Multimed Tools Appl (2013) 63:691–708
DOI 10.1007/s11042-011-0895-x

Fault analysis study of the block cipher FOX64

Ruilin Li · Jianxiong You · Bing Sun · Chao Li

Published online: 15 October 2011

Abstract FOX is a family of symmetric block ciphers from MediaCrypt AG that helps to secure digital media, communications, and storage. The high-level structure of FOX is the so-called (extended) Lai–Massey scheme. This paper presents a detailed fault analysis of the block cipher FOX64, the 64-bit version of FOX, based on a differential property of two-round Lai–Massey scheme in a fault model. Previous fault attack on FOX64 shows that each round-key (resp. whole round-keys) could be recovered through 11.45 (resp. 183.20) faults on average. Our proposed fault attack, however, can deduce any round-key (except the first one) through 4.25 faults on average (4 in the best case), and retrieve the whole round-keys through 43.31 faults on average (38 in the best case). This implies that the number of needed faults in the fault attack on FOX64 can be significantly reduced. Furthermore, the technique introduced in this paper can be extended to other series of the block cipher family FOX.

Keywords Multimedia security · Side-channel attacks · Fault attacks · Block ciphers · Lai–Massey scheme · FOX

The work in this paper is supported by the National Natural Science Foundation of China (No: 61103192, 61070215), the Program for Changjiang Scholars and Innovative Research Team in University of Ministry of Education of China (No: IRT1012), the Fund for Creative Research Groups of the Natural Science Foundation of Hunan Province, China (No: 11FH002), and the open research fund of State Key Laboratory of Information Security (No: 01-02-5).

R. Li (✉) · J. You · B. Sun · C. Li
Department of Mathematics and System Science, Science College,
National University of Defense Technology, Changsha, 410073, China
e-mail: securitylrl@gmail.com

B. Sun
State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing, 100190, China

C. Li
College of Computer, National University of Defense Technology, Changsha, 410073, China

1 Introduction

1.1 Backgrounds

Last fifteen years saw the rapid development of physical attacks on symmetric key primitives based on the implementation of cryptographic algorithms in hardware or software environments. Among these attacks, side-channel attacks and fault attacks attract more attention. Side-channel attack is a kind of passive attack that could be used to retrieve the secrete key by exploiting some leakage of the internal states during the encryption process. These leakage includes timing information, power consumptions, electro-magnetic radiations, etc., and can be collected by some special equipments. After obtaining these leakage, adversary can efficiently deduce the key through some statistical tools or algebraic methods.

Fault attack is another kind of powerful physical attacks, where the adversary could actively disturb part of the internal states, or cause calculation errors during the execution of a cryptographic algorithm, that is to say some faults or errors could be injected in the device running the encryption procedure. The idea of using faults to break the cryptosystems was introduced by Boneh et al. [11] from Bellcore in 1996. They showed that in the RSA-CRT setting, a single computational mistake can completely break the scheme by factoring the public key. Later, such a method was extended by Biham and Shamir [8] to DES-like secret key cryptosystems and referred as Differential Fault Analysis. Since then, fault attacks have been used to attack many other symmetric key ciphers.

1.2 Related works

Fault attacks on block ciphers Besides DES-like (Feistel structure) block ciphers, many research concentrate applying fault attacks against other kinds of block ciphers that adopt (generalized) unbalanced Feistel structure, SPN structure or Lai–Massey structure. For instance, among fault based attacks on AES [1, 10, 14, 16, 17, 24, 30–32, 36], the works in [31] show the feasibility of extracting the 128-bit key through one single byte fault under Piret's attack model [32]. And by studying the properties of mixing operations ($\oplus, \boxplus, \odot$) on different groups, a fault attack on the nice block cipher IDEA was carefully analyzed in [15].

Faults attacks on stream ciphers Hoch and Shamir firstly applied fault based attack to several stream ciphers [19], including LFSR-based traditional ciphers (schemes based on memoryless filters and combiners, such as LILI-128 and Sober) and RC4 etc. in 2004. Then in 2005, Biham et al. investigated impossible fault and differential fault analysis of RC4 [9]. This problem was later solved in [4] by Armknecht and Meier who developed fault attacks against general combiners with memory based on LFSR's (see e.g. E0 and SNOW 2.0). More recently, many winners of the stream ciphers from eSTREAM project, such as Trivium, Rabbit, HC-128, were also shown to be suffered from fault attacks [7, 20, 21, 25, 26].

Fault models and implementation aspects When applying fault attacks, it is usually assumed that the adversary could inject faults during a calculation of the encryption (decryption) algorithm, thus the fault model (the location and timing of fault injection, and the values of faults) is the main issue that one must take into account. Many ingredients influence the model, such as the inner structure, the data register, the powerful equipments possessed by the adversary, etc. In fact, there are many ways of performing a fault attack [2, 3, 5, 18, 35], such as glitch attack, light attack, magnetic attack, etc. These methods could induce faults at bit, byte or even multibytes levels.

Countermeasures Another important issue of fault attacks is the countermeasures. Compared with the existing fault attacks on various ciphers, the countermeasures seem to belong to another independent field, and there are many general countermeasures (hardware or software oriented) [6, 23, 29, 33, 34] for protecting ciphers from fault attacks. The main principle is to provide inherent resistance to prevent specific transient fault, or to offer redundant calculation for error detection. However, as demonstrated in [12, 27], providing countermeasures against fault attacks of block ciphers maybe a challenging and hard task.

