Improved Integral Cryptanalysis of FOX Block Cipher[1]

Wu Wenling, Zhang Wentao, and Feng Dengguo

State Key Laboratory of Information Security, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, P. R. China
E-mail:{wwl}@is.iscas.ac.cn

Abstract. FOX is a new family of block ciphers presented recently, which is based upon some results on proven security and has high performances on various platforms. In this paper, we construct some distinguishers between 3-round FOX and a random permutation of the blocks space. By using integral attack and collision-searching techniques, the distinguishers are used to attack on 4, 5, 6 and 7-round of FOX64, 4 and 5-round FOX128. The attack is more efficient than previous integral attack on FOX. The complexity of improved integral attack is $2^{77.6}$ on 4-round FOX128, $2^{205.6}$ against 5-round FOX128 respectively. For FOX64, the complexity of improved integral attack is $2^{45.4}$ on 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64 respectively. Therefore, 4-round FOX64/64, 5-round FOX64/128, 6-round FOX64/192, 7-round FOX64/256 and 5-round FOX128/256 are not immune to the attack in this paper.

Key words: Block cipher; FOX; Data complexity; Time complexity; Integral Cryptanalysis.

1 Introduction

FOX[1] is a new family of block ciphers, which is the result of a joint project with the company MediaCrypt[2] AG in Zurich, Switzerland. Fox has two versions, both have a variable number rounds which depends on keysize: the first one FOX64/k/r has a 64-bit blocksize with a variable key length which is a multiple of 8 and up to 256 bits. The second one FOX128/k/r uses a 128-bit blocksize with the same possible key lengths. For FOX64 with k=128 and FOX128 with k=256, the designers advise that round number are both 16. The high level of FOX adopts a modified structure of Lai-Massey Scheme[3], which can be proven to have good pseudorandomness properties in the Luby-Rackoff paradigm and decorrelation inheritance properties proposed by Vaudenay[4]. The round function of FOX uses SPS (Substitution-Permutation-Substitution) structure

[1] Supported partially by the National Natural Science Foundation of China under Grant No. 60373047; the National Basic Research 973 Program of China under Grant No.2004CB318004;and the National High-technique 863 Program of China under Grant No.2003AA14403

with three layers of subkey addition, SPS structure has already been proven to have powerful ability to resist differential and linear cryptanalysis. The design rationale of diffusion primitives in FOX is presented in Ref.[5]. The key schedule of FOX is very complex compared with other existing block ciphers, each subkey of FOX is related to the seed key and it's very difficult to acquire information about seed key or other subkeys from some certain subkeys. The complex key schedule, high-level structure with provable security and powerful round function make FOX appear to be a strong block cipher. Since FOX is a new cipher published last year, all we know about its security analysis are limited to be the designer's results and the integral cryptanalysis presented in Ref.[6]. The security of FOX against differential and linear cryptanalysis is easy to estimate for the good property of its S-box, SPS transformation and high level structures. The designers also analyze the security of FOX against differential-linear cryptanalysis[7,8], boomerang[9] and rectangle attacks[10], truncated and higher-order differentials[11], impossible differentials[12],and partitioning cryptanalysis[13,14], algebraic attack[15,16], slide attack[17,18], and related-cipher attacks[19]. Integral attack[20] is one of the most effective attack method against AES, which had been used to analyze the security of other ciphers[20,21]. It's pointed in Ref.[1] that integral attack has a complexity of 2^{72} encryptions against 4-round FOX64, 2^{136} against 5-round FOX64, 2^{200} against 6-round FOX64. The authors also claimed that integral attack has a complexity of 2^{136} encryptions against 4-round FOX128,and 5-round FOX128 is immune to integral attack. In this paper, we combine collision technique and integral attack to analyze the security of FOX. The improved integral attack on FOX is more efficient than known integral attack. The complexity of our improved integral attack is $2^{77.6}$ on 4-round FOX128, $2^{205.6}$ against 5-round FOX128 respectively. For FOX64, the complexity of improved integral attack is $2^{45.4}$ on 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64 respectively.

This paper is organized as follows: Section 2 briefly introduces the structure of FOX128. 3-round distinguishers are presented in section 3. In section 4, we show how to use the 3-round distinguishers to attack 4 and 5 rounds of FOX128. In section 5, we briefly introduce the attacks on 4,5,6 and 7 rounds of FOX64. and Section 6 concludes the paper.

2 Description of FOX

The different members of FOX family are denoted as follows:

Name	Block size	Key size	Round number
FOX64	64	128	16
FOX128	64	128	16
FOX64/k/r	64	k	r
FOX128/k/r	128	k	r

In FOX64/k/r and FOX128/k/r, the round number r must satisfy $12 \leq r \leq 255$,

while the key length k must satisfy $0 \leq k \leq 256$, with k multiple of 8. Due to space limitation, we only introduce FOX128 briefly. Details are shown in Ref.[1]

2.1 Round Function f64

The round function **f64** consists of three main parts: a *substitution* part denoted **sigma8**, a *diffusion* part denoted **MU8**, and a *round key addition* part (see Fig.1). Formally, the i-th round function $f64^i$ takes a 64-bit input $X^i_{(64)} = X^i_{0(8)}||X^i_{1(8)}||\cdots||X^i_{7(8)}$, a 128-bit round key $RK^i_{(128)} = RK0^i_{(64)}||RK1^i_{(64)}$ and returns

$$Y^i_{(64)} = Y^i_{0(8)}||\cdots||X^i_{7(8)} = sigma8(MU8(sigma8(X^i_{(64)} \oplus RK0^i_{(64)})) \oplus RK1^i_{(64)}) \oplus RK0^i_{(64)}.$$

The mapping **sigma8** consists of 8 parallel computations of a non-linear bijective mapping(see the table in Ref.[1]). **MU8** considers an input $(Z_{0(8)}||\cdots||Z_{7(8)})$ as a vector $(Z_{0(8)}||\cdots||Z_{7(8)})^T$ over $GF(2^8)$ and multiply it with a matrix to obtain an output vector of the same size. The matrix is the following:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & a \\ 1 & a & b & c & d & e & f & 1 \\ a & b & c & d & e & f & 1 & 1 \\ b & c & d & e & f & 1 & a & 1 \\ c & d & e & f & 1 & a & b & 1 \\ d & e & f & 1 & a & b & c & 1 \\ e & f & 1 & a & b & c & d & 1 \\ f & 1 & a & b & c & d & e & 1 \end{pmatrix}$$

where $a = \alpha + 1$, $b = \alpha^7 + \alpha$, $c = \alpha, d = \alpha^2$, $e = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$ and $f = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha$. α is a root of the irreducible polynomial $m(x) = x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + 1$.

