# Improved Differential Fault Analysis on ARIA using Small Number of Faults ⋆

Yuseop Lee [a], Kitae Jeong [a], Jaechul Sung [b], Seokhie Hong [a,*]

[a] *Center for Information Security Technologies (CIST), Korea University, Republic of Korea*

[b] *Department of Mathematics, University of Seoul, Republic of Korea*

**Abstract**

In [15], Li et al. firstly proposed a differential fault analysis on ARIA-128. This attack requires average 45 random byte fault injections. In 2012, Park et al. proposed the improve DFA by using 33 random byte fault injection. Also Kim proposed differential fault analysis based on multi byte fault model. In this model, the number of fault injections is reduce to 13 and If access to the decryption oracle is allowed, only 7 faults are required. In this paper, we propose improved differential fault analysis on ARIA. Based on random byte fault model, the proposed attacks can recover the secret key of ARIA-128/192/256 by using 6 fault injections within a few minutes. Moreover, in cases of ARIA-128 and ARIA-256, it is possible to recover the secret key using only 4 fault injections under a fault assumption where an attacker can induce some faults during both encryption and decryption process, respectively. Our results on ARIA-192/256 are the first known DFA results on them.

*Key words:* Differential fault analysis, Block cipher, ARIA, Cryptanalysis.

Differential fault analysis (DFA) is one of the most well-known side-channel analysis on block ciphers. After Biham and Shamir first proposed a DFA on DES [1], it was applied to AES [2,8,11,18,20,**?**], Triple-DES [9], CLEFIA [4], SEED [10], ARIA [15,19,13], SMS4 and MacGuffin [16] and so on. In DFA,

⋆ This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program(NIPA-2012-H0301-12-3007) supervised by the NIPA(National IT Industry Promotion Agency.

* Corresponding author.

*Email addresses:* yusubi@korea.ac.kr (Yuseop Lee), kite.jeong@gmail.com (Kitae Jeong), jcsung@uos.ac.kr (Jaechul Sung), hsh@cist.korea.ac.kr (Seokhie Hong).

an attacker induces some faults to registers of the target round and obtains the corresponding faulty/right ciphertexts. Then the candidates of last round key can be recovered by using differential cryptanalysis. By using them, he can compute the input value of the last round. Thus, he repeats the above procedure to recover more round keys until the secret key is obtained by the key schedule.

ARIA is a 128-bit block cipher supporting 128/192/256-bit secret keys [14]. It was adopted as a Korean Standard block cipher algorithm (KS X 1213). Until now, many cryptanalytic results on this algorithm were proposed [3,5,6,17,21]. First DFA on ARIA-128 proposed in [15]. The attack procedure of this attack is as follows. First, the last four round keys are recovered by using 45 random byte fault injections. Then, the 128-bit secret key is computed by using the recovered round keys. In [19], Park et al. proposed improve DFA on ARIA-128. Thus they reduce the number of fault injections to 33. Also, Kim proposed DFA on ARIA-128 based on multi byte fault model[13]. Thus they use only 13 fault injections. If access to the decryption oracle is allowed, only 7 faults are required.

In this paper, we proposed improved DFAs on ARIA. In our attack, we consider two random byte fault assumptions $A^1$ and $A^2$. Assumption $A^2$ is a usual random byte fault assumption. That is, an attacker does not know the location and value of faults. Under assumption $A^1$, he can control the location of faults but he does not know the value of it. The assumption $A^1$ is based on the result of [7]. In [7], the authors presented that it is possible to control the location of fault injections to ISO/IEC 18033-3 block ciphers such that AES, DES, Camellia, CAST-128, SEED and MISTY1. From this result, it is reasonable that we can control the location of fault injection to ARIA. Furthermore, we subdivide each assumption into $A^i_E$ and $A^i_{DE}$, respectively $(i = 1, 2)$. $A^i_E$ means a fault assumption where a fault injection happens during only the encryption process. $A^i_{DE}$ means a fault assumption where a fault injection happens during the decryption or encryption process, respectively. For example, a fault assumption considered in [15,19] is $A^2_E$.

On the other hand, in [20], the authors constructed differential equations to recover the secret key of AES. Thus, this attack requires only one random byte fault injection. To reduce the number of required fault injections, we apply this idea to ARIA-128/192/256. As a result, our attacks require 6 fault injections under $A^i_E$. In detail, we induce some faults to the input registers of target rounds. Then, we construct differential equations from the differential propagation of injected faults. Thus, by using these equations, we recover the last four round keys. From the key schedule of ARIA, we finally recover the secret key of ARIA by using these round keys. In the case of $A^i_{DE}$, we can recover the secret keys of ARIA-128/256 with only 4 fault injections. Our results on ARIA are summarized in the Table 1.

Table 1
DFA results on ARIA

| Target algorithm | Fault assumption | ♯ of injected faults | Reference |
|---|---|---|---|
| ARIA-128 | $A_E^2$ | 45 | [15] |
| | $A_E^2$ | 33 | [19] |
| | $A_E^{2*}$ | 13 | [13] |
| | $A_{DE}^{2*}$ | 7 | [13] |
| | $A_E^1, A_E^2$ | 6 | Section 5.1, Section 5.2 |
| | $A_{DE}^1, A_{DE}^2$ | 4 | Section 5.3 |
| ARIA-192 | $A_E^1, A_E^2$ | 6 | Section 5.1, Section 5.2 |
| | $A_{DE}^1, A_{DE}^2$ | 4 | Section 5.3 |
| ARIA-256 | $A_E^1, A_E^2$ | 6 | Section 5.1, Section 5.2 |
| | $A_{DE}^1, A_{DE}^2$ | 4 | Section 5.3 |

$*$ multi byte fault model

The difference between $A^1$ and $A^2$ is the computational complexity. In detail, under assumption $A^1$, the computational complexity is $O(2^{24})$. In the case of $A^2$, the computational complexity is $O(2^{32})$. As a simulation result, our attacks under $A^1$ can recover the secret key of ARIA within a few seconds. But, in the case of $A^2$, our attacks requires a few minutes. This paper is organized as follows. In Section 2, we briefly introduce ARIA. In Section 3, we present our fault assumptions and the basic idea of our attacks. The method to compute round keys by using differential equations is described in Section 4. In Section 5, we present the detailed attack procedure of our attacks on ARIA. Finally, we give our conclusion in Section 6.