1.3 Main contribution and outline of this paper

FOX and previous fault attack FOX [22], also known as IDEA-NXT, is a family of block ciphers designed by Junod and Vaudenay. It offers a flexible, scalable platform that ensures optimal usage across a diverse range of devices and applications, and helps to secure digital media, communications, and storage. The block size of FOX is either 64-bit or 128-bit, both of which have a variable key length ranging from 8 to 256 bits. The high-level structure of FOX is the so-called (extended) Lai–Massey scheme, whose provable security had been carefully studied in [28, 37]. The round function is of SPS-type with three layers of round key addition, which could meanwhile provide its security against differential and linear cryptanalysis.

Another feature of FOX is its key schedule, as has been declaimed in [22]. Unlike most other block ciphers, given any single or even several round-keys of FOX, it is hard to invert the round-key generation process to obtain the master key. Due to this, any adversary that basing fault attacks to deduce the secrete key has to retrieve the whole round-keys, which implies that the number of faults injected must be large. In fact, this has been verified by Breveglieri et al. [13], where they introduced a first fault attack on FOX64, the 64-bit version of FOX, and demonstrated that one round-key could be recovered by 11.45 faults on average in the random byte fault model, and the whole round-keys could be recovered by 183.20 faults on average.

Main contribution This paper presents a detailed fault analysis of FOX64, and proposes an improved fault attack. This is mainly based on an observation of *a differential property of two round Lai–Massey scheme in a fault model*. Using the proposed method, faulty ciphertexts that are obtained when faults are injected in the i-th round can be used twice (previous fault attack use those ciphertexts only once for deducing the i-th round-key): one for deducing the $(i+1)$-th round-key, and the other for deducing the i-th round-key, thus the number of needed faults are significantly reduced. The simulation result shows that the 64-bit round-key could be revealed by 4.25 faults on average, and the whole round-keys can be deduced through only 43.31 faults on average. Furthermore, such a method could be generalized to other series of the block cipher family FOX.

Outline The outline of this paper is as follows: we begin with a brief description of FOX in Section 2, and then present some properties of the components of FOX in Section 3. The previous fault attack is described in Section 4, and our improved fault attack is presented in Section 5. Section 6 demonstrates experimental results, and finally, Section 7 is the conclusion.

2 Description of the block cipher FOX

This section briefly describes FOX64, for other series, one can refer [22].

2.1 Encryption of FOX64

FOX64 has a 64-bit block size and a 128-bit key length. It iterates 15 times the round transformation **lmor64**, as illustrated in Fig. 1, followed by a final round transformation called **lmid64**.

The round transformation **lmor64**, which employs a Lai–Massey scheme, transforms a 64-bit input $x_{(64)}$ and a 64-bit round key $rk_{(64)}$ into a 64-bit output $y_{(64)}$, which is defined as

$$\begin{aligned} y_{(64)} &= \texttt{lmor64}\left(x_{l(32)} \| x_{r(32)}, rk_{(64)}\right) \\ &= \textbf{or}\left(x_{l(32)} \oplus \texttt{f32}\left(x_{l(32)} \oplus x_{r(32)}, rk_{(64)}\right)\right) \| \left(x_{r(32)} \oplus \texttt{f32}\left(x_{l(32)} \oplus x_{r(32)}, rk_{(64)}\right)\right), \end{aligned}$$

where f32 is the round function, and **or** is an orthomorphism.

The orthomorphism **or** is a function that takes a 32-bit input $x_{(32)}$ and returns a 32-bit output $y_{(32)}$, as illustrated in Fig. 2, which is defined by a simple Feistel transformation as

$$y_{l(16)} || y_{r(16)} = \textbf{or}\left(x_{l(16)} || x_{r(16)}\right) = x_{r(16)} || \left(x_{l(16)} \oplus x_{r(16)}\right).$$

The **lmid64** function is a slightly modified version of **lmor64**, where the transformation **or** is replaced by a identify transformation.

The encryption $c_{(64)}$ by FOX64 of a 64-bit plaintext $p_{(64)}$ is defined as

$$c_{(64)} = \texttt{lmid64}\left(\texttt{lmor64}\left(\cdots \texttt{lmor64}\left(p_{(64)}, rk_0\right), \cdots, rk_{14}\right), rk_{15}\right),$$

where rk_i, $i = 0, 1, \cdots, 15$, are round-keys generated through the key schedule from the master key.

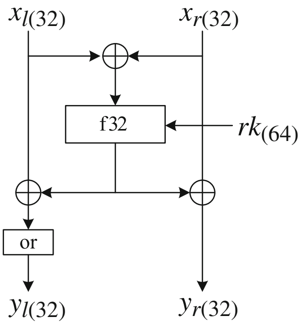


Fig. 1 Round transformation **lmor64**

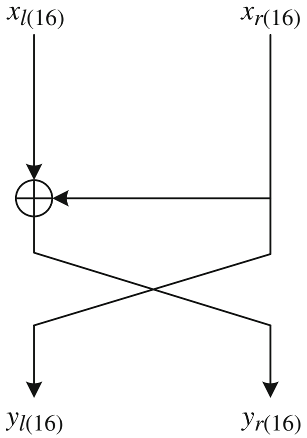


Fig. 2 Orthomorphism **or**

2.2 Round function f32

The round function `f32` consists of three main parts: a substitution part, denoted sigma4, a diffusion part, denoted mu4, and a round key addition part. Formally, the round function `f32` takes a 32-bit input $x_{(32)}$, a 64-bit round key $rk_{(64)} = rk_{0(32)}||rk_{1(32)}$ and returns

$$\begin{aligned} y_{(32)} &= \texttt{f32}\left(x_{(32)}, rk_{(64)}\right) \\ &= \texttt{sigma4}\left(\texttt{mu4}\left(\texttt{sigma4}\left(x_{(32)} \oplus rk_{0(32)}\right)\right) \oplus rk_{1(32)}\right) \oplus rk_{0(32)}. \end{aligned}$$