2.2 Encryption of FOX128

FOX128 is the 15-times iteration of *round transformation* **elmor128**, followed by the application of last round transformation called **elmid128**. **elmor128**, illustrated in Fig.2, is built as an Extended Lai-Massey scheme combined with with two orthomorphisms *or*.

The i-th round transformation—**elmor128** transforms a 128-bit input $LL^i_{(32)}||LR^i_{(32)}||RL^i_{(32)}||RR^i_{(32)}$ and a 128-bit round key $RK^i_{(128)}$ in a 128-bit output $LL^{i+1}_{(32)}||LR^{i+1}_{(32)}||RL^{i+1}_{(32)}||RR^{i+1}_{(32)}$. Let $LL^i_{(32)} \oplus LR^i_{(32)}||RL^i_{(32)} \oplus RR^i_{(32)} = X^i_{(64)} = X^i_{0(8)}||X^i_{1(8)}||\cdots||X^i_{7(8)})$ and $f64^i(X^i_{(64)}, RK^i_{(128)}) = \phi_L||\phi_R$. Then,

$$LL^{i+1}_{(32)}||LR^{i+1}_{(32)}||RL^{i+1}_{(32)}||RR^{i+1}_{(32)} = or(LL^i_{(32)} \oplus \phi_L)||LR^i_{(32)} \oplus \phi_L||or(RL^i_{(32)} \oplus \phi_R)||RR^i_{(32)} \oplus \phi_R.$$

The **elmid128** function is a slightly modified version of **elmor128**, namely the two transformations *or* are replaced by two identity transformation.

The transformation *or* is a function taking a 32-bit input $X_{(32)} = X_{0(16)}||X_{1(16)}$ and returning a 32-bit output $Y_{(32)} = Y_{0(16)}||Y_{1(16)}$ which is in fact a one-round Feistel

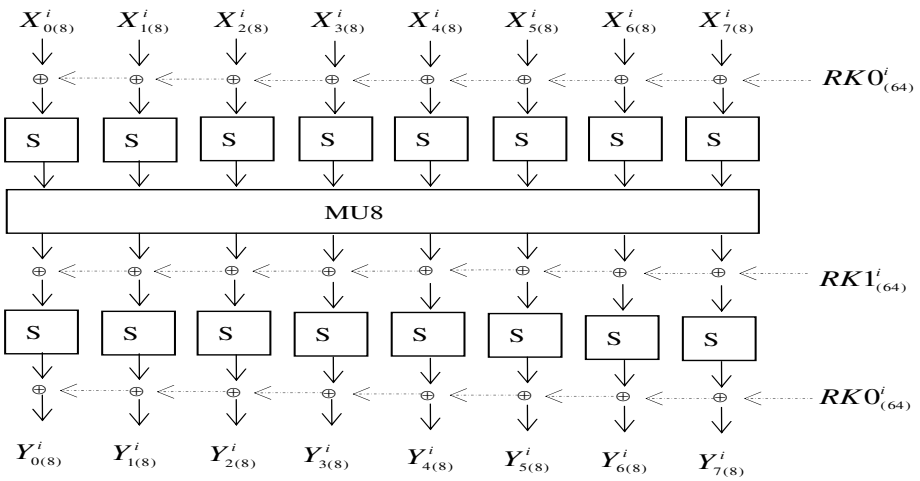

Fig. 1. The i-th Round Function of FOX128

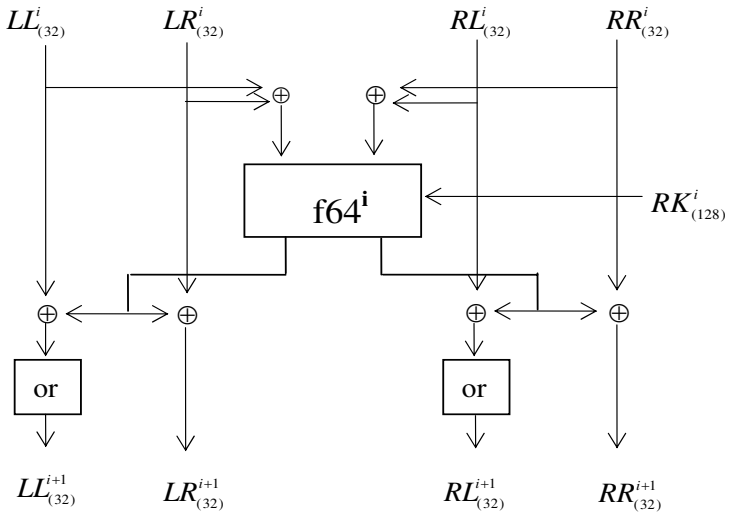

Fig. 2. The i-th Round Transformation—**elmor128**

scheme with the identity function as round function; it is defined as $Y_{0(16)}||Y_{1(16)} = X_{1(16)}||(X_{0(16)} \oplus X_{1(16)})$

The encryption C_{128} by FOX128 of a 128-bit plaintext P_{128} is defined as

$$C_{128} = elmid128(elmor128(\ldots(elmor128(P_{128}, RK^1_{(128)}), \ldots, RK^{15}_{(128)})RK^{16}_{(128)})$$

where $RK^1_{(128)}, \ldots, RK^{16}_{(128)}$ are round subkeys produced by the key schedule algorithm out of the user key. In this paper, subkeys are assumed to be independent of each other. So we omit the key schedule of FOX in this paper.