## 1 Description of ARIA

ARIA is a 128-bit block cipher which supports 128-, 192- and 256-bit secret keys. We call this algorithm ARIA-128/192/256, respectively. ARIA-128/192/256 consist of 12, 14 and 16 rounds, respectively. Throughout this paper, the following notations are used.

- $K_n$: the secret key of ARIA-$n$ ($n = 128, 192, 256$).
- $r$: the number of round of ARIA ($r = 12, 14, 16$).
- $X_i$: the input value of round key addition in round $i$ ($i = 1, 2, \cdots, r$).
- $Y_i$: the input value of substitution layer in round $i$.
- $Z_i$: the input value of diffusion layer in round $i$.
- $\lll$ ($\ggg$) $t$: a left(right) circular rotation of operand by $t$ bits.

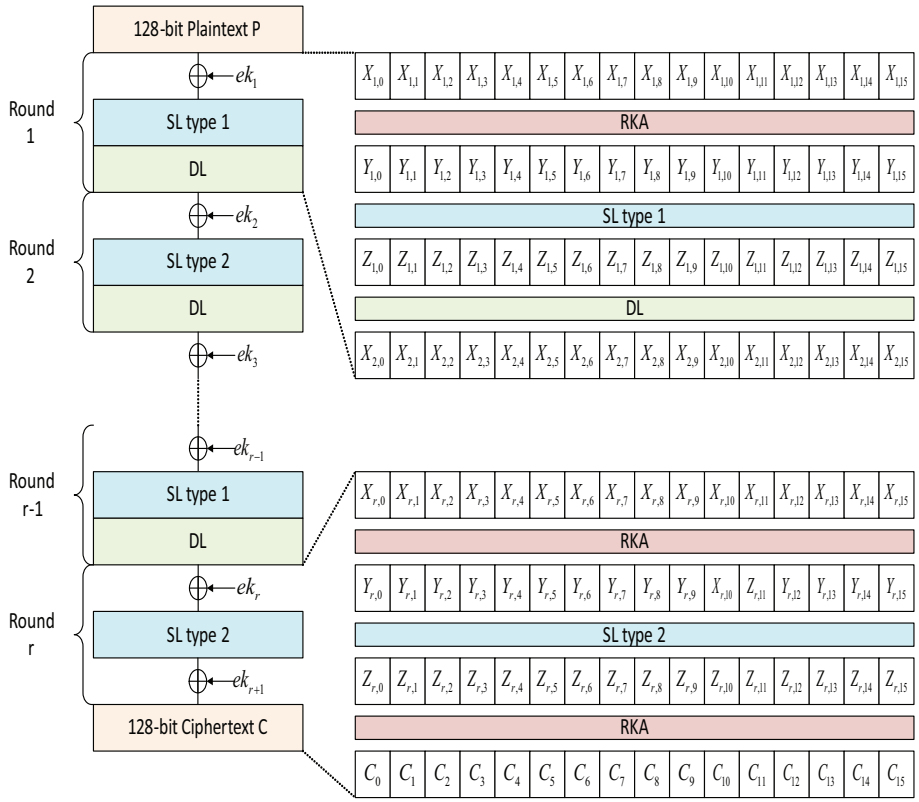


Fig. 1. The structure of ARIA

Each round function is composed of three subfunctions such as Round Key Addition (RKA), Substitution Layer (SL) and Diffusion Layer (DL). In last round, DL is replaced by RKA. They are defined as follows. See [14] for detailed descriptions of them (see the Figure 1).

- RKA is a XORing of the state and a 128-bit round key which is derived from the secret key.
- SL is a non-linear byte substitution operation applied all bytes of state in parallel. In odd rounds and even rounds, two different types of SL are operated.
- DL is a linear matrix multiplication of the state with a 1616 binary matrix.

The key schedule of ARIA consists of an initialization part and a round key generation part. First, a 256-bit $(KL, KR)$ is filled with $K_n$ as follows.

$$KL||KR = K_n||0 \cdots 0.$$

In an initialization part, four 128-bit values $(W_0, W_1, W_2, W_3)$ are computed as follows. Here, $F_o$ and $F_e$ are the even and odd round function, respectively.

Here, $CK_i$ is a 128-bit constant.

$$\begin{aligned} W_0 &= KL, \\ W_1 &= F_o(W_0, CK_1) \oplus KR, \\ W_2 &= F_e(W_1, CK_2) \oplus W_0, \\ W_3 &= F_o(W_2, CK_3) \oplus W_1. \end{aligned}$$

In a round key generation part, $r+1$ round keys $(ek_1, \cdots, ek_{r+1})$ used in the encryption process are generated as follows.

$$\begin{aligned} ek_1 &= W_0 \oplus W_1^{\gg 19},\ ek_2 = W_1 \oplus W_2^{\gg 19},\ ek_3 = W_2 \oplus W_3^{\gg 19}, \\ ek_4 &= W_3 \oplus W_0^{\gg 19},\ ek_5 = W_0 \oplus W_1^{\gg 31},\ ek_6 = W_1 \oplus W_2^{\gg 31}, \\ ek_7 &= W_2 \oplus W_3^{\gg 31},\ ek_8 = W_3 \oplus W_0^{\gg 31},\ ek_9 = W_0 \oplus W_1^{\ll 61}, \\ ek_{10} &= W_1 \oplus W_2^{\ll 61},\ ek_{11} = W_2 \oplus W_3^{\ll 61},\ ek_{12} = W_3 \oplus W_0^{\ll 61}, \\ ek_{13} &= W_0 \oplus W_1^{\ll 31},\ ek_{14} = W_1 \oplus W_2^{\ll 31},\ ek_{15} = W_2 \oplus W_3^{\ll 31}, \\ ek_{16} &= W_3 \oplus W_0^{\ll 31},\ ek_{17} = W_0 \oplus W_1^{\ll 19}. \end{aligned}$$

## 2 Fault assumptions and the basic idea

### *2.1 Fault assumptions*

Our fault assumption is based on a random byte fault model. In general, under random byte fault model, an attacker induces some byte faults into the target rounds and does not know the exact location of injected faults. However, in [7], it is shown that it is possible to control the location a fault injection to ISO/IEC 18033-3 block ciphers. They considered AES, DES, Camellia, CAST-128, SEED and MISTY1. Though ARIA is not included in the list, we expect that the result of [7] is also applicable to ARIA. Thus, we consider the following two assumptions.