The fuction `sigma4` takes a 32-bit input $x_{(32)} = x_{0(8)}\|x_{1(8)}\|x_{2(8)}\|x_{3(8)}$ and returns a 32-bit output $y_{(32)}$, it consists of 4 parallel computations of a non-linear mapping `sbox`, i.e.

$$\begin{aligned} y_{(32)} &= \texttt{sigma4}\left(x_{0(8)}\|x_{1(8)}\|x_{2(8)}\|x_{3(8)}\right) \\ &= \texttt{sbox}\left(x_{0(8)}\right)\|\texttt{sbox}\left(x_{1(8)}\right)\|\texttt{sbox}\left(x_{2(8)}\right)\|\texttt{sbox}\left(x_{3(8)}\right). \end{aligned}$$

The function `mu4` takes a 32-bit input $x_{0(8)}\|x_{1(8)}\|x_{2(8)}\|x_{3(8)}$ and returns a 32-bit output $y_{0(8)}\|y_{1(8)}\|y_{2(8)}\|y_{3(8)}$. It is defined by

$$\begin{pmatrix} y_{0(8)} \\ y_{1(8)} \\ y_{2(8)} \\ y_{3(8)} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \theta \\ 1 & z & \theta & 1 \\ z & \theta & 1 & 1 \\ \theta & 1 & z & 1 \end{pmatrix} \times \begin{pmatrix} x_{0(8)} \\ x_{1(8)} \\ x_{2(8)} \\ x_{3(8)} \end{pmatrix},$$

where $\theta \in \mathrm{GF}(2^8)$ is the root of the irreducible polynomial $m(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1 \in \mathrm{GF}(2)[x]$ and $z = \theta^{-1} + 1$.

2.3 Key schedule

The key schedule procedure of FOX64 generates 16 round-keys rk_i, $i = 0, 1, \ldots, 15$, from a master key K. Each round-key is 64-bit, denoted as a concatenation of two 32-bit strings, i.e. $rk_i = rk_{i,0}\|rk_{i,1}$.

The key schedules of FOX series are very complex compared with other existing block ciphers, each round-key is related to the secret key and it is very difficult to acquire information about secret key or other round-keys from some certain round-keys. Due to this, in this paper, we assume that all round-keys are independent with each other. One can refer [22] for the detail of the key schedule.

3 Some properties of the components of FOX64

In this section, we study some properties of the components of FOX64 that are related to the fault attacks.

3.1 Differential property of the S-box in the substitution layer

Given an 8×8 Sbox $S(\cdot)$, $\alpha, \beta \in \{0,1\}^8$, define $N_S(\alpha,\beta) = \#\{x \in \{0,1\}^8 : S(x) \oplus S(x \oplus \alpha) = \beta\}$, then the differential property of the Sbox can be characterized by all the possible triplets $(\alpha, \beta, N_S(\alpha,\beta))$. Table 1 summarizes the differential property of the Sbox employed in the substitution layer of FOX64.

Remark 1 Assume $S(\cdot)$ is the Sbox of FOX64, if $N_S(\alpha,\beta) \neq 0$, then the expected value of $N_S(\alpha,\beta)$ is

$$\frac{65536}{65536 - 42871} \approx 2.89$$

This indicates that on average, one pair (α, β) could provide about 2.89 inputs x such that $S(x) \oplus S(x \oplus \alpha) = \beta$.

3.2 Inverse property of the diffusion layer

The differential branch number of `mu4` is 5, which implies that any input with one non-zero byte will lead to some output with four non-zero bytes. Moreover, the inversion of `mu4`, denoted as $\texttt{mu4}^{-1}$, can be expressed as follow:

$$\begin{pmatrix} a & c & d & e \\ a & d & e & c \\ a & e & c & d \\ b & a & a & a \end{pmatrix},$$

where $a = \theta^6 + \theta^5 + \theta^4 + \theta^3 + \theta^2 + \theta$, $b = \theta^7 + \theta^6 + \theta + 1$, $c = \theta^7 + \theta^6 + \theta^5 + 1$, $d = \theta^7 + \theta^5 + \theta^3 + \theta^2 + 1$ and $e = \theta^7 + \theta^5 + \theta^4$.

Table 1 Differential property of the Sbox in FOX64

$N_S(\alpha,\beta)$	Frequency	$N_S(\alpha,\beta)$	Frequency
0	42,871	10	19
2	15,377	12	6
4	5,758	16	70
6	680	256	1
8	754	–	–

4 Previous fault attack on FOX64

To briefly describe the previous fault attack [13] and our improved fault attack on FOX64 in the next section, we first introduce the following notations that will be used throughout this paper.

- Assume all the round-keys, generated from the secret master key K through the key schedule, are $rk_i = rk_{i,0} \| rk_{i,1}$, $i = 0, 1, 2, \ldots, 15$.
- Denote a plaintext by $p = p_l \| p_r$, and the corresponding ciphertext by $c = E_K(p) = c_l \| c_r$. This ciphertext is called the *right* ciphertext and any indeterminate states corresponding to it are called the *right* indeterminate states.
- Consider the last round of FOX64: Let A_{16} denote the input of the round function f32; I_1 and B_{16} denote the input and output of the first substitution layer, respectively, i.e. $I_1 = A_{16} \oplus rk_{15,0}$, $B_{16} = \texttt{sigma4}(I_1)$; C_{16} denote the output of the diffusion layer, i.e. $C_{16} = \texttt{mu4}(B_{16})$; I_2 and D_{16} denote the input and output of the second substitution layer, i.e. $I_2 = C_{16} \oplus rk_{15,1}$, $D_{16} = \texttt{sigma4}(I_2)$.
- Given a 32-bit right state X, X^* denotes the *faulty* counterpart, and $\Delta X = X \oplus X^*$ denotes their difference.

