Higher Order Differential Attack of a CAST Cipher

Shiho Moriai[1], Takeshi Shimoyama[1], and Toshinobu Kaneko[1,2]

[1] TAO (Telecommunications Advancement Organization of Japan)
{shiho,shimo}@yokohama.tao.or.jp
[2] Science University of Tokyo
kaneko@ee.noda.sut.ac.jp

Abstract. This paper proposes a new higher order differential attack. The higher order differential attack proposed at FSE'97 by Jakobsen and Knudsen used exhaustive search for recovering the last round key. Our new attack improves the complexity to the cost of solving a linear system of equations. As an example we show the higher order differential attack of a CAST cipher with 5 rounds. The required number of chosen plaintexts is 2^{17} and the required complexity is less than 2^{25} times the computation of the round function. Our experimental results show that the last round key of the CAST cipher with 5 rounds can be recovered in less than 15 seconds on an UltraSPARC station.

1 Introduction

Higher order differential attack is one of the powerful algebraic cryptanalyses. It is useful for attacking ciphers which can be represented as Boolean polynomials with low degrees. After Lai mentioned cryptographic significance of derivatives of Boolean functions in [12], Knudsen used this notion to attack ciphers which were secure against conventional differential attacks[11]. At FSE'97 Jakobsen and Knudsen[7] gave an extension of Knudsen's attacks and broke the cipher with quadratic functions such as the cipher $\mathcal{KN}$[14] and the scheme by Kiefer[10]. These were provably secure ciphers against differential and linear cryptanalysis. Furthermore, at ISW'97, Shimoyama, Moriai, and Kaneko[18] essentially reduced the complexity and the number of chosen plaintexts required for the higher order differential attack of the cipher $\mathcal{KN}$. In this paper we generalize the higher order differential attack described in [18] and apply it to CAST ciphers.

CAST ciphers are a family of symmetric ciphers constructed using the CAST design procedure[1] proposed by Adams and Tavares. The CAST design procedure describes that they appear to have good resistance to differential cryptanalysis[5], linear cryptanalysis[15], and related-key cryptanalysis[4]. A known attack on CAST ciphers is the attack which uses weaknesses of non-surjective round functions and it requires 2^{32} known texts for a CAST cipher with 6 rounds[16].

In this paper we demonstrate that some of symmetric ciphers constructed using the CAST design procedure can be broken by our higher order differential

S. Vaudenay (Ed.): Fast Software Encryption – FSE'98, LNCS 1372, pp. 17–31, 1998.

attack, if the number of rounds is small. CAST-128 is a famous example CAST cipher used in several commercial applications, but this is not our target. CAST-128 seems resistant to our attack.

CAST ciphers use the Feistel structure used in DES. The CAST design procedure allows a wide variety of round functions. It has substitution boxes (S-boxes) with fewer input bits than output bits (e.g. 8×32). There are several proposals for S-boxes. For example, [3] suggested constructing the S-boxes from bent functions. Later on [6] CAST ciphers with random S-boxes were proposed. In our attack, we use the S-boxes proposed for CAST-128[1,2] based on bent functions. As for operations used for combining input and subkey or output results of S-boxes, the CAST design procedure describes that a simple way is to specify that all operations are XORs. Although other operations (addition and subtraction modulo 2^{32}, multiplication modulo $(2^{32} \pm 1)$, etc.) may be used instead, we assume that the CAST cipher of our target uses XORs for all operations.

We explain the higher order differential attack of this CAST cipher with 5 rounds. We begin by finding the Boolean polynomials of all output bits of S-boxes. The polynomials show that all degrees are 4. When all operations in the round function are XORs, the degree of the round function is at most 4. If the right half of plaintext is fixed at any value, the degree of the right half of the 4-th round is at most 16, and the 16-th order differential becomes constant. Thus we can construct the attack equations for recovering the last (i.e. 5-th) round key.

In [7], exhaustive search was used for finding the true key. If their attack were applied to this CAST cipher with 5 rounds, the required complexity would be 2^{48} times the computation of the round function using 2^{17} chosen plaintexts. Our new attack can recover the last round key by solving the linear system of equations. As a result, the required number of chosen plaintexts is 2^{17} and the required complexity is reduced to less than 2^{25} times the computation of the round function. Our experimental results show that all last round key bits of the CAST cipher can be recovered in less than 15 seconds on a Sun Ultra 2 workstation (UltraSPARC, 200MHz).

2 Higher Order Differential Attack

2.1 Preliminaries

Definition 1. *Let $GF(2)^n$ be the n-dimensional vector space over $GF(2)$. We denote addition on $GF(2)^n$ by $+$. We define $V^{(l)}[a_l, \ldots, a_1]$ as the l-dimensional subspace of $GF(2)^n$ which is the set of all 2^l possible linear combinations of $a_l, \ldots, a_1$, where each a_i is in $GF(2)^n$ and linearly independent. We often use $V^{(l)}$ for $V^{(l)}[a_l, \ldots, a_1]$ when $\{a_l, \ldots, a_1\}$ is understood.*

Definition 2. *Let $GF(2)[X]$ be the polynomial ring of $X = \{x_{n-1}, \ldots, x_0\}$ over $GF(2)$. Let Id be the ideal of $GF(2)[X]$ generated by $x_{n-1}^2 + x_{n-1}, \ldots, x_0^2 + x_0 \in GF(2)[X]$. We define $R[X]$ as the quotient ring of $GF(2)[X]$ modulo Id, i.e.*

$GF(2)[X]/Id$. We call $R[X]$ the Boolean polynomial ring of X, and call each element of it a Boolean polynomial. Since an element in $R[X]$ is regarded as a function: $GF(2)^n \to GF(2)$, it is also called a Boolean polynomial function or a Boolean function.