3 3-Round Distinguishers

Choose plaintexts $P_{(128)} = LL^1_{(32)}||LR^1_{(32)}||RL^1_{(32)}||RR^1_{(32)}$ as follows:

$$LL^1_{(32)} = LR^1_{(32)} = c||c||c||c, \qquad RL^1_{(32)} = RR^1_{(32)} = c||c||c||x.$$

where x take values in $\{0,1\}^8$, c is a constant in $\{0,1\}^8$. Thus, the input of the first round function $f64^1$ is $X^1_{(64)} = 0||0\ldots||0$. Let $f64^1(0||0\ldots||0) = (a_0||a_1\ldots a_7)$, where $a_i(0 \le i \le 7)$ are entirely determined by round subkey $RK^1_{(128)}$, so $a_i(0 \le i \le 7)$ are constants when the user key is fixed. Then the output of the 1st round can be written as follows:

$$\begin{aligned}
LL^2_{(32)} &= a_2 \oplus c||a_3 \oplus c||a_0 \oplus a_2||a_1 \oplus a_3,\\
LR^2_{(32)} &= a_0 \oplus c||a_1 \oplus c||a_2 \oplus c||a_3 \oplus c,\\
RL^2_{(32)} &= a_6 \oplus c||a_7 \oplus x||a_4 \oplus a_6||a_5 \oplus a_7 \oplus x \oplus c,\\
LR^2_{(32)} &= a_4 \oplus c||a_5 \oplus c||a_6 \oplus c||a_7 \oplus x,
\end{aligned}$$

Therefore, the input of the 2nd round function $f64^2$ is the following: $X^2_{(64)} = X^2_{0(8)}||X^2_{1(8)}\ldots||X^2_{7(8)}$.

$$\begin{aligned}
X^2_{0(8)} &= a_0 \oplus a_2, & X^2_{4(8)} &= a_4 \oplus a_6,\\
X^2_{1(8)} &= a_1 \oplus a_3, & X^2_{5(8)} &= a_5 \oplus a_7 \oplus c \oplus x,\\
X^2_{2(8)} &= a_0 \oplus c, & X^2_{6(8)} &= a_4 \oplus c,\\
X^2_{3(8)} &= a_1 \oplus c, & X^2_{7(8)} &= a_5 \oplus c.
\end{aligned}$$

Let $f64^2(X^2_{(64)}) = (y_0||y_1\ldots y_7)$, then the output of the 2nd round can be written as follows:

$$\begin{aligned}
LL^3_{(32)} &= y_2 \oplus a_0 \oplus a_2||y_3 \oplus a_1 \oplus a_3||y_0 \oplus y_2 \oplus a_0 \oplus c||y_1 \oplus y_3 \oplus a_1 \oplus c,\\
LR^3_{(32)} &= y_0 \oplus a_0 \oplus c||y_1 \oplus a_1 \oplus c||y_2 \oplus a_2 \oplus c||y_3 \oplus a_3 \oplus c,\\
RL^3_{(32)} &= y_6 \oplus a_4 \oplus a_6||y_7 \oplus a_5 \oplus a_7 \oplus c \oplus x||y_4 \oplus y_6 \oplus a_4 \oplus c||y_5 \oplus y_7 \oplus a_5 \oplus c,\\
RR^2_{(32)} &= y_4 \oplus a_4 \oplus c||y_5 \oplus a_5 \oplus c||y_6 \oplus a_6 \oplus c||y_7 \oplus a_7 \oplus x.
\end{aligned}$$

So the input of the 3rd round function $f64^3$ is the following: $X^3_{(64)} = X^3_{0(8)}||X^3_{1(8)}\ldots||X^3_{7(8)}$.