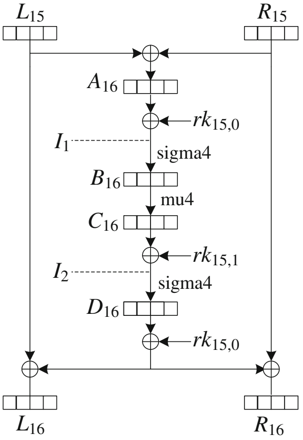
As discussed in Section 2.3, due to the complexity of the key schedule, any fault attack on FOX64 aims to deduce all the round-keys.

Take the last round of FOX64 as an example, see Fig. 3, both the previous fault attack and our improved fault attack (as described later) try to retrieve the 64-bit round-key by recovering the right intermediate state I_1 and I_2. Once I_1 and I_2 are known, one can do as follows:

- According to $I_1 = A_{16} \oplus rk_{15,0} = c_l \oplus c_r \oplus rk_{15,0}$, we thus have $rk_{15,0} = c_l \oplus c_r \oplus I_1$.
- According to $I_2 = C_{16} \oplus rk_{15,1} = \texttt{mu4}(B_{16}) \oplus rk_{15,1} = \texttt{mu4}(\texttt{sigma4}(I_1)) \oplus rk_{15,1}$, we thus have $rk_{15,1} = I_2 \oplus \texttt{mu4}(\texttt{sigma4}(I_1))$.

In order to recover the right state I_1 and I_2, previous fault attack adopts the random byte fault model and divides the attack procedure into the following two phases.

Fig. 3 Last round of FOX64



- In the first phase, the adversary injects faults into the calculation of round function, and the location is between the input of the round function and input of the diffusion layer. By using both the correct ciphertext and faulty ciphertexts, he applies differential cryptanalysis on the second substitution layer to recover I_2. The number of faults in this phase is about $2 \sim 8$ and 2.94 on average.
- In the second phase, the adversary also injects faults into the calculation of the round function and the location this time is only before the first substitution layer. He then applies differential cryptanalysis to the first substitution layer to recover I_1 based on the correct ciphertext and those faulty ciphertexts. The number of faults in this phase is about $8 \sim 28$ and 8.51 on average.

In total, about $8 \sim 31$ faults (11.45 on average) are needed to recover the right state I_1 and I_2.

5 Improved fault attack on FOX64

In this section, we present our improved fault attack on FOX64, which is, in fact, based on a differential property of two-round Lai–Massey Scheme in a fault model.

5.1 A differential property of two-round Lai–Massey scheme in a fault model

Consider a two-round Lai–Massey scheme in a fault model as shown in Fig. 4. Let L_i and R_i be the left and right halves of the round input or output, where $i = 0, 1, 2$. Let A_j and D_j be the input and output of the bijective round function `f32`, where $j = 0, 1$. Assume a fault is induced into A_0, and denote the difference of a 32-bit state X as ΔX, then we have the following proposition:

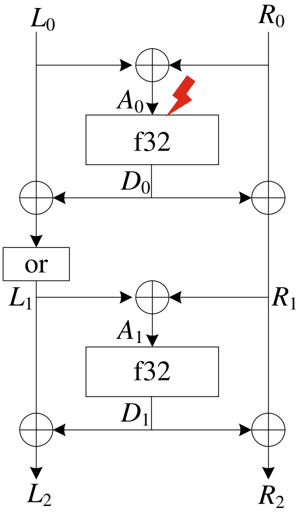


Fig. 4 Two round Lai–Massey scheme in a fault model

Proposition 1 *Given a two-round Lai–Massey scheme as described above, assume $\Delta L_0 = 0$, $\Delta R_0 = 0$, and a fault is induced into A_0, i.e. $\Delta A_0 \neq 0$. Let $\Delta L_2 = (\alpha_0, \alpha_1, \alpha_2, \alpha_3)$, $\Delta R_2 = (\beta_0, \beta_1, \beta_2, \beta_3)$ be known values, then both the input difference and output difference of* `f32` *in the second round, i.e. ΔA_1 and ΔD_1, could be calculated.*

Proof From $\Delta A_0 \neq 0$, we have $\Delta D_0 \neq 0$. Assume $\Delta D_0 = (x_0, x_1, x_2, x_3)$, then

$$\begin{aligned} \Delta L_1 &= \texttt{or}\,(\Delta L_0 \oplus \Delta D_0) = \texttt{or}\,(x_0, x_1, x_2, x_3) = (x_2, x_3, x_0 \oplus x_2, x_1 \oplus x_3)\,, \\ \Delta R_1 &= (x_0, x_1, x_2, x_3)\,. \end{aligned}$$

Notice that $\Delta A_1 = \Delta L_1 \oplus \Delta R_1 = \Delta L_2 \oplus \Delta R_2$, thus

$$\Delta A_1 = (\alpha_0 \oplus \beta_0, \alpha_1 \oplus \beta_1, \alpha_2 \oplus \beta_2, \alpha_3 \oplus \beta_3)$$

is known. Meanwhile, from

$$(x_2, x_3, x_0 \oplus x_2, x_1 \oplus x_3) \oplus (x_0, x_1, x_2, x_3) = (\alpha_0, \alpha_1, \alpha_2, \alpha_3) \oplus (\beta_0, \beta_1, \beta_2, \beta_3)\,,$$

we get

$$(x_0, x_1, x_2, x_3) = (\alpha_2 \oplus \beta_2, \alpha_3 \oplus \beta_3, \alpha_0 \oplus \alpha_2 \oplus \beta_0 \oplus \beta_2, \alpha_1 \oplus \alpha_3 \oplus \beta_1 \oplus \beta_3)\,.$$