Definition 3. *Let X and K be sets of variables. We define $R[X, K]$ as a Boolean polynomial ring of $X \cup K$, i.e. $R[X \cup K]$. For element $f(X, K) \in R[X, K]$, we define $\deg_X(f)$ as the total degree of f with respective to X whose coefficients are in $R[K]$.*

Definition 4. *We call a vector of n Boolean functions a vector Boolean function. For example $f = (f_{n-1}, \ldots, f_0)$ is a vector Boolean function where f_i is a Boolean function. Each element of a vector Boolean function is called a coordinate Boolean function.*

For an iterated block cipher with block size $2n$ bits and key size s bits, we denote a plaintext by $x = (x_{2n-1}, \ldots, x_0) \in GF(2)^{2n}$, a key by $k = (k_{s-1}, \ldots, k_0) \in GF(2)^s$, and a ciphertext by $y = (y_{2n-1}, \ldots, y_0) \in GF(2)^{2n}$. The ciphertext y is represented by a vector Boolean function $y = G(x, k) = (g_{2n-1}(x, k), \ldots, g_0(x, k)) \in R[X, K]^{2n}$, where X and K are sets of variables: $X = \{x_{2n-1}, \ldots, x_0\}$, $K = \{k_{s-1}, \ldots, k_0\}$. A coordinate Boolean function of $G(x, k)$ is a Boolean function $g[k](x)$ on X when k is fixed. In general $g[k](x)$ is represented as follows,

$$g[k](x) = \sum c_{i_{2n-1},\ldots,i_0}(k) \cdot x_{2n-1}^{i_{2n-1}} \cdots x_0^{i_0},$$

where i_* is 0 or 1.

Definition 5. *We define the i-th order differential of $g[k](x)$ with respect to X, denoted by $\Delta^{(i)}_{(a_i,\ldots,a_1)} g[k](x)$, as follows,*

$$\begin{aligned} \Delta^{(1)}_{(a)} g[k](x) &= g[k](x) + g[k](x + a) \\ \Delta^{(i)}_{(a_i,\ldots,a_1)} g[k](x) &= \Delta^{(1)}_{(a_i)}(\Delta^{(i-1)}_{(a_{i-1},\ldots,a_1)} g[k](x)) \end{aligned}$$

where $\{a_i, \ldots, a_1\} \subseteq GF(2)^{2n}$ are linearly independent and any $a \in GF(2)^{2n}$. In this paper, since we consider only the higher order differential with respect to X, we omit "with respect to X."

Definition 6. *We define the i-th order differential of vector Boolean function $G = (g_{2n-1}, \ldots, g_0)$ as follows,*

$$\Delta^{(i)}_{(a_i,\ldots,a_1)} G = (\Delta^{(i)}_{(a_i,\ldots,a_1)} g_{2n-1}, \ldots, \Delta^{(i)}_{(a_i,\ldots,a_1)} g_0).$$

The following propositions are known on the higher order differential of Boolean functions.

Proposition 7. [12] *The following equation holds for any* $a \in GF(2)^{2n}$ *and* $\{a_i, \ldots, a_1\} \subseteq GF(2)^{2n}$.

$$\Delta^{(i)}_{(a_i,\ldots,a_1)} g[k](a) = \sum_{x \in V^{(i)}[a_i,\ldots,a_1]} g[k](x+a)$$

Proposition 8. [18] *Let* $\{a_{d+1}, \ldots, a_1\} \subseteq GF(2)^{2n}$ *be linearly independent. If* $\deg_X(g[k](x)) = d$, *then we have the following equations.*

$$\Delta^{(d)}_{(a_d,\ldots,a_1)} g[k](x) \in R[K],$$
$$\Delta^{(d+1)}_{(a_{d+1},\ldots,a_1)} g[k](x) = 0$$

2.2 Attack Procedure

The following is the attack procedure of our higher order differential attack of an iterated block cipher with block size $2n$ and R rounds. Let $k^{(i)}$ be the i-th round subkey, and each subkey be m bits, i.e. $k^{(i)} = (k^{(i)}_{m-1}, \ldots, k^{(i)}_0)$. Let $K^{(i)}$ be a set of variables of $k^{(i)}$, i.e. $K^{(i)} = \{k^{(i)}_{m-1}, \ldots, k^{(i)}_0\}$.

Here we describe "$(R-1)$-round attack", where we find a certain constant value which is independent of the key (e.g. the higher order differential of the output of the $(R-1)$-th round), and construct the attack equations for recovering the last round key. Of course a "$(R-2)$-round attack" is possible though solving of the attack equations becomes rather difficult.

We assume that the attacker has or can compute all chosen plaintexts in $V^{(d)}[a_d, \ldots, a_1]+a$ and the corresponding ciphertexts, where d is the total degree of the output of the $(R-1)$-th round, and a is any value in $GF(2)^{2n}$. For some block ciphers, the total degree of the output of the $(R-1)$-th round may be difficult with choices of $\{a_d, \ldots, a_1\}$ and a. However we don't consider it in this paper.