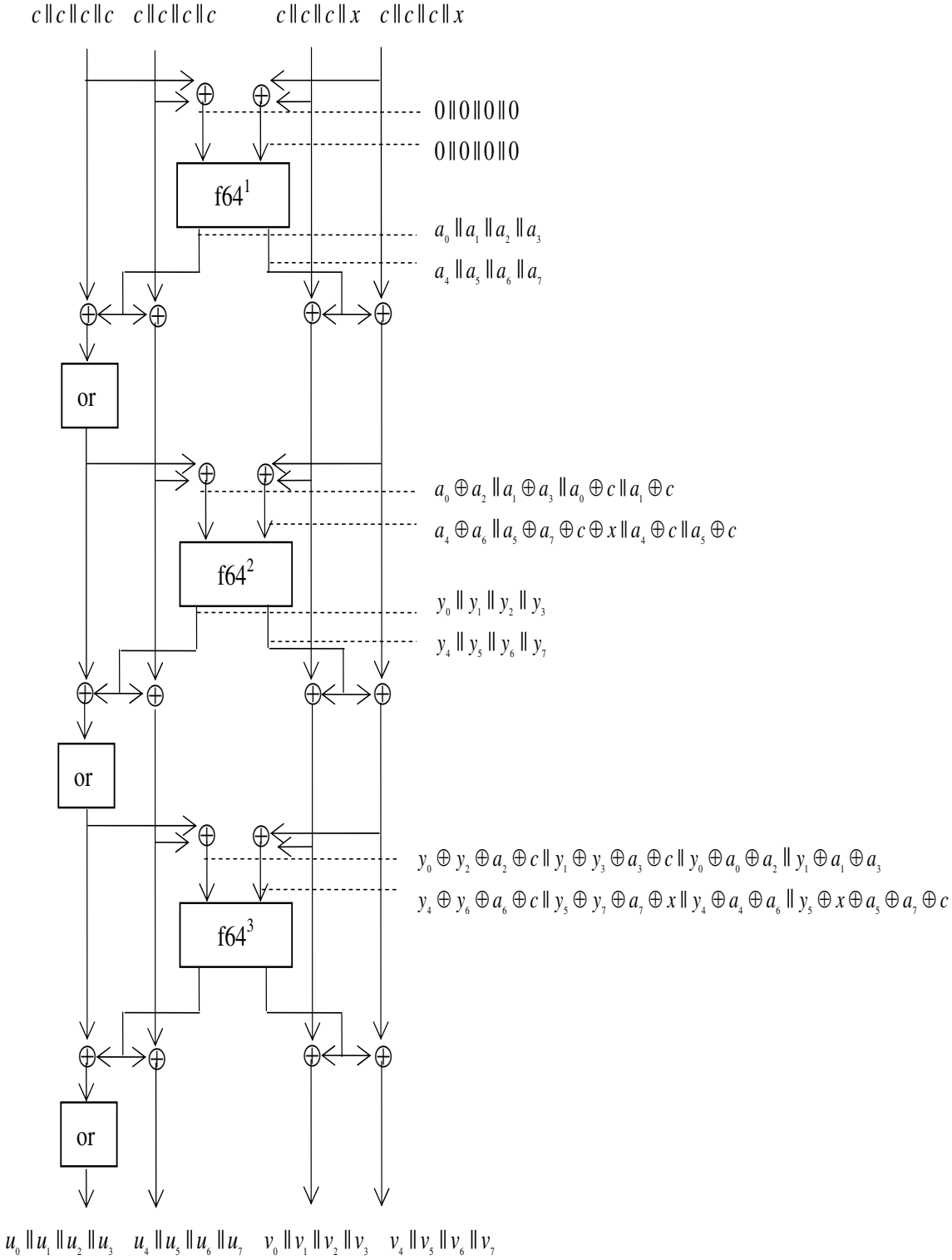

Fig. 3. 3-Round Distinguishers of FOX128

$$
\begin{aligned}
X^3_{0(8)} &= y_0 \oplus y_2 \oplus a_2 \oplus c, & X^3_{4(8)} &= y_4 \oplus y_6 \oplus a_6 \oplus c,\\
X^3_{1(8)} &= y_1 \oplus y_3 \oplus a_3 \oplus c, & X^3_{5(8)} &= y_5 \oplus y_7 \oplus a_7 \oplus x,\\
X^3_{2(8)} &= y_0 \oplus a_0 \oplus a_2, & X^3_{6(8)} &= y_4 \oplus a_4 \oplus a_6,\\
X^3_{3(8)} &= y_1 \oplus a_1 \oplus a_3, & X^3_{7(8)} &= y_5 \oplus x \oplus a_5 \oplus a_7.
\end{aligned}
$$

By observing the high-level structure of FOX128, we get

$$
\begin{aligned}
or^{-1}(LL^4_{(32)}) \oplus LR^4_{(32)} &= X^3_{0(8)}||X^3_{1(8)}||X^3_{2(8)}||X^3_{3(8)},\\
or^{-1}(RL^4_{(32)}) \oplus RR^4_{(32)} &= X^3_{4(8)}||X^3_{5(8)}||X^3_{6(8)}||X^3_{7(8)}.
\end{aligned}
$$

From the definition of or^{-1}, we have

$$
or^{-1}(LL^4_{(32)}) = LL^4_{0(8)} \oplus LL^4_{2(8)}||LL^4_{1(8)} \oplus LL^4_{3(8)}||LL^4_{0(8)}||LL^4_{1(8)},
$$

$$
or^{-1}(RL^4_{(32)}) = RL^4_{0(8)} \oplus RL^4_{2(8)}||RL^4_{1(8)} \oplus RL^4_{3(8)}||RL^4_{0(8)}||RL^4_{1(8)},
$$

Thus,we have the following from the above equations.

$$
\begin{aligned}
LL^4_{0(8)} \oplus LL^4_{2(8)} \oplus LR^4_{0(8)} &= y_0 \oplus y_2 \oplus a_2 \oplus c,\\
LL^4_{1(8)} \oplus LL^4_{3(8)} \oplus LR^4_{1(8)} &= y_1 \oplus y_3 \oplus a_3 \oplus c,\\
LL^4_{0(8)} \oplus LR^4_{2(8)} &= y_0 \oplus a_0 \oplus a_2,\\
LL^4_{1(8)} \oplus LR^4_{3(8)} &= y_1 \oplus a_1 \oplus a_3,\\
RL^4_{0(8)} \oplus RL^4_{2(8)} \oplus RR^4_{0(8)} &= y_4 \oplus y_6 \oplus a_6 \oplus c,\\
RL^4_{1(8)} \oplus RL^4_{3(8)} \oplus RR^4_{1(8)} &= y_5 \oplus y_7 \oplus a_7 \oplus x,\\
RL^4_{0(8)} \oplus RR^4_{2(8)} &= y_4 \oplus a_4 \oplus a_6,\\
RL^4_{1(8)} \oplus RR^4_{3(8)} &= y_5 \oplus x \oplus a_5 \oplus a_7.
\end{aligned}
$$

Further we have the following:

$$
\begin{aligned}
LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)} &= y_2 \oplus a_0 \oplus c,\\
LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)} &= y_3 \oplus a_1 \oplus c,\\
LL^4_{0(8)} \oplus LR^4_{2(8)} &= y_0 \oplus a_0 \oplus a_2,\\
LL^4_{1(8)} \oplus LR^4_{3(8)} &= y_1 \oplus a_1 \oplus a_3,\\
RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)} &= y_6 \oplus a_4 \oplus c,\\
RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)} &= y_7 \oplus a_5,\\
RL^4_{0(8)} \oplus RR^4_{2(8)} &= y_4 \oplus a_4 \oplus a_6,
\end{aligned}
$$

Now we analyze the property of $y_i (0 \le i \le 7)$. Let $y = s(x \oplus a_5 \oplus a_7 \oplus c \oplus RK0^2_{0(8)})$, then $y_i = s(y \oplus b_i) \oplus RK0^2_{i(8)}$, here $b_i (0 \le i \le 7)$ are entirely determined by $a_i (0 \le i \le 7)$, c and $RK^2_{(128)}$, so $b_i (0 \le i \le 7)$ are constants when the user key is fixed.