Thus

$$\begin{aligned} \Delta D_1 &= \Delta R_2 \oplus \Delta R_1 \\ &= (\beta_0, \beta_1, \beta_2, \beta_3) \oplus (x_0, x_1, x_2, x_3) \\ &= (\alpha_2 \oplus \beta_0 \oplus \beta_2, \alpha_3 \oplus \beta_1 \oplus \beta_3, \alpha_0 \oplus \alpha_2 \oplus \beta_0, \alpha_1 \oplus \alpha_3 \oplus \beta_1)\,. \end{aligned}$$

which ends the proof. □

Remark 2 Proposition 1 also holds in the situation, where the second round transformation in the two-round Lai–Massey scheme contains the orthomorphism **or**. This is due to the simplicity of **or**, leading to easy calculation of the input from the output.

5.2 General idea of the improved fault attack

We adopt the same attack model as in [13] and improve the efficiency of the previous fault attacks in the following two ways:

- To retrieve the 64-bit round-key for some certain round, say the *i*-th round, previous fault attack injects sufficient faults (about $8 \sim 31$) to the same *i*-th round, while ours is to induce (less) faults at both the *i*-th and $(i-1)$-th rounds to deduce the round-key, and it can decrease the number of needed faults.
- In our improved attack, the faulty ciphertexts that are obtained when faults are injected in the *i*-th round are used twice (one for deducing the $(i+1)$-th round-key, and the other for deducing the *i*-th round-key), thus the number of needed faults is reduced (previous fault attack use those ciphertexts only once for deducing the *i*-th round-key).

We briefly summarize the improved fault attack below:

1. Choose an arbitrary plaintext, encrypt it with the secret key and obtain the ciphertext. For the same plaintext, induce several random byte faults into the input of the round function in each round, obtain these faulty ciphertexts.
2. Deduce the last round-key through the right ciphertext and the faulty ciphertexts that are obtained when faults are induced in the last round and penultimate round.
   (a) Consider the right ciphertext and faulty ciphertexts that are obtained when faults are induced into the last round, recover the right input state before the second substitution layer sigma4 in the last round.
   (b) Consider the right ciphertext and faulty ciphertexts that are obtained when faults are induced into the penultimate round, recover the right input state before the first substitution layer sigma4 in the last round.
   (c) Use the right intermediate states obtained from (a)(b) and the correct ciphertext, directly deduce the 64-bit round-key in the last round.
3. Since the last round-key is obtained, we can peel off the last round encryption to obtain the output of the penultimate round, thus the other faulty ciphertexts could be similarly used to recover the last second round-key. In general, this technique can be iteratively adopted to deduce all round-keys from the 2nd round to the 15-th round in the reverse order.
4. Recover the first round-key as described in Section 4.

5.3 Attack procedure

Step 1 Data gathering

Choose an arbitrary plaintext $p = p_l \| p_r$, and obtain the right ciphertext $c = c_l \| c_r$ under the secret user key K. For the same plaintext p, induce several random byte faults into the input of the round function f32 in each round, and obtain these faulty ciphertexts.

Step 2 Recover the last round-key rk_{15}

Step 2.1 Recover the right input state I_2 before the second substitution layer in the last round This is finished by considering the right ciphertext and faulty ciphertexts $c^* = c_l^* \| c_r^*$, which are obtained when faults are induced into the last round. The attack procedure is depicted in Fig. 5.

On the basis of assumption of one random byte fault, ΔA_{16} has $4 \times 255 = 1{,}020$ possibilities, so is ΔB_{16}, thus $\Delta C_{16} = \mathtt{mu4}(\Delta B_{16})$ also has only 1,020 possibilities. The output difference after the second substitution layer can be calculated by $\Delta D_{16} = \Delta L_{16} \oplus \Delta L_{15} = c_l \oplus c_l^*$. For all possible values of $(\Delta C_{16}, \Delta D_{16})$, when apply differential cryptanalysis, this would lead to many possibilities of $I_2 = C_{16} \oplus rk_{15,1}$. We can further decrease the number of I_2 candidates by repeating the above method through other collected faulty ciphertexts, until the candidate set of I_2 has only one element.

Step 2.2 Recover the right input state I_1 before the first substitution layer in the last round This is finished by considering the right input state I_2 deduced in step 2.1

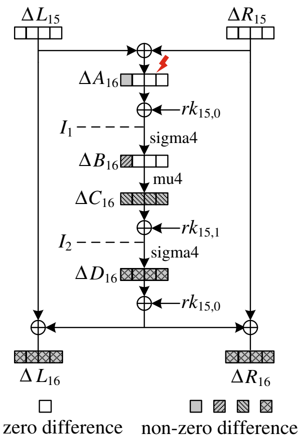


Fig. 5 Attack last round in Step 2.1

and the faulty ciphertexts $c^* = c_l^* \| c_r^*$, which are obtained when faults are induced in the penultimate round. The attack procedure is depicted in Fig. 6.

Since the round-key addition layer doesn't influence the difference, according to proposition 1, both ΔA_{16} and ΔD_{16} could be calculated by $\Delta L_{16} = \Delta c_l$ and $\Delta R_{16} = \Delta c_r$. Moreover, D_{16} is obtained through $D_{16} = \mathtt{sigma4}(I_2)$, thus $D^*_{16} = D_{16} \oplus \Delta D_{16}$ is known, and ΔC_{16} can be obtained as

$$\begin{aligned}\Delta C_{16} &= C_{16} \oplus C^*_{16} \\ &= \mathtt{sigma4}^{-1}\left(D_{16}\right) \oplus rk_{15,1} \oplus \mathtt{sigma4}^{-1}\left(D^*_{16}\right) \oplus rk_{15,1} \\ &= \mathtt{sigma4}^{-1}\left(D_{16}\right) \oplus \mathtt{sigma4}^{-1}\left(D^*_{16}\right),\end{aligned}$$

where $\mathtt{sigma4}^{-1}$ is the inversion of $\mathtt{sigma4}$.