- Assumptions $A^1$

(1) An attacker can induce some random byte faults to the input register in the target round.
(2) The attacker **can control the location** of injected faults.
(3) The value of faults is unknown.

- Assumptions $A^2$

(1) An attacker can induce some random byte faults to the input register in the target round.
(2) The **location** and value of injected faults are both **unknown**.

Furthermore, $A^i$ is partitioned into two assumptions $A^i_E$ and $A^i_{DE}$, respectively $(i = 1, 2)$. $A^i_E$ means a fault assumption where a fault injection happens during only the encryption process. $A^i_{DE}$ means a fault assumption where a fault injection happens during the decryption and encryption process, respectively. For example, $A^1_E$ and $A^1_{DE}$ are defined as follows.

- Assumptions $A^1_E$

(1) An attacker can induce some random byte faults to the input register in the target round **during the encryption process**.
(2) The attacker can control the location of injected faults.
(3) The value of faults is unknown.

- Assumptions $A^1_{DE}$

(1) An attacker can induce some random byte faults to the input register in the target round **during the decryption and encryption process, respectively**.
(2) The attacker can control the location of injected faults.
(3) The value of faults is unknown.

Since ARIA satisfies the involutional property, the encryption process is equal to the decryption process except for the round keys. Thus, it is reasonable that we consider both processes.

### *2.2 Constructing differential equations*

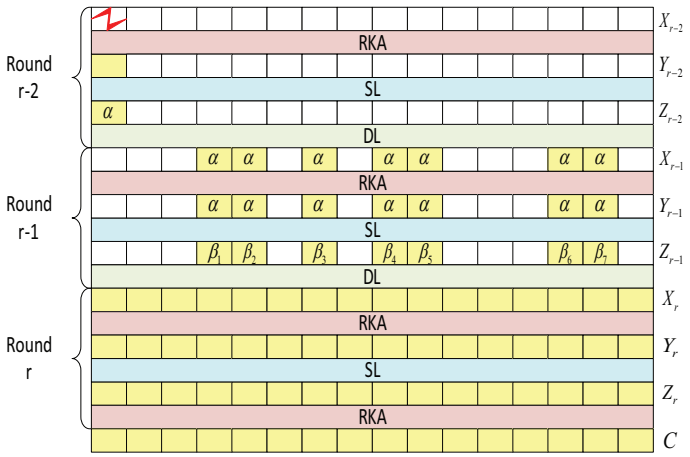


Fig. 2. The differential propagation where a fault is injected to $X_{r-2,0}$

When a random byte fault is induced to the first byte of input register $X_{r-2,0}$ of round $r-2$, the differential propagation is shown in Figure 2. From this figure, the input difference $\Delta X_{r-1}$ of round $r-1$ has the following pattern. Here, $\alpha$ is the first byte difference of SL in round $r-2$.

$$\Delta X_{r-1} = (0, 0, 0, \alpha, \alpha, 0, \alpha, 0, \alpha, \alpha, 0, 0, 0, \alpha, \alpha, 0).$$

That is, $\Delta X_{r-1}$ has the same 7 difference bytes and 9 zero difference bytes. Then $\Delta Z_{r-1}$ has the following pattern. Here, $\beta_i$ is a nonzero difference ($i = 1, \cdots, 7$).

$$\Delta Z_{r-1} = (0, 0, 0, \beta_1, \beta_2, 0, \beta_3, 0, \beta_4, \beta_5, 0, 0, 0, \beta_6, \beta_7, 0).$$

Then, we can construct differential equations on $\Delta Y_r$ by using $\Delta Z_{r-1}$. Let $M$ be a matrix derived from DL. Then, $\Delta Z_{r-1}$ and $\Delta X_r$ satisfy the following equation.

$$\Delta X_r = M \cdot (\Delta Z_{r-1})^T.$$

Since $M$ is an involutory matrix, that is $M^{-1} = M$, the above equation can be rewritten as follows.

$$\Delta Z_{r-1} = M \cdot (\Delta X_r)^T.$$

In our attack, we consider only positions of zero differences in $\Delta Z_{r-1}$. That is, $\Delta Z_{r-1,0}, \Delta Z_{r-1,1}, \Delta Z_{r-1,2}, \Delta Z_{r-1,5}, \Delta Z_{r-1,7}, \Delta Z_{r-1,10}, \Delta Z_{r-1,11}$, $\Delta Z_{r-1,12}$ and $\Delta Z_{r-1,15}$. Thus, we consider the following matrix $M_0$. Here, $M_0$ consists of 0-, 1-, 2-, 5-, 7-, 10-, 11-, 12-, 15-th rows of $M$.

$$M_0 = \begin{pmatrix}
0&0&0&1&1&0&1&0&1&1&0&0&0&1&1&0\\
0&0&1&0&0&1&0&1&1&1&0&0&1&0&0&1\\
0&1&0&0&1&0&1&0&0&0&1&1&1&0&0&1\\
0&1&0&1&1&0&0&0&0&1&1&0&0&0&1&1\\
0&1&0&1&0&0&1&0&1&0&0&1&1&1&0&0\\
0&0&1&1&0&1&1&0&1&0&0&0&0&1&0&1\\
0&0&1&1&1&0&0&1&0&1&0&0&1&0&1&0\\
0&1&1&0&0&0&1&1&0&1&0&1&1&0&0&0\\
0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&1
\end{pmatrix}.$$

Then $\Delta X_r$ satisfies the following equation.

$$M_0 \cdot (\Delta X_r)^T = (00\cdots0)^T.$$

Also, since $\Delta X_r$ is equal to $\Delta Y_r$, we can construct the following 9 differential equations.

$$M_0 \cdot (\Delta Y_r)^T = (00\cdots0)^T.$$

Note that all equations do not include $\Delta Y_{r,0}$.

