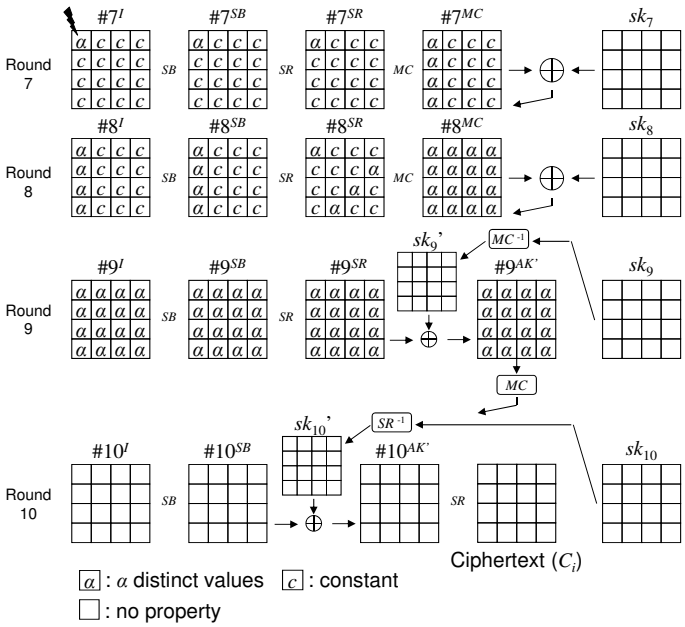


**Fig. 1.** Fault injection at $\#7^I$. The key is a fixed value but the notation $c$ is omitted.

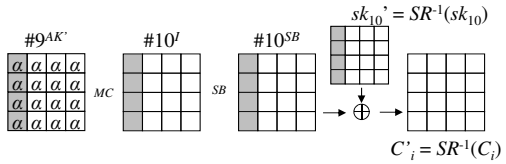


**Fig. 2.** The structure of the key recovery phase.

**Key Recovery Mechanism 1:** *For the right guess,* 256 *distinct values appear for each byte of* $\#9^{AK'}[Col(0)]$*. This is unlikely to occur for wrong guesses. In fact, the probability of this event for a wrong guess is*

$$\left(\prod_{i=0}^{255} \frac{(256-i)}{256}\right)^4 \approx 0. \tag{1}$$

Finally, the right key value is recovered with 255 fault injections and about $2^{32} \cdot 256 \cdot 1/10 \approx 2^{36.68}$ AES computations.

### 2.3 Improved SQUARE DFA with a Small Number of Fault Injections

Kim proposed an improved SQUARE DFA on AES [12]. The entire attack structure again follows Fig. 1. Suppose that the attacker obtains $\alpha$ distinct values

at the target single byte at $\#7^I$, where $1 \leq \alpha \leq 256$. The only difference from the original SQUARE DFA [10] is the value of $\alpha$. Phan and Yen only considered $\alpha = 256$ while Kim regarded it as a variable ranging from 1 to 256. Due to the same reason as [10], each byte at $\#9^{AK'}$ takes $\alpha$ distinct values.

**Fault model 2 (random byte-fault and deterministic fault)**
- *An attacker can inject $\alpha - 1$ distinct fault values in an intended byte position. Fault value is uncontrollable and is uniformly distributed.*
- *The fault injection to the target byte always succeeds.*

For a fixed $\alpha$, the right key can be identified based on the following mechanism.

**Key Recovery Mechanism 2:** *For the right guess, $\alpha$ distinct values appear for each byte of $\#9^{AK'}[Col(0)]$. If this event is unlikely to occur for wrong guesses, the right key can be identified. The probability of this event for a wrong guess is*

$$\Big(\prod_{i=0}^{\alpha-1} \frac{(256-i)}{256}\Big)^4. \tag{2}$$

Kim showed that when $\alpha = 44$, Eq. (2) is smaller than $2^{-32}$ and thus the right key is recovered. Finally, the right key value is recovered with 43 fault injections and about $2^{32} \cdot 44 \cdot 1/10 \approx 2^{34.14}$ AES computations.

## 3 SQUARE DFA Based on Coupon Collector's Problem

We generalize the previous SQUARE DFA to accept the noise. Our attack has two parameters. 1) how many distinct values can be obtained at $\#7^I$, $\alpha$, and 2) how many ciphertexts are obtained before $\alpha$ intended ciphertexts are collected, $n$. Note that $n$ is the sum of the numbers of intended and unintended ciphertexts. If we set the number of noisy fault to 0 ($n = \alpha$), the analysis becomes the same as [12]. If we further set $\alpha = 256$, the analysis becomes the same as [10]. Hence, we call our approach generalized SQUARE DFA. For the further generalization, we also parameterize the size of each cell by $\beta$, e.g. $\beta = 256$ for AES.

### 3.1 Overview

The parameters of the attacker's abilities $\alpha$ and $n$ basically depend on the environment e.g., how much cost can be spent for the attack. We assume that the attacker's ability is independently measured before the analysis is started.