According to $\mathtt{mu4}^{-1}$, ΔB_{16} can be calculated by $\Delta B_{16} = \mathtt{mu4}^{-1}(\Delta C_{16})$. After getting sufficient pairs $(\Delta A_{16}, \Delta B_{16})$ from the correct and faulty ciphertexts, we can apply the differential cryptanalysis on the first substitution layer to uniquely deduce the right input state $I_1 = A_{16} \oplus rk_{15,0}$ before this substitution layer.

Step 2.3 Recover the last round-key $rk_{15,0}$ and $rk_{15,1}$ This is finished by directly calculating these round-keys from the right intermediate state I_1, I_2 as described in Section 4.

Step 3 Recover round-keys from the 2nd round to the 15th round in reverse order

Since the last round-key is obtained, we can peel off the last round to obtain the output of the penultimate round. Then the same technique as described in Step 2 could be used to recover other round-keys. More precisely, we can do as follows:

For $i = 2, 3, \ldots, 15$, consider the right ciphertext and faulty ciphertexts when faults are induced in the $(16 - i)$-th and $(17 - i)$-th round, peel off the last $(i - 1)$-th round(s) according to the deduced round-key(s), and use these outputs of the $(17 - i)$-th round to recover the $(17 - i)$-th round-key. This can be finished by adopting the same technique as in Step 2.

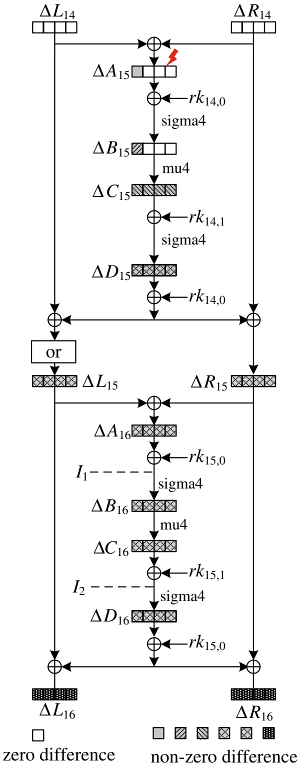


Fig. 6 Attack last round in Step 2.2

Step 4 Recover the first round-key

According to the recovered round-keys from the 2nd round to the 16th round, obtain the outputs of the first round for the right ciphertext and faulty ciphertexts when faults are induced in the first round. Retrieve the first round-key using the same technique as described in Section 4.

5.4 Complexity analysis

To evaluate how many faults are needed to recover the whole round-keys, we firstly concentrate on the one round situation. It is easy to show that, to recover one round-key, the main complexity is dominated by step 2.1 and step 2.2.

In step 2.1, ΔC_{16} has 1,020 possibilities, so to uniquely deduce all bytes of I_2 (thus D_{16}), the number of faulty ciphertexts, denoted by N, must be satisfied

$$256^4 \times \left(\frac{1{,}020}{255^4}\right)^N \leq 1.$$

In general, if the differential distribution table of the S-box is good enough, two faults (i.e. $N = 2$) can uniquely retrieve I_2.

According to the differential property of the Sbox of FOX64, if $N_S(\alpha, \beta) \neq 0$, this difference pair (α, β) will lead to about $2.89 \approx 3$ possible inputs to the Sbox. Then we can use a similar technique [17] to analyze the complexity in step 2.2 as follows:

When the number of faulty ciphertexts is 2, we have one possible value left for I_1 with probability

$$\mathbf{Prob} = \left(\frac{\binom{255}{2} \times \binom{255-2}{2}}{\binom{255}{2}^2} \right)^4 \approx 93.88\%.$$

When the number of faulty ciphertexts is 3, we have one possible value left for I_1 with probability

$$\mathbf{Prob} = \sum_{k=0}^{2} \left(\frac{\binom{255}{2} \times \binom{2}{k} \times \binom{255-2}{2-k}}{\binom{255}{2}^2} \times \frac{\binom{255}{k} \times \binom{255-k}{2}}{\binom{255}{k} \times \binom{255}{2}} \right)^4 \approx 99.95\%.$$

The above analysis indicates that, in most cases, about 4 ∼ 6 faults are enough to uniquely deduce the round-key (excluding the first round). Note that, to recover the first round-key, we have to use the same technique as in [13], i.e. we must induce about 8 ∼ 31 faults. Thus, to retrieve the whole round-keys, we need $2 \times 16 + 6 = 38$ faults in the best case. The general case of the needed faults for obtaining the whole (each) round-key(s) are analyzed in Section 6.