Even if the algorithm of the cipher is not open (i.e. if it is a blackbox), our attack is applicable when we know the total degree d of the output of the $(R-1)$-th round by some ways. In this case, we start from step 2.

1. Find the degree of round function. In attacking iterated ciphers by higher order differential attacks, it is useful to represent the round function by Boolean polynomials. We can get the degree over $GF(2)$ of each output bit of the round function from these polynomials. The information on which terms are included in the polynomials is also helpful in step 3.

We begin by representing S-boxes by Boolean polynomial functions. When the description of the S-boxes is not given as some algebraic expressions, we construct Boolean polynomial functions from the description tables (see Section4.1).

2. Compute the higher order differential of output of the $(R-1)$-th round. Our higher order differential attack is possible, for an integer $1 \leq d \leq 2n$, when the d-th order differential of the output of the $(R-1)$-th round is a certain constant value which is independent of the key. When is this condition true? One is when the degree of the output of the $(R-1)$-th round is $d-1$. Another is when the input and subkeys are combined with XORs simply, and the degree of the output of the $(R-1)$-th round is d. In this case, the total degree of the output of the $(R-1)$-th round with respect to X is equal to the total degree with respect to X and $K^{(i)}$ $(i = 1, \cdots, R-1)$ before the degree reaches $2n$ (see [18, Proposition 1]).

In these cases, the d-th order differential of the output of the $(R-1)$-th round can be computed by using Proposition 1 without knowing the true key.

3. Construct attack equations for recovering the last round key. We give the details in the case of a Feistel cipher. Let $x = (x_L, x_R)$ and $y = (y_L(x), y_R(x))$, where x_L denotes the left half of plaintext, x_R denotes the right half, y_L denotes the Boolean polynomial function of left half of ciphertext, and y_R denotes the vector Boolean polynomial function of right half. Let $\tilde{y}_R(x)$ be the vector Boolean polynomial function of the right half of the output of the $(R-1)$-th round. Then we have

$$F[k^{(R)}](y_R(x)) + y_L(x) = \tilde{y}_R(x).$$

If the d-th order differential of $\tilde{y}_R(x)$ is constant, we have the following equation for linearly independent $\{a_i, \ldots, a_1\} \subseteq GF(2)^{2n}$ and any $a \in GF(2)^{2n}$.

$$\Delta^{(d)}_{(a_d,\ldots,a_1)} F[k^{(R)}](y_R(a)) + \Delta^{(d)}_{(a_d,\ldots,a_1)} y_L(a) = \Delta^{(d)}_{(a_d,\ldots,a_1)} \tilde{y}_R(a) \quad (= const.)$$

If we have all plaintexts in $V^{(d)}[a_d, \ldots, a_1] + a$ and corresponding ciphertexts, we obtain the following equation by computing each term using Proposition 1.

$$\sum_{x \in V^{(d)}[a_d,\ldots,a_1]+a} \Big(F[k^{(R)}](y_R(x)) + y_L(x) \Big) = \sum_{x \in V^{(d)}[a_d,\ldots,a_1]+a} \tilde{y}_R(x) \qquad (1)$$

If the total degree of F is D (≥ 1), equation (1) has degree $D-1$ with respect to $k^{(R)}$. This is because we can rewrite the first term of equation (1) as follows. (The first order differential of a function of degree D has degree $D-1$.)

$$\begin{aligned}
&\sum_{x \in V^{(d)}+a} F[k^{(R)}](y_R(x)) \\
&= \sum_{x \in V^{(d)}+a\setminus\{a\}} F[k^{(R)}](y_R(x)) + F[k^{(R)}](y_R(a)) \\
&= \sum_{x \in V^{(d)}+a\setminus\{a\}} \Big(F[k^{(R)}](y_R(x)) + F[k^{(R)}](y_R(a)) \Big) \\
&= \sum_{x \in V^{(d)}+a\setminus\{a\}} \Delta_{y'} F[k^{(R)}](y_R(x)) \quad (\, y' = y_R(x) + y_R(a)\,)
\end{aligned}$$

Since F is a vector Boolean function composed of n coordinate Boolean functions, equation (1) forms the system of algebraic equations of degree $D-1$ with m unknowns. (Note that m is the number of bits of the last round key $k^{(R)}$.) We have some ways to solve the system of algebraic equations, and in this paper we take a similar way as one described in [8]. That is, we transform it to the system of linear equations where we regard all monomials on $k^{(R)}$ in equation (1) as independent unknown variables. Hereafter M denotes the number of the unknown variables. When $D = 2$, the unknown variables are $\{k_{m-1}^{(R)}, \ldots, k_0^{(R)}\}$ and $M = m$. When $D = 3$, the unknown variables are $\{k_{m-1}^{(R)}, \ldots, k_0^{(R)}, k_{m-1}^{(R)}k_{m-2}^{(R)}, \ldots, k_0^{(R)}k_1^{(R)}\}$ and $M = m + {}_mC_2$. Similarly, when $D = 4$, $M = m + {}_mC_2 + {}_mC_3$. When the total degree of F is D, M is at most $\sum_{i=1}^{D-1} {}_mC_i$. Actually, M is much smaller than this upper bound because coefficients of some of the unknown variables can cancel each other out, or because some of these unknown variables don't exist for some F. Finding a small M is important for reducing the complexity. General theory on a tighter upper bound of M will appear in another paper.