Because s is a permutation, $y = s(x \oplus a_5 \oplus a_7 \oplus c \oplus RK0^2_{0(8)})$ differs when x takes different values and the user key is fixed. As a consequence, $y_i = s(y \oplus b_i) \oplus RK0^2_{i(8)}$ will have different values when x takes different values and the user key is fixed. Thus, from the above discussion we know that $LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)}$, $LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)}$, $LL^4_{0(8)} \oplus LR^4_{2(8)}$, $LL^4_{1(8)} \oplus LR^4_{3(8)}$, $RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)}$, $RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)}$

and $RL^4_{0(8)} \oplus RR^4_{2(8)}$ each will have different values when x takes different values. Therefore we get the following theorem.

Theorem 1. *Let* $P_{(128)} = LL^1_{(32)}||LR^1_{(32)}||RL^1_{(32)}||RR^1_{(32)}$ *and* $P^*_{(128)} = LL^{1*}_{(32)}||LR^{1*}_{(32)}||RL^{1*}_{(32)}||RR^{1*}_{(32)}$ *be two plaintexts of 3-round FOX128,* $C_{(128)} = LL^4_{(32)}||LR^4_{(32)}||RL^4_{(32)}||RR^4_{(32)}$ *and* $C^*_{(128)} = LL^{4*}_{(32)}||LR^{4*}_{(32)}||RL^{4*}_{(32)}||RR^{4*}_{(32)}$ *be the corresponding ciphertexts.* $RR_{i(8)}(0 \leqslant i \leqslant 7)$ *denotes the* $(i+1)^{th}$ *byte of* $RR_{(32)}$*. If* $LL^1_{(32)} = LR^1_{(32)} = LL^{1*}_{(32)} = LR^{1*}_{(32)}, RL^1_{(32)} = RR^1_{(32)}$, $RL^{1*}_{(32)} = RR^{1*}_{(32)}$, $RR^1_{i(8)} = RR^{1*}_{i(8)}(i = 0, 1, 2)$, $RR^1_{3(8)} \neq RR^{1*}_{3(8)}$, *then* $C_{(128)}$ *and* $C^*_{(128)}$ *satisfy the following inequalities:*

$$LL^4_{2(8)} \oplus LR^4_{0(8)} \oplus LR^4_{2(8)} \neq LL^{4*}_{2(8)} \oplus LR^{4*}_{0(8)} \oplus LR^{4*}_{2(8)}, \tag{1}$$

$$LL^4_{3(8)} \oplus LR^4_{1(8)} \oplus LR^4_{3(8)} \neq LL^{4*}_{3(8)} \oplus LR^{4*}_{1(8)} \oplus LR^{4*}_{3(8)} \tag{2}$$

$$LL^4_{0(8)} \oplus LR^4_{2(8)} \neq LL^{4*}_{0(8)} \oplus LR^{4*}_{2(8)} \tag{3}$$

$$LL^4_{1(8)} \oplus LR^4_{3(8)} \neq LL^{4*}_{1(8)} \oplus LR^{4*}_{3(8)} \tag{4}$$

$$RL^4_{2(8)} \oplus RR^4_{0(8)} \oplus RR^4_{2(8)} \neq RL^{4*}_{2(8)} \oplus RR^{4*}_{0(8)} \oplus RR^{4*}_{2(8)} \tag{5}$$

$$RL^4_{3(8)} \oplus RR^4_{1(8)} \oplus RR^4_{3(8)} \neq RL^{4*}_{3(8)} \oplus RR^{4*}_{1(8)} \oplus RR^{4*}_{3(8)} \tag{6}$$

$$RL^4_{0(8)} \oplus RR^4_{2(8)} \neq RL^{4*}_{0(8)} \oplus RR^{4*}_{2(8)} \tag{7}$$

From the above discussion, we have $RL^4_{1(8)} \oplus RR^4_{3(8)} = y_5 \oplus x \oplus a_5 \oplus a_7$, and y_5 will have different values when x take different values. So we can get the following Corollary, which is similar to the integral distinguisher presented in Ref.[1] and Ref.[6].

Corollary 1. *Let* $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)}(0 \leq j \leq 255)$ *be 256 plaintexts of 3-round FOX128,* $Cj_{(128)} = LLj^4_{(32)}||LRj^4_{(32)}||RLj^4_{(32)}||RRj^4_{(32)}$ *be the corresponding ciphertexts. If* $LLj^1_{(32)} = LRj^1_{(32)}, RLj^1_{(32)} = RLj^1_{(32)}$, $RLj_{i(8)}(i = 0, 1, 2)$ *are constants, and* $LRj_{3(8)}$ *take all possible values between 0 and 255, then* $Cj_{(128)}(0 \leq j \leq 255)$ *satisfy:*

$$\bigoplus_{j=0}^{255}(RLj^4_{1(8)} \oplus RRj^4_{3(8)}) = 0 \tag{8}$$

4 Attacks on Reduced-Round FOX128

4.1 Attacking 4-round FOX128

This section explains the attack on 4-round FOX128 in detail. The last round omit the *or* transformation. First we recover 72 bits subkey $RK0^4_{(64)}$ and $RK1^4_{0(8)}$.