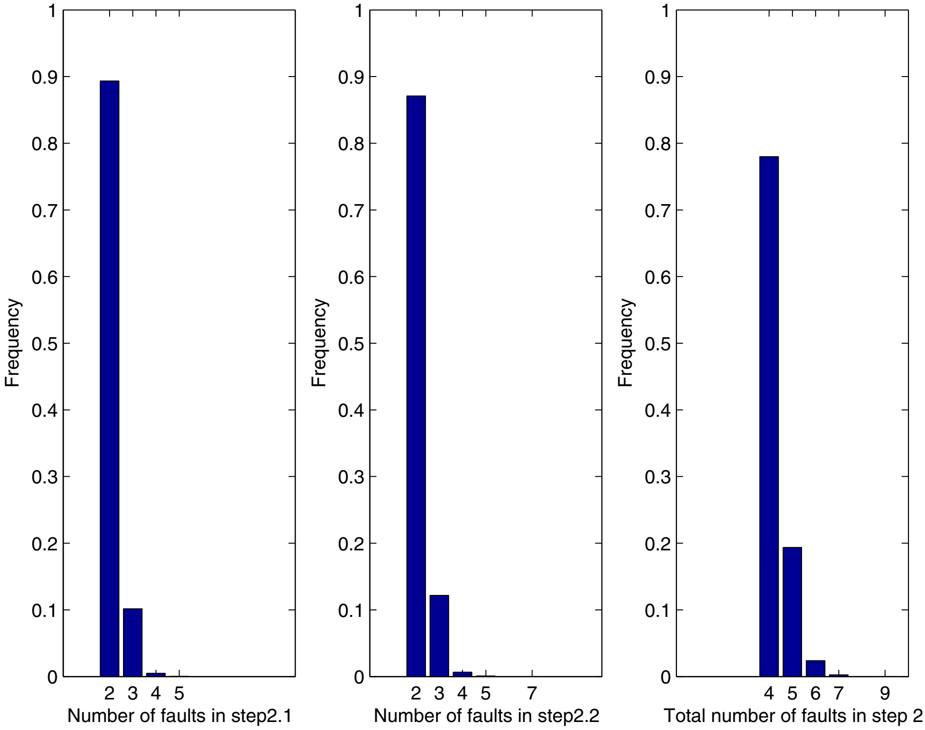


Fig. 7 Simulation results of the improved fault attack against the last round of FOX64

Table 2 Comparison with fault attacks against the last round of FOX64

Fault location	No. of faults minimum	No. of faults average	No. of faults maximum	Source
Last round	8	11.45	31	[13]
Last and penultimate round	4	4.25	9	Section 5

Table 3 Comparison with fault attacks against the whole rounds of FOX64

No. of fault locations	No. of faults best case	No. of faults average	Source
16	128	183.20	[13]
16	38	43.31	Section 5

6 Experimental results

Our proposed fault attack against the last round of FOX64 has been successfully implemented through computer simulation, and the fault injection is simulated by computer software. We implement the attack procedure in C++ code and execute it on a PC with Intel Pentium 1.80 GHz processor. We repeat the attack 10,000 times and the results are shown in Fig. 7.

From Fig. 7, it is observed that step 2.1 requires from 2 to 5 faults (2.11 on average), while step 2.2 requires from 2 to 7 faults (2.14 on average). The complete attack requires from 4 to 9 faults and the average value is 4.25. A comparison between our proposed and the previous fault attack [13] against the last round of FOX64 is shown in Table 2.

Table 2 shows that to derive the last round-key, our improved fault attack requires less faulty ciphertexts, an average of about 4.25 faults compared to 11.45 faults required in the previous fault attack [13]. Thus to recover the whole round-keys, our fault attack needs about $4.25 \times 8 + (11.45 - 2.14) = 43.31$ ($4 \times 8 + (8 - 2) = 38$ in the best case) faults on average, while about $11.45 \times 16 = 183.20$ ($8 \times 16 = 128$ in the best case) faults on average are needed using the method in [13]. This is shown in Table 3.

7 Conclusion

A detailed fault analysis of FOX64 is studied in this paper, after carefully observing a differential property of two round Lai–Massey scheme in a fault model, an improved fault attack on FOX64 is proposed. Compared with the previous attack, the number of needed faults in the proposed attack is significantly reduced, this is caused by the fact that many faulty ciphertexts can be used twice according to the observed property.

We remind that the technique of the new attack on FOX64 can also be extended to other series of the block cipher family FOX. We also point out that, due to the characteristic of the key schedule, it seems hard to improve the efficiency of the fault based attack on FOX.