**Fault model 3 (random byte-fault and probabilistic-fault)**
- *An attacker can inject at most $\alpha - 1$ distinct fault values in an intended byte position, The fault value is uncontrollable and is uniformly distributed.*
- *The intended fault injection can be obtained only probabilistically, with a probability of $(\alpha - 1)/(n - 1)$.*

The attack firstly obtains 1 ciphertext without the fault injection. While the same plaintext is encrypted $n-1$ times, the attacker injects the fault $n-1$ times and obtains the corresponding $n-1$ ciphertexts. $\alpha-1$ intended faulted ciphertexts are included in the $n-1$ faulted ciphertexts, but which ciphertexts are intended ones is unknown. After collecting such $1+(n-1)=n$ ciphertexts, we recover the last round key $sk'_{10}$ column by column.

> **Key Recovery Mechanism 3:** *For the right guess, at least $\alpha$ distinct values appear for each byte of $\#9^{AK'}[Col(0)]$. If this event is unlikely to occur for wrong guesses, the right key can be identified.*

Therefore, for each guess of $sk'_{10}[Col(0)]$, if less than $\alpha$ distinct values are observed in at least one byte of $\#9^{AK'}[Col(0)]$, we know the guess is wrong, and thus the key space is reduced. Let $P(\alpha,\beta,n)$ be the probability that, for each key guess, at least $\alpha$ distinct values appear for one byte of $\#9^{AK'}[Col(0)]$. The key space of $sk'_{10}[Col(0)]$ is reduced from $\beta^4$ to $\beta^4 \times P(\alpha,\beta,n)^4$ after the analysis.

For simplicity, let us discuss the case where $\alpha=\beta$, which implies that all values are collected at $\#7^I$. For AES, $\alpha=\beta=256$. In this case, $P(\alpha,\beta,n)$ is equivalent to the success probability of the coupon collector's problem.

> *There are $\beta$ coupons. With 1 coupon-drawing event, 1 coupon which is uniformly distributed from $\beta$ coupons is obtained. The success probability of the coupon collector's problem for parameters $(\beta,n)$ is the one that all $\beta$ coupons are completed with $n$ coupon-drawing events.*

For $\alpha<\beta$, $P(\alpha,\beta,n)$ is equivalent to a variant of the coupon collector's problem where at least $\alpha$ kinds out of $\beta$ kinds of coupons are collected with $n$ coupon-drawing events. In the next section, we will evaluate the value of $P(\alpha,\beta,n)$.

### 3.2 Probability Evaluation of $P(\alpha,\beta,n)$

**Proposition 1** *Let $Q(\alpha,n)$ be the number of permutations of $n$ coupons including all of the $\alpha$ different coupons completely $(1\le\alpha\le n)$. Then*

$$Q(\alpha,n)=\sum_{k=1}^{\alpha}\left((-1)^{\alpha-k}\binom{\alpha}{k}k^n\right). \tag{3}$$

*Proof.* As $Q(j,n)\binom{\alpha}{j}$ covers all the possible ways to collect $j$ different coupons chosen from $\alpha$ coupons $(j\le\alpha)$ using $n$ random coupon-drawing trials, its summation for $j=1,2,\ldots,\alpha$, where $\alpha=0,1,\ldots,n$, always becomes $\alpha^n$ as

$$\sum_{k=1}^{\alpha}\left(Q(k,n)\binom{\alpha}{k}\right)=\alpha^n, \tag{4}$$

For $a_n=\sum_{k=1}^{n}\binom{n}{k}b_k$, we have $b_n=\sum_{k=1}^{n}(-1)^{n-k}\binom{n}{k}a_k$. Therefore for Eq. (4), consider $\alpha^n$ as $a_n$ and $Q(k,n)$ as $b_k$, we can derive Eq. (3). □

**Proposition 2** *Let $P(\alpha, \beta, n)$ be the probability that one collects at least $\alpha$ out of $\beta$ coupons through $n$ trials, where $2 \leq \alpha \leq \beta$. Then*

$$P(\alpha, \beta, n) = \binom{\beta}{\alpha}\binom{\alpha}{1}\sum_{i=\alpha}^{n}\frac{Q(\alpha-1, i-1)}{\beta^i}, \tag{5}$$

*Proof.* We have the probability that one collects $\alpha$ coupons out of $\beta$ exactly at the $i$-th trial, *i.e.*, $\alpha - 1$ coupons are collected through the $i-1$ trials, as

$$\frac{Q(\alpha-1, i-1)\binom{\beta}{\alpha-1}\binom{\beta-\alpha+1}{1}}{\beta^i} = \binom{\beta}{\alpha}\binom{\alpha}{1}\frac{Q(\alpha-1, i-1)}{\beta^i}.$$

As the probability of $P(\alpha, \beta, n)$ equals the summation of the above probabilities for $i = \alpha, \alpha+1, \ldots, n$, Eq. (5) is derived. □

## 4 Details of the Generalized SQUARE DFA on AES-128

The parameters $(\alpha, n)$ should be measured depending of the attacker's ability. The parameter $\beta$ is fixed to 256. In this section, we explain the attack with the parameter $(\alpha, \beta, n) = (256, 256, 1553)$, where $P(\alpha, \beta, n) = 1/2$ according to Eq. (5). Note that $(\alpha, n) = (256, 1553)$ is just an example for the case study.