Recall that 7 bytes of $\Delta Y_{r-1}$ have the same difference value. By using this property, we can easily obtain six additional differential equations. Hence, when a random byte fault is induced to $X_{r-2,0}$, we can construct total fifteen differential equations. When a random byte fault is injected to other bytes of $X_{r-2}$, we can obtain fifteen differential equations under other cases in a similar fashion.

### *2.3 Recovery of the secret key from round keys*

We assume that four consecutive round keys are obtained under assumption $A_E^i$. Then, it is known that the secret key of ARIA can be easily computed by using linear equations. For example, in the case of ARIA-128, $W_0$ satisfies the following equation from the key schedule.

$$W_0 \oplus W_0^{\lll 86} = ek_{13} \oplus ek_{10}^{\lll 31} \oplus ek_{11}^{\lll 92} \oplus ek_{12}^{\lll 25}.$$

Thus, if we know 512-bit $(ek_{10}, ek_{11}, ek_{12}, ek_{13})$, we can compute $W_0$. In detail, the left side of the above equation can be rewritten as $G \cdot W_0^T$. Here, $G$ is a $128 \times 128$ binary matrix with rank 126. Thus we can compute $2^2$ candidates of $W_0$ by using this linear equation. For each candidate of $W_0$, we can compute $(W_1, W_2, W_3)$ by using the following equations.

$$\begin{aligned} ek_{13} &= W_0 \oplus W_1^{\lll 31}, \\ ek_{10} &= W_1 \oplus W_2^{\lll 61}, \\ ek_{11} &= W_2 \oplus W_3^{\lll 61}. \end{aligned}$$

$(W_0, W_1, W_2, W_3)$ of ARIA-192/256 can be computed in a similar fashion.

For each candidate of four consecutive round keys, we obtained $2^2$ candidates of $(W_0, W_1, W_2, W_3)$ from the above procedure. Furthermore, we can discard wrong candidates of $(W_0, W_1, W_2, W_3)$ by using an initialization part of the key schedule. Recall that an initialization part conducts the following operations.

$$W_0 = KL, W_1 = F_o(W_0, CK_1) \oplus KR, \quad (1)$$
$$W_2 = F_e(W_1, CK_2) \oplus W_0, \quad (2)$$
$$W_3 = F_o(W_2, CK_3) \oplus W_1. \quad (3)$$

For each candidate of $(W_0, W_1, W_2, W_3)$, we check that Equation (1), (2) and (3) hold. In the case of ARIA-128, $KR = 0$ from the key schedule. Thus, the probability that a wrong candidate passes this test is $2^{-384} (= 2^{-128 \cdot 3})$. Similarly, in the cases of ARIA-192/256, the probability is $2^{-320}$ and $2^{-256}$, respectively. Hence we further reduce the number of candidate of $(W_0, W_1, W_2, W_3)$. The method to compute the secret key from $(W_0, W_1, W_2, W_3)$ is simple. From the key schedule, $KL$ is equal to $W_0$. And $KR$ is computed by using Equation (1).

Now, we consider assumption $A_{DE}^i$. We assume that the first round key and the last round key are obtained. In cases of ARIA-128/256, $W_1$ satisfies the following equation, respectively.

- ARIA-128: $W_1^{\lll 109} \oplus W_1^{\lll 31} = ek_1 \oplus ek_{13}$.
- ARIA-256: $W_1^{\lll 109} \oplus W_1^{\lll 19} = ek_1 \oplus ek_{17}$.

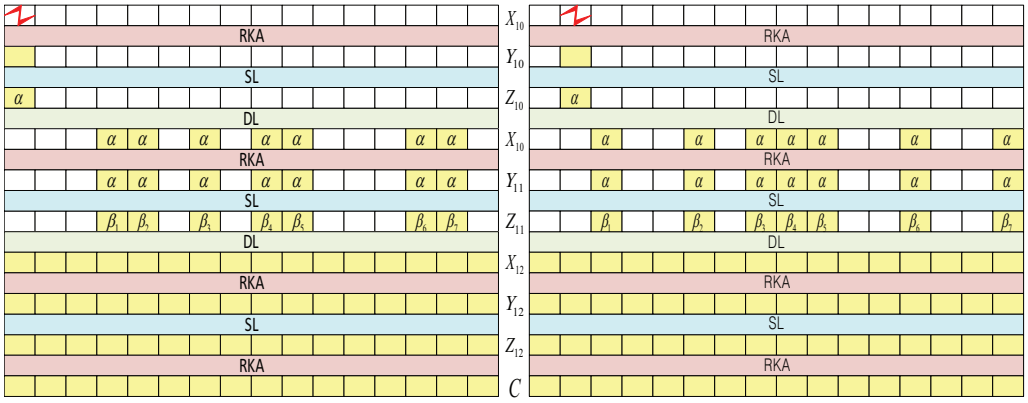


Fig. 3. In ARIA-128, two differential propagations where a fault is injected to $X_{10,0}$ and $X_{10,1}$

Similarly to the method to computing $W_0$ under assumption $A_E^i$, we can compute $2^2$ candidates of $W_1$ from the above equation. Then, we can also compute $W_0$ by using the following equation.

$$ek_1 = W_0 \oplus W_1^{\ggg 19}.$$

In the case of ARIA-128, we can reduce the number of candidates of $(W_0, W_1)$ by checking Equation (1). However, since we do not know KR in the case of ARIA-256, we need some trial encryptions.

## 3 Computation of round keys

In this section, we describe the method to obtaining round keys from differential equations considered in Section 3.2. We first consider assumption $A^1$. Under $A_E^1$, to recover the secret key, we need four consecutive round keys as mentioned in Section 3.3. However, we consider only the method to computing the last two round keys $(ek_r, ek_{r+1})$. When we obtain them, we repeat this procedure to compute $(ek_{r-2}, ek_{r-1})$. In the case of $A_{DE}^1$, we need to obtain $(ek_1, ek_{r+1})$. Thus, we compute $ek_{r+1}$ and $ek_1$ by using the above procedure in the encryption and decryption processes, respectively. Under $A^2$, we do not know the location of injected faults. Thus, we should guess it. Moreover, ARIA-128/192/256 have the same structure except for the number of rounds and the key schedule. Thus we concentrate on ARIA-128 in this section.