If the number of unknown variables of the linear equations ($= M$) is larger than n, we have to set up equations (1) using plaintexts in different d-dimensional spaces $V^{(d)}[a'_d, \ldots, a'_1] + a'$ to determine M unknowns. However, this does not increase the required number of chosen plaintexts by $\lceil \frac{M}{n} \rceil$ times because some plaintexts can be used repeatedly. That is, for an integer $\delta > d$, we can obtain ${}_\delta C_d$ different d-dimensional vector spaces from δ-dimensional vector space. Therefore, if we let δ_{min} be the smallest δ s.t. $\lceil \frac{M}{n} \rceil \leq {}_\delta C_d$, then the required number of the chosen plaintexts is at most $2^{\delta_{min}}$.

2.3 Comparison with Jakobsen-Knudsen[7]

In this section we compare the complexity of our higher order different attack with Jakobsen and Knudsen's attack[7]. The dominant complexity is setting up the system of linear equations, i.e. computing the coefficients (see also Section 4.3). For the second and third terms of equation (1), $R \times 2^{\delta min}$ times the computation of the round function is needed. For the first term of equation (1), at most $(M+1) \times 2^{\delta min}$ times the computation of the round function[1] is required. Therefore, the required complexity is at most $(M + R + 1) \times 2^{\delta_{min}}$. On the other hand, the required complexity for Jakobsen and Knudsen's attack[7] was 2^{m+d}[7, Theorem 1]. Since $d \approx \delta_{min}$ and $M + R + 1 \ll 2^m$, the complexity is reduced.

3 CAST

The family of the ciphers constructed using the CAST design procedure[1] are known as CAST ciphers, and [1] describes that they appear to have good resistance to differential cryptanalysis[5], linear cryptanalysis[15], and related-key cryptanalysis[4].

[1] There is another way of computing the coefficients of the terms of degree $D-1$ with less complexity. See Appendix B.

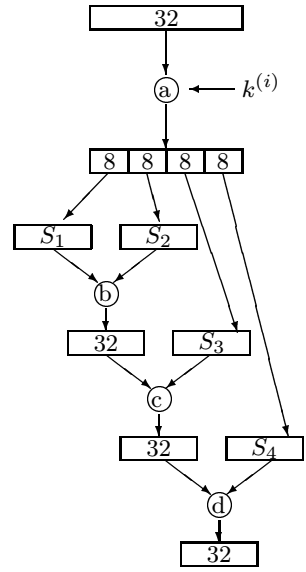

Fig. 1. CAST round function

CAST ciphers are based on the framework of the Feistel cipher. The round function is specified as follows (see also Fig.1.). A 32-bit data half is input to the function along with a subkey $k^{(i)}$. These two quantities are combined using operation "a" and the 32-bit result is split into four 8-bit pieces. Each piece is input to a different 8×32 S-box (S_1, S_2, S_3, and S_4). S-boxes S_1 and S_2 are combined using operation "b"; the result is combined with S_3 using operation "c"; this second result is combined with S_4 using operation "d". The final 32-bit result is the output of the round function.

The CAST design procedure allows a wide variety of possible round functions: 4 S-boxes and 4 operations (a,b,c, and d). As for S-boxes, [3] suggested constructing the S-boxes from bent functions. Later on [6] CAST with random S-boxes was proposed. In our attack, we use the S-boxes based on bent functions proposed for CAST-128. As for operations, a simple way to define the round function is to specify that all operations are XORs, which is addition on $GF(2)$, although other operations may be used instead. Actually, according to [1], some care in the choice of operation "a" can conceivably give intrinsic immunity to differential and linear cryptanalysis. The immunity to higher order differential for choices of operations (a,b,c, and d) will be discussed in Section 5.

As for the number of rounds, it seems that the CAST design procedure doesn't specify a concrete number. However, in [1] it is described that CAST ciphers possess a number of improvements compared to DES in both the round function and the key schedule which provide good cryptographic properties in

fewer rounds[2] than DES. There are also several key schedules for CAST ciphers, but for the purpose of our attack the key schedule makes no difference.

4 Higher Order Differential Attack of a CAST Cipher

4.1 Boolean Polynomials of S-boxes

We begin by representing S-boxes by Boolean polynomial functions. We use the S-boxes proposed for CAST-128. The description of the S-boxes is given by tables. One way to construct them can be seen in [19]. Another more efficient method using a matrix transformation is also known. The obtained Boolean polynomials of S-boxes occupy a lot of space, and we show those of only some bits of S_1 in Appendix A.

From the obtained Boolean polynomials, it is confirmed that all the degrees of all output bits of all S-boxes are 4, which doesn't contradict the property of bent functions: the degree of a bent function: $GF(2)^{2n} \to GF(2)$ is at most n. When the operations a,b,c, and d are XORs, all the degrees of all output bits of the round function are at most 4. We discuss the higher order differential attack of this CAST cipher with 5 rounds.

4.2 Linear Equations for Recovering the Last Round Key

If the right half of plaintext is fixed at any value, the degree of the right half of the 4-th round, $\tilde{y}_R(x_L)$, is at most 16, and the 16-th order differential of $\tilde{y}_R(x_L)$ becomes constant. Therefore, we can compute it without knowing the true key, and we have the following attack equations for recovering the last round key, $k^{(5)}$.

$$\sum_{x_L \in V^{(16)}+a} F[k^{(5)}](y_R(x_L)) + \sum_{x_L \in V^{(16)}+a} y_L(x_L) = \sum_{x_L \in V^{(16)}+a} \tilde{y}_R(x_L), \qquad (2)$$

where $y_R, y_L, \tilde{y}_R : GF(2)^{32} \to GF(2)^{32}$ and $a \in GF(2)^{32}$.