The attack consists of the data collecting phase and the key recovery phase. The procedure of the data collecting phase is as follows.

1. Obtain a ciphertext $C_0$ without injecting the fault, and store it in a table.
2. While the same plaintext is encrypted, try to inject a fault in a target single byte at $\#7^I$. If the obtained ciphertext does not overlap with already stored ones in a table, add this ciphertext into a table. Otherwise, discard it.
3. Repeat Step 2 until $n$ ciphertexts $C_0, \ldots, C_{n-1}$ are obtained.

For one plaintext, a set of $n$ ciphertexts is constructing. For $(\alpha, n) = (256, 1553)$, we collect 8 sets of $n$ ciphertexts by using 8 different plaintexts.

The input to the key recovery phase is 1553 ciphertexts $C_0, \ldots, C_{1552}$, where 256 values are collected in the target single byte at $\#7^I$. Then, do as follows.

1. Compute $C'_i \leftarrow SR^{-1}(C_i)$ for $0 \leq i \leq 1552$.
2. For each column ($j = 0, 1, 2, 3$), exhaustively guess the value for $sk'_{10}[Col(j)]$ and compute $\#9^{AK'}[Col(j)]$.
3. If only less than 256 distinct values are observed in at least one byte of $\#9^{AK'}[Col(j)]$, discard the guessed $sk'_{10}[Col(j)]$ from the key candidates.
4. Repeat steps 1 to 3 by 8 times with changing the ciphertext set.

The attack cost is evaluated as follows. The number of the faulty ciphertexts is $8 \cdot 1553 = 12424$. In the key recovery phase, Step 1 costs $8 \cdot 1553$ inverse $SR$ computations, which essentially does not cost anything. Step 2 costs $8 \cdot 1553 \cdot 2^{32}$ one round computations, which is $(8 \cdot 1553 \cdot 2^{32})/10 \approx 2^{42.28}$ AES computations. The memory requirement is $2^{32}$ AES state to count the remaining key space.

The time complexity can be optimized. Once a key candidate is identified to be a wrong key, it does not have to be examined again. Therefore, the complexity is $1553 \cdot (2^{32} + 2^{28} + 2^{24} + \cdots + 2^4)/10 \approx 2^{39.37}$ AES computations.

If not only for the ciphertext, but also the plaintext is available for the attacker, she can combine the brute force attack. Hence, step 4 can be stopped when the key space is reduced to a sufficiently small size, rather than 1.

## 5 Concluding Remarks

In this paper, we presented the new fault analysis called generalized SQUARE DFA, which was an extension of the previous SQUARE DFA with respect to the noisy fault injections. The probability that a key candidate is judged as a correct key is estimated with a coupon collector's problem and its variant.

## References


1. Fischer, W.: Aspects of the development of secure and fault-resistant hardware. In: FDTC. (2008) 18–22
2. Guilley, S., Sauvage, L., Danger, J.L., Selmane, N.: Fault injection resilience. In: FDTC. (2010) 51–65
3. Satoh, A., Sugawara, T., Homma, N., Aoki, T.: High-Performance Concurrent Error Detection Scheme for AES Hardware. In Oswald, E., Rohatgi, P., eds.: CHES 2008. Volume 5154 of LNCS., Springer (2008) 100–112
4. Blömer, J., Seifert, J.P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In Wright, R.N., ed.: FC 2003. Volume 2742 of LNCS., Springer (2003) 162–181
5. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In Zhou, J., Yung, M., Han, Y., eds.: ACNS 2003. Volume 2846 of LNCS., Springer (2003)
6. Giraud, C.: DFA on AES. In Dobbertin, H., Rijmen, V., Sowa, A., eds.: AES Conference 2004. Volume 3373 of LNCS., Springer (2004) 27–41
7. Moradi, A., Shalmani, M.T.M., Salmasizadeh, M.: A Generalized Method of Differential Fault Attack Against AES Cryptosystem. In Goubin, L., Matsui, M., eds.: CHES 2006. Volume 4249 of LNCS., Springer (2006) 91–100
8. Piret, G., Quisquater, J.J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In Walter, C.D., Çetin Kaya Koç, Paar, C., eds.: CHES 2003. Volume 2779 of LNCS., Springer (2003) 77–88
9. Daemen, J., Rijmen, V.: AES Proposal: Rijndael. (1998)
10. Phan, R.C.W., Yen, S.M.: Amplifying Side-Channel Attacks with Techniques from Block Cipher Cryptanalysis. In Domingo-Ferrer, J., Posegga, J., Schreckling, D., eds.: CARDIS 2006. Volume 3928 of LNCS., Springer (2006) 135–150
11. Derbez, P., Fouque, P.A., Leresteux, D.: Meet-in-the-Middle and Impossible Differential Fault Analysis on AES. In Preneel, B., Takagi, T., eds.: CHES 2011. Volume 6917 of LNCS., Springer (2011) 274–291
12. Kim, C.H.: Efficient methods for exploiting faults induced at AES middle rounds. Cryptology ePrint Archive, Report 2011/349 (2011)