### *3.1 Assumption $A^1$*

Since methods to obtaining round keys from differential equations under $A_E^1$ and $A_{DE}^1$ are similar, we consider only $A_E^1$ here. First, an attacker chooses a plaintext $P$ and obtains one right ciphertext $C$. And he injects two random byte faults to

$Y_{10,0}$ and $Y_{10,1}$, and computes the corresponding two faulty ciphertexts $C^1$ and $C^2$, respectively (see Figure 3). From Section 3.2, we construct two matrices $M_0$ (corresponding to the event where a random byte fault is injected to $Y_{10,0}$) and $M_1$ (corresponding to the event where a random byte fault is injected to $Y_{10,1}$). Thus, we can obtain total 18 differential equations on $(\Delta Y^1_{12}, \Delta Y^2_{12})$. Note that $\Delta Y^1_{12}$ and $\Delta Y^2_{12}$ are related to $\Delta C^1 (= C \oplus C^1)$ and $\Delta C^2 (= C \oplus C^2)$, respectively. On the other hand, $Y_{12}$ satisfies the following equation.

$$Y_{12} = SL^{-1}(C \oplus ek_{13}).$$

Thus, guessing a 128-bit $ek_{13}$, we can check the guessed $ek_{13}$ by using these 18 equations. The probability that a wrong-guessed $ek_{13}$ passes this test is $2^{-144}$. So only the right $ek_{13}$ passes it. However, the computational complexity of this method is $2^{128}$.

In order to reduce the computational complexity, we consider divide and conquer strategy. First, we modify $M_0$ and $M_1$ by using elementary row operations. Among all possible patterns, we found the following two matrices $H_0$ and $H_1$.

$$H_0 = \begin{pmatrix}
0&1&1&0&0&0&0&0&0&0&0&0&0&1&0&0&1\\
0&0&1&0&0&0&1&0&1&0&0&0&1&0&0&0&\\
0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0&\\
0&0&0&0&1&0&0&0&1&0&0&0&1&1&1&0&\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&\\
0&0&0&0&0&0&1&0&0&1&0&0&0&1&1&1&\\
0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&
\end{pmatrix}
H_1 = \begin{pmatrix}
1&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&\\
0&0&1&1&0&0&0&0&1&1&0&0&0&0&0&0&\\
0&0&0&1&0&0&0&1&0&1&0&0&0&1&0&0&\\
0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&\\
0&0&0&0&0&1&1&0&0&1&1&0&0&1&0&0&\\
0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&
\end{pmatrix}$$

Under this fault assumption where two random byte faults are injected to $Y_{10,0}$ and $Y_{10,1}$ respectively, these matrices result in the lower computational complexity. As a result, we obtain the following 18 differential equations to appropriate for divide and conquer strategy.

$$\Delta Y^1_{12,1} \oplus \Delta Y^1_{12,2} \oplus \Delta Y^1_{12,12} \oplus \Delta Y^1_{12,15} = 0, \tag{4}$$
$$\Delta Y^1_{12,2} \oplus \Delta Y^1_{12,6} \oplus \Delta Y^1_{12,8} \oplus \Delta Y^1_{12,12} = 0, \tag{5}$$
$$\Delta Y^1_{12,3} \oplus \Delta Y^1_{12,6} \oplus \Delta Y^1_{12,9} \oplus \Delta Y^1_{12,12} = 0, \tag{6}$$
$$\Delta Y^1_{12,4} \oplus \Delta Y^1_{12,8} \oplus \Delta Y^1_{12,12} \oplus \Delta Y^1_{12,13} \oplus \Delta Y^1_{12,14} = 0, \tag{7}$$
$$\Delta Y^1_{12,5} \oplus \Delta Y^1_{12,14} = 0, \tag{8}$$
$$\Delta Y^1_{12,6} \oplus \Delta Y^1_{12,9} \oplus \Delta Y^1_{12,13} \oplus \Delta Y^1_{12,14} \oplus \Delta Y^1_{12,15} = 0, \tag{9}$$
$$\Delta Y^1_{12,7} \oplus \Delta Y^1_{12,13} = 0, \tag{10}$$
$$\Delta Y^1_{12,10} \oplus \Delta Y^1_{12,13} = 0, \tag{11}$$
$$\Delta Y^1_{12,11} \oplus \Delta Y^1_{12,14} = 0, \tag{12}$$
$$\tag{13}$$

$$\Delta Y^2_{12,0} \oplus \Delta Y^2_{12,3} \oplus \Delta Y^2_{12,13} \oplus \Delta Y^2_{12,14} = 0, \quad (14)$$
$$\Delta Y^2_{12,2} \oplus \Delta Y^2_{12,3} \oplus \Delta Y^2_{12,8} \oplus \Delta Y^2_{12,9} = 0, \quad (15)$$
$$\Delta Y^2_{12,3} \oplus \Delta Y^2_{12,7} \oplus \Delta Y^2_{12,9} \oplus \Delta Y^2_{12,13} = 0, \quad (16)$$
$$\Delta Y^2_{12,4} \oplus \Delta Y^2_{12,10} = 0, \quad (17)$$
$$\Delta Y^2_{12,5} \oplus \Delta Y^2_{12,6} \oplus \Delta Y^2_{12,9} \oplus \Delta Y^2_{12,10} \oplus \Delta Y^2_{12,13} = 0, \quad (18)$$
$$\Delta Y^2_{12,6} \oplus \Delta Y^2_{12,11} = 0, \quad (19)$$
$$\Delta Y^2_{12,7} \oplus \Delta Y^2_{12,8} \oplus \Delta Y^2_{12,10} \oplus \Delta Y^2_{12,11} \oplus \Delta Y^2_{12,14} = 0, \quad (20)$$
$$\Delta Y^2_{12,10} \oplus \Delta Y^2_{12,15} = 0, \quad (21)$$
$$\Delta Y^2_{12,11} \oplus \Delta Y^2_{12,12} = 0. \quad (22)$$

By using the above differential equations, we recover $ek_{13}$ as follows. Here, '$ek \Rightarrow_T E$' means that we check equations $E$ by using candidates contained in the table $T$ and the guessed $ek$. Then, $ek$ passing the test is included to $T$.