As we described in Section 2.2, since the total degree of F is 4, equation (2) has degree 3 with respect to $\{k_{31}^{(5)}, k_{30}^{(5)}, \ldots, k_0^{(5)}\}$. It follows that equation (2) forms a system of equations of degree 3 with 32 unknowns. Hereafter, we write $\{k_{31}, k_{30}, \ldots, k_0\}$ for $\{k_{31}^{(5)}, k_{30}^{(5)}, \ldots, k_0^{(5)}\}$ for simplicity.

Here we transform the system of equations of degree 3 to a system of linear equations with M unknowns. For decreasing the complexity, it is important to find as small M as possible. In this paper we find a small M by considering the structure of the round function of CAST ciphers. The output of round function F is the sum (XOR) of the outputs of S_1, S_2, S_3, and S_4, whose sets of input variables are disjoint: i.e. the set of input variables of S_1 is $\{k_{31}, k_{30}, \ldots, k_{24}\}$, that of S_2 is $\{k_{23}, k_{22}, \ldots, k_{16}\}$, that of S_3 is $\{k_{15}, k_{14}, \ldots, k_8\}$, and that of S_4 is

[2] For example, CAST-128 is a 12 or 16 round Feistel cipher[RFC2144].

$\{k_7, k_6, \ldots, k_0\}$. Consequently, all the terms included in equation (2) are products of variables from one of the sets above. Therefore, equation (2) is transformed to the system of linear equations below with the following M unknown variables, where $M = 32 + (4 \times {}_8C_2) + (4 \times {}_8C_3) = 368$.

$$\underbrace{\{k_0, k_1, \ldots, k_{31}}_{\text{degree-1 } (32)}, \underbrace{k_0k_1, k_0k_2, \ldots, k_{30}k_{31}}_{\text{degree-2 } (4\times{}_8C_2)}, \underbrace{k_0k_1k_2, k_0k_1k_3, \ldots, k_{29}k_{30}k_{31}\}}_{\text{degree-3 } (4\times{}_8C_3)}$$

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,M-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{31,0} & a_{31,1} & \cdots & a_{31,M-1} \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{31} \\ \hline k_0k_1 \\ k_0k_2 \\ \vdots \\ k_{30}k_{31} \\ \hline k_0k_1k_2 \\ k_0k_1k_3 \\ \vdots \\ k_{29}k_{30}k_{31} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{31} \end{pmatrix}$$

We need M equations to determine the M unknown variables. However, since F, y_L, and $\tilde{y}_R$ are vector functions composed of $n\,(= 32)$ functions, only n equations are obtained from equation (2). Therefore, we have to compute equation (2) for $\lceil M/n \rceil\,(= \lceil 368/32 \rceil = 12)$ different $V^{(16)}[a'_{16}, \ldots, a'_1]$. This does not increase the required number of chosen plaintexts by as many as $\lceil M/n \rceil\,(= 12)$. Because we can take ${}_{17}C_{16}\,(= 17)$ different $V^{(16)}$ from $V^{(17)}[a_{17}, \ldots, a_1]$, it only doubles the required number of chosen plaintexts.

In order to set up the system of linear equations above, we compute $M' \times M$ coefficient matrix described below, where $M' > M$. We prepare $M' \times M$ coefficient matrix because $M \times M$ matrix is not always normal. Our experimental results show that $M' = 32 \times 12$ is enough to determine the key.

How to compute coefficients $a_{i,j}$ and $b_i \in GF(2)$ in the matrices is as follows. Here we describe the computation of only the coefficients of upper 32 rows. The remaining coefficients can be computed using 11 different $V^{(16)}[a'_{16}, \ldots, a'_1]$ in the same way.

$$\begin{pmatrix} a_{0,0} & a_{0,1} & \cdots\cdots\cdots & a_{0,M-1} \\ a_{1,0} & a_{1,1} & \cdots\cdots\cdots & a_{1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{31,0} & a_{31,1} & \cdots\cdots\cdots & a_{31,M-1} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots\cdots\cdots & a_{M-1,M-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M'-1,0} & a_{M'-1,1} & \cdots\cdots\cdots & a_{M'-1,M-1} \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{31} \\ \hline k_0k_1 \\ k_0k_2 \\ \vdots \\ k_{30}k_{31} \\ \hline k_0k_1k_2 \\ k_0k_1k_3 \\ \vdots \\ k_{29}k_{30}k_{31} \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{31} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ b_{M-1} \\ \vdots \\ b_{M'-1} \end{pmatrix}$$

All coefficients $a_{i,j}$ and b_i can be computed by using

$$\mathcal{F}_j = \sum_{x_L \in V^{(16)}+a} F[e_j](y_R(x_L)) \tag{3}$$

where e_j $(0 \le j \le M)$ is as follows:

$$\begin{aligned} e_j &= \bar{e}_{i_1} && (0 \le j < 32) \\ e_j &= \bar{e}_{i_1} + \bar{e}_{i_2} && (32 \le j < 144) \\ e_j &= \bar{e}_{i_1} + \bar{e}_{i_2} + \bar{e}_{i_3} && (144 \le j < M) \\ e_j &= (0, \ldots, 0) \in GF(2)^{32} && (j = M), \end{aligned}$$

where $\bar{e}_i = (0, \ldots, \overset{i}{\check{1}}, \ldots, 0) \in GF(2)^{32}$, and $0 \le i_1 < i_2 < i_3 \le 31$.