Choose plaintext $P_{(128)} = LL^1_{(32)}||LR^1_{(32)}||RL^1_{(32)}||RR^1_{(32)}$, and let $C_{(128)} = LL^5_{(32)}||LR^5_{(32)}||RL^5_{(32)}||RR^5_{(32)}$ be the corresponding ciphertext. The input of the fourth round function $f64^4$ is $LL^5_{(32)} \oplus LR^5_{(32)}||RL^5_{(32)} \oplus RR^5_{(32)}$, and we can calculate the value of $LL^4_{2(8)} \oplus LR^4_{2(8)}$ because $LL^4_{2(8)} \oplus LR^4_{2(8)} = LL^5_{2(8)} \oplus LR^5_{2(8)}$. If we guess the value of $LR^4_{0(8)}$, then we can guess $LL^4_{2(8)} \oplus LR^4_{2(8)} \oplus LR^4_{0(8)}$. From the structure of $f64^4$, it is known that the value of $LR^4_{0(8)}$ is entirely determined by the input $LL^5_{(32)} \oplus LR^5_{(32)}||RL^5_{(32)} \oplus RR^5_{(32)}$ and subkey $RK0^4_{(64)}$, $RK1^4_{0(8)}$. Thus using the inequality (1) of **Theorem 1** , we construct the following algorithm to recover $RK0^4_{(64)}$ and $RK1^4_{0(8)}$.

Algorithm 1

Step1, Choose 166 plaintexts $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)}(0 \leq j \leq 165)$ as follows:

$$LLj^1_{(32)} = (c||c||c||c),$$
$$LRj^1_{(32)} = (c||c||c||c),$$
$$RLj^1_{(32)} = (c||c||c||j),$$
$$RRj^1_{(32)} = (c||c||c||j).$$

where c is a constant, $0 \leq j \leq 165$. The corresponding ciphertexts are $Cj_{(128)} = LLj^5_{(32)}||LRj^5_{(32)}||RLj^5_{(32)}||RRj^5_{(32)}$.

Step2, For each possible value of $RK0^4_{(64)}||RK1^4_{0(8)}$, first compute the first byte $Yj^4_{0(8)}$ of $f64^4(LLj^5_{(32)} \oplus LRj^5_{(32)}||RLj^5_{(32)} \oplus RRj^5_{(32)})$, and then compute

$$\triangle_j = Yj^4_{0(8)} \oplus LLj^5_{2(8)} \oplus LRj^5_{2(8)} \oplus LRj^5_{0(8)}.$$

Check if there ia a collision among $\triangle_j$. If so, discard the value of $RK0^4_{(64)}||RK1^4_{0(8)}$. Otherwise, output $RK0^4_{(64)}||RK1^4_{0(8)}$.

Step3, From the output values of $RK0^4_{(64)}||RK1^4_{0(8)}$ in Step2, choose some other plaintexts, and repeat Step2.

The probability of at least one collision occurs when we throw 166 balls into 256 buckets at random is larger than $1 - e^{-166(166-1)/2\times 2^8} \geq 1 - 2^{-76}$. So the probability of passing the test of Step 2 is less than2^{-76}. Because the right subkey candidates must pass the test of Step2, the number of subkey candidates passing Step2 is about $1 + (2^{72} \times 2^{-76}) \approx 1.06$. Then, only two plaintexts are needed in Step3. The data complexity of this attack is about 168 chosen plaintexts. And the main time complexity of **Algorithm 2** is in step2, the time of computing each $\triangle_j$ is less than 1-round encryption, so the time complexity is less than $2^{72} \times 168/4 \approx 42 \times 2^{72}$ encryptions.

Next we recover $RK1^4_{1(8)}$. The steps are very similar to **Algorithm1**, except $RK0^4_{(64)}$ is known here. So the number of candidates is 2^8, only 64 chosen plaintexts are needed(we can use the data in **Algorithm 1** again). Using the inequality (2) in **Theorem 1**, we can recover $RK1^4_{1(8)}$ by computing

$$\triangle_j = Yj^4_{1(8)} \oplus LLj^5_{3(8)} \oplus LRj^5_{3(8)} \oplus LRj^5_{1(8)}.$$

and the attack requires $2^8 \times 64/4 = 2^{12}$ encryptions.

Knowing $RK0^4_{(64)}$ and $RK1^4_{0(8)}$, using inequality (3) in **Theorem 1** and the plaintexts chosen in **Algorithm 1**, we can recover $RK1^4_{2(8)}$ by computing

$$\triangle_j = Yj^4_{0(8)} \oplus Yj^4_{2(8)} \oplus LLj^5_{0(8)} \oplus LRj^5_{2(8)}$$

and the attack requires 2^{12} encryptions.

Similarly, knowing $RK0^4_{(64)}$ and $RK1^4_{1(8)}$, using inequality (4) in Theorem 1 and the plaintexts chosen in **Algorithm 1**, we can recover $RK1^4_{3(8)}$ by computing

$$\triangle_j = Yj^4_{1(8)} \oplus Yj^4_{3(8)} \oplus LLj^5_{1(8)} \oplus LRj^5_{3(8)}$$

and the attack requires 2^{12} encryptions.

Furthermore,using inequality (5) in Theorem 1 and the plaintexts chosen in **Algorithm 1**, we can recover $RK1^4_{4(8)}$ by computing

$$\triangle_j = Yj^4_{4(8)} \oplus RLj^5_{2(8)} \oplus RRj^5_{2(8)} \oplus RRj^5_{0(8)}$$

and the attack requires 2^{12} encryptions.

And using inequality (6) in **Theorem 1** and the plaintexts chosen in **Algorithm 1**, we can recover $RK1^4_{5(8)}$ by computing

$$\triangle_j = Yj^4_{5(8)} \oplus RLj^5_{3(8)} \oplus RRj^5_{3(8)} \oplus RRj^5_{1(8)}$$

and the attack requires 2^{12} encryptions.

Knowing $RK0^4_{(64)}$ and $RK1^4_{4(8)}$, using iinequality(7) in Theorem 1 and the plaintexts chosen in **Algorithm 1**, we can recover $RK1^4_{6(8)}$ by computing

$$\triangle_j = Yj^4_{4(8)} \oplus RLj^5_{0(8)} \oplus Yj^5_{6(8)} \oplus RRj^5_{2(8)}$$

and the attack requires 2^{12} encryptions.