(1) Checking $(ek_{13,5}, ek_{13,6}, ek_{13,11}, ek_{13,12}, ek_{13,14})$.
   (a) Initialize table $T_1$ to empty table.
   (b) $(ek_{13,5}, ek_{13,14}) \Rightarrow_{T_1}$ Equation (8).
   (c) $ek_{13,11} \Rightarrow_{T_1}$ Equation (12).
   (d) $ek_{13,12} \Rightarrow_{T_1}$ Equation (21).
   (e) $ek_{13,6} \Rightarrow_{T_1}$ Equation (18).

(2) Checking $(ek_{13,4}, ek_{13,7}, ek_{13,10}, ek_{13,13}, ek_{13,15})$.
   (a) Initialize table $T_2$ to empty table.
   (b) $(ek_{13,7}, ek_{13,13}) \Rightarrow_{T_2}$ Equation (10).
   (c) $ek_{13,10} \Rightarrow_{T_2}$ Equation (11).
   (d) $ek_{13,15} \Rightarrow_{T_2}$ Equation (20).
   (e) $ek_{13,4} \Rightarrow_{T_2}$ Equation (16).

(3) Construct a table $T$ by combining $T_1$ and $T_2$.

(4) Checking $(ek_{13,0}, ek_{13,1}, ek_{13,2}, ek_{13,3}, ek_{13,8}, ek_{13,9})$
   (a) $ek_{13,8} \Rightarrow_T$ Equation (7) and (19).
   (b) $ek_{13,9} \Rightarrow_T$ Equation (9) and (17).
   (c) $ek_{13,1} \Rightarrow_T$ Equation (5).
   (d) $ek_{13,2} \Rightarrow_T$ Equation (4).
   (e) $ek_{13,3} \Rightarrow_T$ Equation (6), (14) and (15).
   (f) $ek_{13,0} \Rightarrow_T$ Equation (13).

Table 2 presents the length of guessed key, the computational complexity, the filtering probability and the size of table for each step. From this table, the computational complexity of the above procedure is $O(2^{24})$.

Theoretically, we expect that only the right $ek_{13}$ passes the above procedure. However, note that $ek_{13,0}$ and $ek_{13,1}$ are checked only one time in the above procedure.

Table 2
Computing $ek_{13}$

| Step | Length of guessed key | Computational complexity | Filtering probability | Size of table |
|---|---|---|---|---|
| 1-(a) | · | · | · | · |
| 1-(b) | 16 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 1-(c) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 1-(d) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 1-(e) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 2-(a) | · | · | · | · |
| 2-(b) | 16 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 2-(c) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 2-(d) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 2-(e) | 8 | $2^{16}$ | $2^{-8}$ | $2^8$ |
| 3 | · | · | · | $2^{16}$ |
| 4-(a) | 8 | $2^{24}$ | $2^{-16}$ | $2^8$ |
| 4-(b) | 8 | $2^{16}$ | $2^{-16}$ | 1 |
| 4-(c) | 8 | $2^8$ | $2^{-8}$ | 1 |
| 4-(d) | 8 | $2^8$ | $2^{-8}$ | 1 |
| 4-(e) | 8 | $2^8$ | $2^{-24}$ | 1 |
| 4-(f) | 8 | $2^8$ | $2^{-8}$ | 1 |

From the differential distribution table of S-box, given an input/output difference pair, the number of possible input values is 0, 2 or 4. Thus, the number of candidates of $ek_{13}$ is 4, 8 or 16. On the other hand, we construct the following additional 12 differential equations on $(\Delta Y^1_{11}, \Delta Y^2_{11})$ as mentioned in Section 3.2.

$$\begin{aligned}
\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,4} = 0, &\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,6} = 0,\\
\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,8} = 0, &\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,9} = 0,\\
\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,13} = 0, &\Delta Y^1_{11,3} \oplus \Delta Y^1_{11,14} = 0,\\
\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,5} = 0, &\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,7} = 0,\\
\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,8} = 0, &\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,9} = 0,\\
\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,12} = 0, &\Delta Y^2_{11,2} \oplus \Delta Y^2_{11,15} = 0.
\end{aligned}$$

With $(C^1, C^2)$ and each candidate of $ek_{13}$, we can compute candidates of $ek_{12}$ by checking the above differential equations. The computational complexity of this procedure is $O(2^{16})$. Note that $ek_{12,0}, ek_{12,2}, ek_{12,10}$ and $ek_{12,11}$ do not affect the above differential equations. Thus, the expected number of candidates of $ek_{12}$ is $2^{32}$.

Hence, we obtain at most $2^{36} (= 2^4 \cdot 2^{32})$ candidates of $(ek_{12}, ek_{13})$ with the $O(2^{24})$ computational complexity and two fault injections.

### *3.2 Assumption $A^2$*

Under assumption $A^2$, we assume that an attacker cannot know the location of injected faults. Thus, to construct differential equations, he should guess it. Since there exist 16 possible positions for each fault injection, we need to consider $256(= 16 \cdot 16)$ possible cases for two fault injections.

These 256 cases can be partitioned into two types. The first type includes 240 cases where the positions of two injected faults are different. For each case, we compute candidates of $ek_{13}$ by using the similar method to that under assumption $A^1$. Here, if a guessed position of injected faults is wrong, the expected number of the survived $ek_{13}$ is $2^{-16}(= 2^{128} \cdot 2^{-144})$. In the case of the right-guessed position, 4, 8 or 16 candidates of $ek_{13}$ are remained.

The second type includes 16 cases where the positions of two injected faults are same. By using this type, we can only recover 15 bytes of $ek_{13}$. Recall that, in Section 3.2, all constructed equations do not include $\Delta Y_{r,0}$ assuming that a random byte fault is induced to $X_{r-2,0}$. In general, when a random byte fault is induced to $X_{r-2,i}$, all constructed equations do not include $\Delta Y_{r,i}(i = 0, \cdots, 15)$. Thus, for each wrong-guessed position, the expected number of the remained $ek_{13}$ is $2^{-16}(= 2^8 \cdot 2^{120} \cdot 2^{-144})$. In the case of the right-guessed position, $2^8$ candidates of $ek_{13}$ are survived.