Let $B = {}^t(b_0, b_1, \ldots, b_{31})$. B is computed as follows:

$$B = \mathcal{F}_M + \sum_{x_L \in V^{(16)}+a} y_L(x_L) + \sum_{x_L \in V^{(16)}+a} \tilde{y}_R(x_L). \tag{4}$$

Let $A_j = {}^t(a_{0,j}, a_{1,j}, \ldots, a_{31,j})$ $(0 \le j < M)$. Elements of A_j are coefficients of the unknown variable located at the j-th row. When $0 \le j < 32$, A_j is a column vector of coefficients of k_j. Therefore, A_j is computed as follows:

$$A_j = \mathcal{F}_j + \mathcal{F}_M.$$

When A_j is a column vector of coefficients of $k_{i_1} \cdot k_{i_2}$, i.e. when $32 \le j < 144$, A_j is computed as follows:

$$A_j = \mathcal{F}_j + \mathcal{F}_{i_1} + \mathcal{F}_{i_2} + \mathcal{F}_M.$$

When A_j is a column vector of coefficients of $k_{i_1} \cdot k_{i_2} \cdot k_{i_3}$, i.e. when $144 \le j < M$, A_j can be computed similarly. We have another method with less complexity in Appendix B.

4.3 Complexity

This section discusses the required complexity for our higher order differential attack. Most of the execution time is spend in the following procedures.

- computing ciphertexts & higher order differentials
- computing all coefficients in the system of linear equations
- solving the linear equations

Computing ciphertexts & higher order differential
In order to compute equation (4), we have to prepare 12 sums of 2^{16} ciphertexts (=output of 5-th round) and output of the 4-th round. This can be done with 2^{17} ciphertexts and output of the 4-th round as explained in the previous section. Therefore, the required complexity is 5×2^{17} times the computation of the round function. Note that we assume that working out the sum (i.e. XOR) is negligible compared with the computation of the round function.

Computing all coefficients in the system of linear equations
All coefficients in the system of linear equations can be computed by computing equation (3) for e_j $(0 \leq j \leq 368)$. Therefore, the required complexity is $(368 + 1) \times 2^{17}$ times the computation of the round function. This is the dominant part of the higher order differential attack. Since in [7] Jakobsen and Knudsen described that the average complexity was $2^{31} \times 2^{17}$, our attack has achieved speedup by 2^{23} times.

Solving the linear equations
We used Gauss-Jordan's elimination method for solving the linear equations. The size of matrix is $M' \times M$, where $M' = 384$ and $M = 368$. The required complexity is negligible compared with the computations above.

Consequently, the total complexity is $(5 + 368 + 1) \times 2^{17} < 2^{26}$ times the computation of the round function. The way to reduce the complexity by half is in Appendix B.

4.4 Experimental Results

Our experimental results showed that the all last round key bits of the CAST cipher with 5 rounds could be recovered in 13.79 seconds (average time of 100 trials) on a SunUltra2 workstation (UltraSPARC, 200MHz). Table 1 shows an execution profile of the program produced by `gprof`, which is a GNU command to display call-graph profile data.

5 Discussion

In this Section, the immunity to higher order differential attack for choices of S-boxes and operations (a,b,c, and d) is discussed.

Section 4 showed that a CAST cipher with 5 rounds which uses S-boxes proposed for CAST-128 and XORs for all operations (a,b,c, and d) can be broken

procedures	CPU time	ratio
computing ciphertexts & higher order differentials	0.83 sec.	6.0 %
computing all coefficients in the linear equations	12.92 sec.	93.7 %
solving the linear equations	0.04 sec.	0.3 %
total	13.79 sec.	100%

Table 1. Execution profile of the program

by our higher order differential attack. Since the degrees of S-boxes for CAST-128 are 4, the CAST cipher can be broken up to only 5-round. However, if the degree of the round function is lower, the CAST cipher could be broken up to more number of rounds. On [6] CAST ciphers with random S-boxes are proposed, and we must be careful of the degrees of the S-boxes in such cases. Note that it is shown that when randomly generated S-boxes are used, the resulting cipher is resistant to both differential and linear attack in [13].

Let's discuss for other choices of operations (a,b,c, and d). Some modifications of operation "a" are proposed in [1]. One example is the insert of key-dependent rotation, which is used in CAST-128, i.e. $\mathrm{a}(x,k) = \mathrm{a}(x,k_1,k_2) = ((x+k_1) \lll k_2)$, where k_1 is a 32-bit key, k_2 is a 5-bit key, and $\lll$ is the rotation specified by k_2. If only operation "a" is extended to XOR and rotation and "b","c", and "d" are still XOR, the CAST cipher with 5 rounds of our target can be broken by our higher order differential attack, though the complexity increases (rough estimate is 2^{40}).

There are some ways to strengthen CAST-like ciphers against the higher order differential attack. One is the increase of the number of rounds. Another is the mixture of using operations on different groups (e.g. XOR, and addition, (or subtraction) modulo 2^{32}) for "b","c", and "d". This makes the degree higher so sharply that it seems difficult to cryptanalyze by the higher order differential attack at this stage. Actually this idea is used in CAST-128 and Blowfish[17]. Moreover, Blowfish uses key-dependent S-boxes. However, note that these ways are not sufficient conditions to immune to the higher order differential attacks. How to prove the security against higher order differential attacks is open.