We can't use similar approach to recover $RK1^4_{7(8)}$, fortunately integral technique can be used here. Knowing $RK0^4_{(64))}$ and $RK1^4_{5(8)}$, using equation (8) in Theorem 1,we can construct the following algorithm to recover $RK1^4_{7(8)}$.

Algorithm 2

Step1, Choose 256 plaintexts $Pj_{(128)} = LLj^1_{(32)}||LRj^1_{(32)}||RLj^1_{(32)}||RRj^1_{(32)}(0 \leq j \leq 122)$ as follows:

$$\begin{aligned} LLj^1_{(32)} &= (c||c||c||c), \\ LRj^1_{(32)} &= (c||c||c||c), \\ RLj^1_{(32)} &= (c||c||c||j), \\ RRj^1_{(32)} &= (c||c||c||j). \end{aligned}$$

where c is a constant, $0 \leq j \leq 255$. The corresponding ciphertexts are $Cj_{(128)} = LLj^5_{(32)}||LRj^5_{(32)}||RLj^5_{(32)}||RRj^5_{(32)}$.

Step2, For each possible value of $RK1^4_{7(8)}$, first compute $Yj^4_{7(8)}$, and then compute

$$\Delta = \bigoplus_{j=0}^{255}(RL^5_{1(8)} \oplus RR^5_{3(8)} \oplus Yj^4_{5(8)} \oplus Yj^4_{7(8)}).$$

Check if $\Delta = 0$. If not, discard the value of $RK1^4_{7(8)}$, Otherwise, output $RK1^4_{7(8)}$.

Step3, From the output values of $RK1^4_{7(8)}$ in Step2, choose another group of plaintexts, and repeat Step2 until the key candidate is unique.

Wrong values will pass step2 successfully with probability 2^{-8}. Thus **Algorithm 2** requires about 2^9 chosen plaintexts, and the time complexity is about $2^9 \times 2^8/4 = 2^{15}$ encryptions. The data in **Algorithm 1** can be repeatedly used here again, so the data complexity for recovering $RK^4_{(128))}$ is about 2^9, and the time complexity is about $42 \times 2^{72} + 6 \times 2^{12} + 2^{15}$.

Now we have recovered $RK^4_{(128)}$ using 2^9 chosen plaintexts and $42\times2^{72}+6\times2^{12}+2^{15}$ encryptions. By decrypting the 4th round, we can recover $RK^3_{(128)}$, the time complexity is less than $2^{73} + 6 \times 2^{12} + 2^{15}$. Similarly, we can recover $RK^2_{(128))}$ and $RK^1_{(128))}$, the time complexity are both less than $2^{73} + 6 \times 2^{12} + 2^{15}$. Therefore, the attack on the 4-round FOX128 requires 2^9 chosen plaintexts and about $2^{77.6}$ encryptions.

4.2 Attacking 5-round FOX128

We could extend the previous attack on 5-round FOX128, using a key exhaustive search on the fifth subkey $RK^5_{(128)}$. The attack requires $2^{205.6}$ encryptions, which is less expensive than a key exhaustive search.

5 Attacks on Reduced-Round FOX64

Similar to FOX128, we can get the following theorem for FOX64.

Theorem 2. *Let* $P_{(64)} = L^1_{(32)}||R^1_{(32)}$ *and* $P^*_{(64)} = L^{1*}_{(32)}||R^{1*}_{(32)}$ *be two plaintexts of 3-round FOX64,*$C_{(64)} = L^4_{(32)}||R^4_{(32)}$ *and* $C^*_{(64)} = L^{4*}_{(32)}||R^{4*}_{(32)}$ *be the corresponding ciphertexts,* $L_{i(8)}$ *denotes the* $(i+1)^{th}$ *byte of* $L_{(32)}$*, If* $L^1_{(32)} = R^1_{(32)}, L^{1*}_{(32)} = R^{1*}_{(32)}$*, and* $L^1_{i(8)} = L^{1*}_{i(8)} (i = 0, 1, 2)$,$L^1_{3(8)} \neq L^{1*}_{3(8)}$*, then* $C_{(64)}$ *and* $C^*_{(64)}$ *satisfy:*

$$L^4_{2(8)} \oplus R^4_{2(8)} \oplus R^4_{0(8)} \neq L^{4*}_{2(8)} \oplus R^{4*}_{2(8)} \oplus R^{4*}_{0(8)}, \tag{9}$$

$$L^4_{3(8)} \oplus R^4_{3(8)} \oplus R^4_{1(8)} \neq L^{4*}_{3(8)} \oplus R^{4*}_{3(8)} \oplus R^{4*}_{1(8)} \tag{10}$$

$$L^4_{0(8)} \oplus R^4_{2(8)} \neq L^{4*}_{0(8)} \oplus R^{4*}_{2(8)} \tag{11}$$

Corollary 2. *Let* $Pj_{(64)} = Lj^1_{(32)}||Rj^1_{(32)} (0 \leq j \leq 255)$ *be 256 plaintexts of 3-round FOX64,* $Cj_{(64)} = Lj^4_{(32)}||Rj^4_{(32)}$ *be the corresponding ciphertexts, If* $Lj^1_{(32)} = Rj^1_{(32)}$*,* $Lj_{i(8)} (i = 0, 1, 2)$ *are constants, and* $Lj_{3(8)}$ *take all possible values between 0 and 255, then* $Cj_{(64)} (0 \leq j \leq 255)$ *satisfy:*

$$\bigoplus_{j=0}^{255} (Lj^4_{1(8)} \oplus Rj^4_{3(8)}) = 0 \tag{12}$$

Using **Theorem 2** and **Corollary 2**, we can construct the Algorithms similar to those in Section 4, and get four subkeys of 4-round FOX64. The attack requires

less than 2^9 chosen plaintexts, and the time complexity is about $2^{45.4}$ 4-round FOX64 encryptions.