In summary, if some candidates of $ek_{13}$ pass this test, we may expect that a guessed position is right. And, in this case, the number of candidates of $ek_{13}$ is at most $2^8$. Note that we can obtain the exact position of injected faults from the above procedure. Thus, as mentioned in Section 3.2, we can construct 12 differential equations on $(\Delta Y^1_{11}, \Delta Y^2_{11})$. By using them, we obtain candidates of $ek_{12}$ for each candidate of $ek_{13}$. Similarly to Section 4.1, the expected number of candidates of $ek_{12}$ is $2^{32}$. Thus, in the worst case, $2^{40}(= 2^8 \cdot 2^{32})$ candidates of $(ek_{12}, ek_{13})$ are survived.

Hence, we obtain at most $2^{40}(= 2^8 \cdot 2^{32})$ candidates of $(ek_{12}, ek_{13})$ with the $O(2^{32}(= 2^8 \cdot 2^{24}))$ computational complexity and two random byte fault injections.

## 4 DFAs on ARIA

Now, we are ready to propose DFAs on ARIA-128/192/ 256 under four fault assumptions, that is $A^1_E, A^1_{DE}, A^2_E, A^2_{DE}$. Table 3 summarizes our attack results on ARIA.

### *4.1 DFA on ARIA under $A^1_E$*

Under assumption $A^1_E$, the attack procedure on ARIA-128/192/256 is as follows.

Table 3
DFA results on ARIA

| Fault assumption | Target algorithm | Computational complexity | ♯ of injected faults |
|---|---|---|---|
| $A_E^1$ | ARIA-128 | $O(2^{24})$ | 6 |
| | ARIA-192 | $O(2^{24})$ | 6 |
| | ARIA-256 | $O(2^{24})$ | 6 |
| $A_E^2$ | ARIA-128 | $O(2^{32})$ | 6 |
| | ARIA-192 | $O(2^{32})$ | 6 |
| | ARIA-256 | $O(2^{32})$ | 6 |
| $A_{DE}^1$ | ARIA-128 | $O(2^{24})$ | 4 |
| | ARIA-256 | $O(2^{24})$ | 4 |
| $A_{DE}^2$ | ARIA-128 | $O(2^{32})$ | 4 |
| | ARIA-256 | $O(2^{32})$ | 4 |

(1) [**Collection of right ciphertext**] Choose a plaintext $P$ and obtain the corresponding right ciphertext $C$.

(2) [**Collection of faulty ciphertexts**] After inducing six random byte faults to $(Y_{r-2,0}, Y_{r-2,1}, Y_{r-3,4}, Y_{r-4,0}, Y_{r-4,1}, Y_{r-5,4})$, respectively, and obtain the corresponding faulty ciphertexts $(C^1, C^2, C^3, C^4, C^5, C^6)$.

(3) [**Computation of candidates of** $(ek_r, ek_{r+1})$]
   (a) With $(C^1, C^2)$, compute candidates of $(ek_r, ek_{r+1})$ by using the method presented in Section 4.1.
   (b) With $C^3$, construct additional 9 differential equations on $Y_{r-2}$ and reduce the number of candidates of $(ek_r, ek_{r+1})$ similarly to Step 3-(a).

(4) [**Computation of candidates of** $(ek_{r-2}, ek_{r-1})$] By repeating Step 3 with $(C^4, C^5, C^6)$, obtain candidates of $(ek_{r-2}, ek_{r-1})$.

(5) [**Recovery of the secret key** $K_n$] By using the method introduced in Section 3.3, recover the secret key $K_n$ from candidates of $(ek_{r-2}, ek_{r-1}, ek_r, ek_{r+1})$.

As mentioned in Section 4.1, the expected number of candidates of $(ek_r, ek_{r+1})$ passing Step 3-(a) is at most $2^{36}$. Moreover, since the filtering probability of Step 3-(b) is $2^{-72}$, we expect that only the right $(ek_r, ek_{r+1})$ passes Step 3. Similarly to Step 3, we expect that only the right $(ek_{r-2}, ek_{r-1})$ passes Step 4. It is known that, if we know the four consecutive round keys, it is possible to recover $K_n$. Since we know the right $(ek_r, ek_{r+1}, ek_{r-2}, ek_{r-1})$, we can recover $K_n$.

The computation complexity of the above attack algorithm depends on heavily Step 3 and Step 4. In Step 3-(a), the computational complexity is $O(2^{24})$ as mentioned in Section 4.1. Compared to Step 3-(a), the computational complexity of Step 3-(b) is negligible. Thus, the computational complexity of Step 3 is $O(2^{24})$. Since the computational complexity of Step 4 is equal to that of Step 3, The total computational complexity of our attack is $O(2^{24})$.

We simulated this attack on a PC 1000 times. As a result, we could always recover $K_n$ within a few seconds. Hence, our attack can recover the secret key of ARIA-128/192/256 with six random byte fault injections and the computational complexity of $O(2^{24})$.

### *4.2 DFA on ARIA under $A_E^2$*

Under assumption $A_E^2$, the overall attack procedure is similar to that under $A_E^1$. As mentioned in Section 4.2, we obtain at most $2^{40}$ candidates of $(ek_r, ek_{r+1})$ with the $O(2^{32})$ computational complexity and two random byte fault injections. However, similarly to Step 3-(b) under $A_E^1$, the right $(ek_r, ek_{r+1})$ is only survived by using an additional fault injection.

We simulated this attack on a PC 100 times. As a result, we could always recover $K_n$ within ten minutes. Hence, our attack can recover the secret key of ARIA-128/192/256 with six random byte fault injections and the computational complexity of $O(2^{32})$.

### *4.3 DFAs on ARIA-128/256 under $A_{DE}^1$ and $A_{DE}^2$*

Note that, under $A_{DE}^1$ and $A_{DE}^2$, we need to obtain the first and last round keys of ARIA. In the encryption and decryption processes, we obtain the last round key $ek_{r+1}$ and $dk_{r+1}$, respectively. Here, it is known that $dk_{r+1}$ is equal to $ek_1$. Thus, if we obtain $(ek_{r+1}, dk_{r+1})$, we can obtain $(ek_1, ek_{r+1})$.