Acknowledgments

We would like to thank the referees for many comments. We also thank Serge Vaudenay for essential advice which can improve our attack, Bruce Schneier and Kazumaro Aoki for helpful suggestions for improving the paper.

References

1. C.M.Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure," Designs, Codes and Cryptography, Vol.12, No.3, Nov., pp.283–316, Kluwer Academic Publishers, 1997.

2. C.M.Adams, "The CAST-128 Encryption Algorithm," Request for Comments (RFC) 2144, Network Working Group, Internet Engineering Task Force, May, 1997.
3. C.M.Adams and S.E.Tavares, "Designing S-boxes for ciphers resistant to differential cryptanalysis," In Proceedings of the 3rd symposium on State and Progress of Research in Cryptography, pp.181–190, 1993.
4. E.Biham, "New Types of Cryptanalytic Attacks Using Related Keys," Advances in Cryptology–EUROCRYPT'93, Lecture Notes in Computer Science 765, pp.398–409, Springer-Verlag, 1994.
5. E.Biham and A.Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," Journal of Cryptology, Vol.4, No.1, pp.3–72, Springer-Verlag, 1991.
6. H.M.Heys and S.E.Tavares, "On the security of the CAST encryption algorithm," Canadian Conference on Electrical and Computer Engineering, pp.332–335, 1994.
7. T.Jakobsen and L.R.Knudsen, "The Interpolation Attack on Block Ciphers," In Preproceedings of Fast Software Encryption Workshop'97, pp.28–40, 1997.
8. T.Kaneko, "A known-plaintext attack of FEAL-4 based on the system of linear equations on difference (Extended Abstract) ," Advances in Cryptology–ASIACRYPT'91, Lecture Notes in Computer Science 739, pp.485–488, Springer-Verlag, 1993.
9. T.Kaneko, "A Known Plaintext Cryptanalytic Attack of FEAL-4," (in Japanese), IEICE Trans. Vol.76-A, No.5, May, pp.781–786, 1993.
10. K.Kiefer, "A New Design Concept for Building Secure Block Ciphers," In Proceedings of PRAGOCRYPT'96, pp.30–41, CTU Publishing House, 1996.
11. L.R.Knudsen, "Truncated and Higher Order Differentials," Fast Software Encryption–Second International Workshop, Lecture Note in Computer Science 1008, pp.196–211, Springer-Verlag, 1995.
12. X.Lai, "Higher Order Derivatives and Differential Cryptanalysis," Communications and Cryptography, pp.227–233, Kluwer Academic Publishers, 1994.
13. J.Lee, H.M.Heys, S.E.Tavares, "Resistance of a CAST-Like Encryption Algorithm to Linear and Differential Cryptanalysis," Designs, Codes and Cryptography, Vol.12, No.3, Nov., pp.267–282, Kluwer Academic Publishers, 1997.
14. K.Nyberg and L.R.Knudsen, "Provable Security Against a Differential Attack," Journal of Cryptology, Vol.8, No.1, pp.27–37, Springer-Verlag, 1995.
15. M.Matsui, "Linear Cryptanalysis Method for DES Cipher," Advances in Cryptology–EUROCRYPT'93, Lecture Notes in Computer Science 765, pp.386–397, Springer-Verlag, 1994.
16. V.Rijmen, B.Preneel, and E.De Win "On Weaknesses of Non-surjective Round Functions," Designs, Codes and Cryptography, Vol.12, No.3, Nov., pp.253–266, Kluwer Academic Publishers, 1997.
17. B.Schneier, "Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)," Fast Software Encryption–Cambridge Security Workshop, Lecture Note in Computer Science 809, pp.191–204, Springer-Verlag, 1994.
18. T.Shimoyama, S.Moriai, and T.Kaneko, "Improving the Higher Order Differential Attack and Cryptanalysis of the $\mathcal{KN}$ Cipher," In Pre-Proceedings of 1997 Information Security Workshop, pp.1–8, 1997. (to appear in Lecture Notes in Computer Science, Springer-Verlag)
19. T.Shimoyama, S.Amada, and S.Moriai, "Improved Fast Software Implementation of Block Ciphers (Extended Abstract)," ICICS'97, Beijing, Nov. 1997, Lecture Notes in Computer Science 1334, pp.269–273, Springer-Verlag, 1997.

A Boolean Polynomials of S1 for CAST-128

Due to limitations of space, we show the Boolean polynomials of only 4 bits from the least significant bit of S-box S_1 of CAST-128. Those of all S-boxes of CAST-128 can be downloaded from `http://www.yokohama.tao.or.jp/shiho/`. We used a computer algebra system Risa/Asir to find them.