Similarly, we can get subkeys of 5(6, 7)-round FOX64 just through guessing the overall key bits behind the fourth round, then using the attack procedure for 4-round FOX64. The time complexity on 5,6 and 7-round FOX64 is about $2^{109.4}$, $2^{173.4}$ and $2^{237.4}$ respectively.

6 Concluding remarks

Since FOX is a new cipher published last year, all we know about its security analysis are limited to be the designer's results[1] and Ref.[6]. In this paper, we combine collision technique and integral attack to analyze the security of FOX. The improved integral attack on FOX is more efficient than known integral attacks. The complexity of improved integral attack is $2^{77.6}$ on 4-round FOX128, $2^{205.6}$ against 5-round FOX128 respectively. For FOX64, the complexity of improved integral attack is about $2^{45.4}$ on 4-round FOX64, $2^{109.4}$ against 5-round FOX64, $2^{173.4}$ against 6-round FOX64, $2^{237.4}$ against 7-round FOX64 respectively. Our results also show that 4-round FOX64/64, 5-round FOX64/128, 6-round FOX64/192 ,7-round FOX64/256 and 5-round FOX128/256 are not immune to improved integral attack in this paper.

There are some mistakes about the integral cryptanalysis in Ref.[6], so we only compare the performance of known integral attacks on FOX in Ref.[1] and that of this paper in the following table.

Name	round	Time	Notes
FOX64	4	2^{72}	Ref.[1]
FOX64	4	$2^{45.4}$	this paper
FOX64	5	2^{136}	Ref.[1]
FOX64	5	$2^{109.4}$	this paper
FOX64	6	2^{200}	Ref.[1]
FOX64	6	$2^{173.4}$	this paper
FOX64	7	$2^{237.4}$	this paper
FOX128	4	2^{136}	Ref.[1]
FOX128	4	$2^{77.6}$	this paper
FOX128	5	$2^{205.6}$	this paper

References

1. P.Junod and S. Vaudenay, "FOX: a new Family of Block Ciphers," *Selected Areas in Cryptography-SAC 2004*,LNCS 2595, pp.131-146,Springer-Verlag.
   FOX Specifications Version 1.2 appeared on http://crypto.junod.info
2. Mediacrypt AG. http://www.mediacrypt.com
3. X.Lai and J. Massey, "A proposal for a new block encryption standard," *Advances in Cryptology-EUROCRYPT'90*, LNCS 473,pp.389-404,Springer-Verlag.
4. S.Vaudenay, "On the Lai-Massey scheme," *Advances in Cryptology-ASIACRYPT'99*, LNCS 1716, pp.8-19,Springer-Verlag.
5. P.Junod and S. Vaudenay,"Perfect diffusion primitives for block ciphers—building efficient MDS matrices", *Selected Areas in Cryptography-SAC 2004*,LNCS 2595, pp.131-146,Springer-Verlag.

6. Marine Minier, "An integral cryptanalysis against a five rounds version of FOX", *Western European Workshop on Research in Crpytography - WEWoRC,* July 05-07, Leuven, Belgium, 2005.
7. K.Lanford and E.Hellman,"Differential-linear cryptanalysis",*Advances in Cryptology-CRYPTO'94,*LNCS 893, pp.17-25,Springer-Verlag.
8. E.Biham, A.Biryukov, and A.Shamir, "Enhancing differential-linaer cryptanalysis," *Advances in Cryptology-ASIACRYPT'02,* LNCS 2501, pp.254-266,Springer-Verlag,
9. D.Wagner, "The boomerang attack",*Fast Software Encryption-FSE'99*, LNCS 1636,pp.157-170,Springer-Verlag.
10. E.Biham,O.Dunkelman,and N.Keller,"The rectangle attack-rectangling the Serpent", *Advances in Cryptology-EUROCRYPT'01,* LNCS 2045,pp.340-357,Springer-Verlag.
11. L.Knudsen, "Truncated and higher order differentials," *Fast Software Encryption-FSE'95*, LNCS 2595,pp.196-211,Springer-Verlag.
12. E.Biham, A.Biryukov, and A.Shamir,"Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials", *Advances in Cryptology-EUROCRYPT'99,*LNCS 2595, pp.12-23,Springer-Verlag.
13. C.Harpes and J.Massey, "Partitioning cryptanalysis",*Fast Software Encryption-FSE'97*, LNCS 1267,pp.13-27,Springer-Verlag.
14. T.Jakobsen and L.Knudsen,"The interpolation attack against block ciphers", *Fast Software Encryption-FSE'99*, LNCS 1267,pp.28-40,Springer-Verlag.
15. N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations", *Advances in Cryptology-ASIACRYPT'02,* LNCS 2595, pp.267-287,Springer-Verlag.
16. S.Murphy and M.Robshaw,"Comments on the security of the AES and the XSL technique",*Electronic Letters,*39(1):36-38.2003.
17. A. Biryukov and D.Wagner, "Slide attacks," *Fast Software Encryption-FSE'99*, LNCS 1636, pp.245-259,Springer-Verlag.
18. A. Biryukov and D.Wagner, " Advanced slide attacks," *Advances in Cryptology-EUROCRYPT'00,* LNCS 1807,pp.589-606,Springer-Verlag.
19. H.Wu,"Related-cipher attacks", *Information and Communications Security,ICICS 2002,*LNCS 2513, pp.447-455,Springer-Verlag.
20. L. Knudsen and D. wagner, "Integral cryptanalysis(extended abstract)" *Fast Software Encryption-FSE2002,*LNCS 2595, pp.112-127,Springer-Verlag.
21. Y.Yeom, S.Park, and I. Kim, "On the security of Camellia against the square attack," *Fast Software Encryption-FSE'02,*LNCS 2356, Springer-Verlag 2002,pp.89-99.
22. Y.Yeom, I. Park, and I. Kim, "A study of Integral type cryptanalysis on Camellia," *The 2003 Symposium on Cryptography and Information Security-SCIS'03.*