The attack procedure under $A_{DE}^1$ is as follows. The case of $A_{DE}^2$ can be presented in a similar fashion.

(1) **[Collection of right ciphertext]** Choose a plaintext $P$ and obtain the corresponding right ciphertext $C$.

(2) **[Collection of faulty ciphertexts in the encryption process]** After inducing two random byte faults to $(Y_{r-2,0}, Y_{r-2,1})$ in the encryption process, and obtain the corresponding faulty ciphertexts $(C^1, C^2)$.

(3) **[Computation of candidates of $ek_{r+1}$]** With $(C^1, C^2)$, compute candidates of $ek_{r+1}$ by using the method presented in Section 4.1.

(4) **[Collection of faulty ciphertexts in the decryption process]** After inducing two random byte faults to $(Y_{r-2,0}, Y_{r-2,1})$ in the decryption process, and obtain the corresponding faulty plaintexts $(P^1, P^2)$.

(5) **[Computation of candidates of $ek_1$]** By repeating Step 3 with $(P^1, P^2)$, obtain candidates of $dk_{r+1}$. And convert $dk_{r+1}$ to $ek_1$.

(6) [**Recovery of the secret key** $K_n$] By using the method introduced in Section 3.3, recover the secret key $K_n$ from candidates of $(ek_1, ek_{r+1})$.

The computational complexity of the attack under $A^1_{DE}$ and $A^2_{DE}$ is $O(2^{24})$ and $O(2^{32})$, respectively. The simulation results are also similar to them under $A^1_E$ and $A^2_E$, respectively.

## 5 Conclusion

In this paper, we proposed DFAs on ARIA-128/192/256. The proposed attacks can recover the secret key with the $O(2^{24})$ computational complexity and 6 random byte fault injections. Furthermore, in cases of ARIA-128/256, it is possible to recover the secret key with the $O(2^{24})$ computational complexity and 4 random byte fault injections under a fault assumption where an attacker can induce some faults during both encryption and decryption process, respectively. As simulation results, our attacks recovered the secret key within a few minutes. Particularly, our results on ARIA-192/256 are the first known DFA results on them.

In the future, we will study the method to reduce the fault injections and computational complexity.

## References


[1] E. Biham and A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, Crypto 1997, LNCS 1294, pp. 513–525, Springer-Verlag, 1997.

[2] J. Blomer and J. Seifert, *Fault Based Cryptan alysis of the Advanced Encryption Standard (AES)*, FC 2003, LNCS 2742, pp. 162–181, Springer-Verlag, 2003.

[3] A. Byukov, C. Canniere, J. Lano, S. Ors and B. Preneel, *Security and Performance Analysis of Aria*, Version 1.2, 2004.

[4] H. Chen, W. Wu and D. Feng, *Differential fault ana lysis on CLEFIA*, ICICS 2007, LNCS 4861, pp. 284–295, Springer-Verlag, 2007.

[5] C. Du and J. Chen, *Impossible Differential Crypt ana- lysis of ARIA Reduced to 7 Rounds*, CANS 2010. LNCS 6467, pp. 20-30, Springer-Verlag, 2010.

[6] E. Fleischmann, C. Forler, M. Gorski and S. Lucks, *New Boomerang Attacks on ARIA*, INDOCRYPT 2010, LNCS 6498, pp. 163–175, Springer-Verlag, 2010.

[7] T. Fukunaga and J. Takahashi, *Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers*, FDTC 2009, pp. 84–92, IEEE Computer Society, 2009.

[8] C. Giraud, *DFA on AES*, AES 2004, LNCS 3373, pp 27–41, Springer-Verlag, 2005.

[9] L. Hemme, *A differential fault analysis against early rounds of (Triple-) DES*, CHES 2004, LNCS 3156, pp. 254–267, Springer-Verlag, 2006.

[10] K. Jeong, Y. Lee, J. Sung and S. Hong, *Differential fault analysis on block cipher SEED*, Mathematical and Computer Modelling, Vol. 55, Issue 1-2, pp. 26–34, Elsevier, 2012.

[11] C. Kim, *Differential fault analysis against AES-192 and AES-256 with minimal faults*, FDTC 2010, pp 3–9, IEEE Computer Society, 2010.

[12] C. Kim, *Differential fault analysis of AES: Toward reducing number of faults*, Information Sciences, Vol. 199, pp. 43–57, ELSEVIER, 2012.

[13] C. H. Kim, *Differentialfault analysis of ARIA in multi-byte fault models*, Jounal of Systems and Software, Vol 85, No. 9, pp.2096–2103, 2012

[14] D. Kwon, J. Kim, S. Park, S. Sung, Y. Sohn, J. Song, Y. Yeom, E. Yoon, S. Lee, J. Lee, S. Chee, D. Han and J. Hong, *New block cipher ARIA*, ICISC 2003, LNCS 2971, pp. 432–445, Springer-Verlag, 2003.

[15] W. Li, D. Gu and J. Li, *Differential fault analysis on the ARIA algorithm*, Information Sciences, Vol. 178, No. 19, pp. 3727–3737, Elsevier, 2008.

[16] W. Li, D. Gu, and Y. Wang, *Differential fault analysis on the contracting UFN structure, with application to SMS4 and Macguffin*, Journal of Systems and Software, Vol. 82, No. 2, pp. 346–354, Elsevier, 2009.

[17] P. Li , B. Sun , C. Li, *Integral cryptanalysis of ARIA*, Proceedings of the 5th international conference on Information security and cryptology, 2009.

[18] D. Mukhopadhyay, *An improved fault based attack of the advanced encryption standard*, In AFRICACRYPT 2009, LNCS 5580, pp. 421–434, Springer-Verlag, 2009.

[19] J. Park and j. Ha, *Improved Dierential Fault Analysis on Block Cipher ARIA*, The 13th International Workshop on Information Security Applications, 2012.

[20] M. Tunstall and D. Mukhopadhyay, *Differential fault analysis of the advanced encryption standard using a single fault*, IACR eprint archive, 2009/575, 2009.

[21] W. Wu, W. Zhang, D. Feng, *Impossible differential cryptanalysis of reduced-round ARIA and Camellia*, Journal of Computer Science and Technology, Vol. 22, No. 3, pp. 449–456, 2007.