References

1. Ali S, Mukhopadhyay D, Tunstall M (2010) Differential fault analysis of AES using a single multiple-byte fault. Cryptology ePrint archive: Report 2010/636, available through: http://eprint.iacr.org/2010/636
2. Anderson R, Kuhn M (1996) Tamper resistance—a cautionary note. In: Second USENIX workshop on eletronic commerce, pp 1–11
3. Anderson R, Kuhn M (1997) Low cost attacks on tamper resistant devices. Security protocols 1997. LNCS 1361. Springer, pp 125–136
4. Armknecht F, Meier W (2006) Fault attacks on combiners with memory. In: SAC 2005, LNCS 3897. Springer, pp 36–50
5. Bar-El H, Choukri H, Naccache D, Tunstall M, Whelan C (2006) The sorcerer's apprentice guide to fault attacks. Proc IEEE 94(2):370–386
6. Bertoni G, Breveglieri L, Koren I, Maistri P, Piuri V (2003) Error analysis and detection procedures for a hardware implementation of the advanced encryption standard. IEEE Trans Comput 52(4):492–505
7. Berzati A, Canovas-Dumas C, Goubin L (2009) Fault analysis of Rabbit: toward a secret key leakage. In: INDOCRYPT 2009, LNCS 5922. Springer, pp 72–87
8. Biham E, Shamir A (1997) Differential fault analysis of secret key cryptosystems. In: CRYPTO 97, LNCS 1294. Springer, pp 513–525
9. Biham E, Granboulan L, Nguyn PQ (2005) Impossible fault analysis of RC4 and differential fault analysis of RC4. In: FSE 2005, LNCS 3557. Springer, pp 359–367
10. Blömer J, Seifert J-P (2003) Fault based cryptanalysis of the advanced encryption standard (AES). In: FC 2003, LNCS 2742. Springer, pp 162–181
11. Boneh D, DeMillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: EUROCRYPT'97, LNCS 1233. Springer, pp 37–51
12. Boseher A, Handschuh H (2008) Masking does not protect against differential fault attacks. In: FDTC 2008. IEEE Computer Society, pp 35–40
13. Breveglieri L, Koren I, Maistri P (2006) A fault attack against the FOX cipher family. In: FDTC 2006, LNCS 4236. Springer, pp 98–105
14. Chen C-N, Yen S-M (2003) Differential fault analysis on AES key schedule and some countermeasures. In: ACISP 2003, LNCS 2727. Springer, pp 118–129
15. Clavier C, Gierlichs B, Verbauwhede I (2008) Fault analysis study of IDEA. In: CT-RSA 2008, LNCS 4964. Springer, pp 274–287
16. Dusart P, Letourneux G, Vivolo O (2003) Differential fault analysis on A.E.S. In: ACNS 2003, LNCS 2846. Springer, pp 293–306
17. Giraud C (2005) DFA on AES. In: AES 2004, LNCS 3373. Springer, pp 27–41
18. Giraud C, Thiebeauld H (2004) A survey on fault attacks. In: CARDIS 2004. Springer, pp 159–176
19. Hoch JJ, Shamir A (2004) Fault analysis of stream ciphers. In: CHES 2004. LNCS 3156. Springer, pp 240–253
20. Hojsik M, Rudolf B (2008) Differential fault analysis of Trivium. In: FSE 2008, LNCS 5086. Springer, pp 158–172
21. Hojsik M, Rudolf B (2008) Floating fault analysis of Trivium. In: INDOCRYPT 2008, LNCS 5365. Springer, pp 239–250
22. Junod P, Vaudenay S (2005) FOX: a new family of block ciphers. In: SAC 2004, LNCS 3357. Springer, pp 114–129
23. Karpovsky M, Kulikowski KJ, Taubin A (2004) Robust protection against fault-injection attacks on smart cards implementing the advanced encryption standard. In: DSN 2004. IEEE Computer Society, pp 93–101
24. Kim CH, Quisquater J-J (2008) New differential fault analysis on AES key schedule: two faults are enough. In: CARDIS 2008, LNCS 5189. Springer, pp 48–60
25. Kircanski A, Youssef AM (2009) Differential fault analysis of Rabbit. In: SAC 2009, LNCS 5867. Springer, pp 197–214
26. Kircanski A, Youssef AM (2010) Differential fault analysis of HC-128. In: AFRICACRYPT 2010, LNCS 6055. Springer, pp 261–278
27. Li Y, Sakiyama K, Gomisawa S, Fukunaga T (2010) Fault sensitivity analysis. In: CHES 2010, LNCS 6225. Springer, pp 320–334
28. Luo Y, Lai X, Gong Z (2010) Pseudorandomness analysis of the (extended) Lai–Massey scheme. Inf Process Lett 111(2):90–96 (Elsevier)

29. Maistri P, Leveugle R (2008) Double-data-rate computation as a countermeasure against fault analysis. IEEE Trans Comput 57(11):1528–1539
30. Moradi A, Shalmani MTM, Salmasizadeh M (2006) A generalized method of differential fault attack against AES cryptosystem. In: CHES 2006, LNCS 4249. Springer, pp 91–100
31. Mukhopadhyay D (2009) An improved fault based attack of the advanced encryption standard. In: AFRICACRYPT 2009, LNCS 5580, pp 421–434
32. Piret G, Quisquater J-J (2003) A differential fault attack technque against SPN structures, with application to the AES and KHAZAD. In: CHES 2003, LNCS 2779. Springer, pp 77–88
33. Rajendran J, Borad H, Mantravadi S, Karri R (2010) SLICED: Slide-based concurrent error detection technique for symmetric block ciphers. In: HOST 2010, IEEE International Symposium, pp 70–75
34. Satoh A, Sugawara T, Homma N, Aoki T (2008) High-performance concurrent error detection scheme for AES hardware. In: CHES 2008, LNCS 5154. Springer, pp 100–112
35. Skorobogatov SP, Anderson RJ (2003) Optical fault induction attacks. In: CHES 2002, LNCS 2523. Springer, pp 2–12
36. Takahashi J, Fukunaga T, Yamakoshi K (2007) DFA mechanism on the AES key schedule. In: FDTC 2007. IEEE Computer Society, pp 62–74
37. Vaudenay S (2000) On the Lai–Massey scheme. In: ASIACRYPT 1999, LNCS 1716. Springer, pp 8–19

Ruilin Li received his bachelor and master’s degrees in applied mathematics from National University of Defense Technology in 2005 and 2007. He is currently studying towards PhD degree in department of Mathematics and System Science, National University of Defense Technology, Changsha, China. His research fields include cryptography and information security.

Jianxiong You received his bachelor and master's degrees in applied mathematics from National University of Defense Technology in 2006 and 2010. His research fields include cryptography and information security.

Bing Sun received his master's and doctoral degrees in applied mathematics from the National University of Defense Technology, Changsha, China in 2005 and 2009. He is now a Lecturer with the Department of Mathematics and System Science, National University of Defense Technology of China. His research fields include cryptography and information security.

Chao Li received the bachelor degree in mathematics in 1987 from the University of Information Engineering of China, the master's degree in mathematics in 1990 from the University of Science and Technology of China, and the doctoral degree in engineering in 2002 from the National University of Defense Technology of China. Since December 2001, he has been a Professor with the Department of Mathematics and System Science, National University of Defense Technology. His research fields include: coding theory, cryptography and sequences.