$$
\begin{aligned}
y_0 = {} & x_4x_3x_2x_1 + x_5x_4x_2x_1 + x_5x_4x_3x_1 + x_6x_3x_2x_0 + x_6x_4x_3x_0 + x_6x_4x_3x_1 \\
& + x_6x_5x_2x_1 + x_6x_5x_3x_0 + x_6x_5x_3x_1 + x_7x_3x_2x_0 + x_7x_3x_2x_1 + x_7x_4x_2x_1 \\
& + x_7x_4x_3x_1 + x_7x_4x_3x_2 + x_7x_5x_2x_0 + x_7x_5x_3x_2 + x_7x_5x_4x_3 + x_7x_6x_2x_1 \\
& + x_7x_6x_4x_1 + x_7x_6x_4x_2 + x_7x_6x_5x_2 + x_7x_6x_5x_3 + x_7x_6x_5x_4 + x_4x_2x_0 + x_4x_2x_1 \\
& + x_5x_2x_1 + x_5x_4x_0 + x_5x_4x_2 + x_5x_4x_3 + x_6x_2x_1 + x_6x_3x_1 + x_6x_4x_0 + x_6x_4x_2 \\
& + x_6x_5x_0 + x_7x_2x_0 + x_7x_2x_1 + x_7x_3x_1 + x_7x_3x_2 + x_7x_5x_0 + x_7x_5x_1 + x_7x_5x_4 \\
& + x_7x_6x_0 + x_7x_6x_1 + x_7x_6x_4 + x_7x_6x_5 + x_1x_0 + x_4x_1 + x_5x_2 + x_5x_3 + x_7x_0 \\
& + x_7x_1 + x_7x_3 + x_7x_4 + x_7x_5 + x_3 + x_4 + x_5 + x_6 + x_7. \\
y_1 = {} & x_5x_3x_2x_0 + x_5x_4x_2x_0 + x_5x_4x_3x_0 + x_5x_4x_3x_2 + x_6x_4x_2x_0 + x_6x_4x_3x_0 \\
& + x_6x_4x_3x_2 + x_6x_5x_2x_0 + x_6x_5x_3x_0 + x_6x_5x_3x_1 + x_6x_5x_4x_0 + x_6x_5x_4x_1 \\
& + x_6x_5x_4x_3 + x_7x_3x_2x_0 + x_7x_3x_2x_1 + x_7x_4x_3x_0 + x_7x_4x_3x_2 + x_7x_5x_3x_1 \\
& + x_7x_5x_4x_0 + x_7x_5x_4x_1 + x_7x_6x_3x_1 + x_7x_6x_4x_2 + x_7x_6x_4x_3 + x_4x_2x_0 + x_4x_2x_1 \\
& + x_4x_3x_2 + x_5x_2x_0 + x_5x_2x_1 + x_5x_4x_1 + x_5x_4x_2 + x_5x_4x_3 + x_6x_2x_0 + x_6x_2x_1 \\
& + x_6x_3x_0 + x_6x_4x_0 + x_6x_4x_1 + x_6x_4x_3 + x_6x_5x_1 + x_6x_5x_3 + x_6x_5x_4 + x_7x_2x_1 \\
& + x_7x_3x_2 + x_7x_5x_1 + x_7x_5x_4 + x_7x_6x_0 + x_7x_6x_3 + x_7x_6x_5 + x_1x_0 + x_2x_1 \\
& + x_3x_1 + x_4x_1 + x_4x_3 + x_5x_1 + x_5x_2 + x_5x_3 + x_6x_1 + x_7x_0 + x_7x_5 + x_7x_6 + x_1 \\
& + x_3 + x_4 + x_5 + x_6 + x_7. \\
y_2 = {} & x_5x_3x_2x_1 + x_5x_4x_2x_1 + x_5x_4x_3x_0 + x_5x_4x_3x_2 + x_6x_3x_2x_0 + x_6x_3x_2x_1 \\
& + x_6x_4x_2x_1 + x_6x_4x_3x_1 + x_6x_4x_3x_2 + x_6x_5x_2x_0 + x_6x_5x_3x_0 + x_6x_5x_3x_1 \\
& + x_6x_5x_4x_2 + x_7x_3x_2x_0 + x_7x_4x_2x_0 + x_7x_4x_2x_1 + x_7x_4x_3x_0 + x_7x_4x_3x_2 \\
& + x_7x_5x_4x_0 + x_7x_5x_4x_1 + x_7x_5x_4x_3 + x_7x_6x_3x_1 + x_7x_6x_4x_1 + x_7x_6x_4x_2 \\
& + x_7x_6x_4x_3 + x_7x_6x_5x_1 + x_7x_6x_5x_2 + x_3x_2x_1 + x_4x_2x_1 + x_4x_3x_1 + x_5x_2x_1 \\
& + x_5x_3x_2 + x_5x_4x_1 + x_5x_4x_2 + x_5x_4x_3 + x_6x_2x_0 + x_6x_2x_1 + x_6x_3x_0 + x_6x_3x_2 \\
& + x_6x_4x_0 + x_6x_4x_1 + x_6x_5x_0 + x_6x_5x_4 + x_7x_2x_1 + x_7x_4x_0 + x_7x_4x_2 + x_7x_4x_3 \\
& + x_7x_5x_0 + x_7x_5x_1 + x_7x_6x_0 + x_7x_6x_1 + x_7x_6x_2 + x_7x_6x_3 + x_7x_6x_4 + x_7x_6x_5 \\
& + x_1x_0 + x_2x_0 + x_2x_1 + x_3x_0 + x_3x_1 + x_3x_2 + x_4x_2 + x_5x_1 + x_5x_2 + x_5x_3 \\
& + x_5x_4 + x_6x_0 + x_6x_1 + x_6x_2 + x_6x_3 + x_6x_4 + x_6x_5 + x_7x_1 + x_7x_4 + x_0 + x_1 \\
& + x_2 + x_7 + 1. \\
y_3 = {} & x_4x_3x_2x_0 + x_4x_3x_2x_1 + x_5x_3x_2x_1 + x_5x_4x_2x_1 + x_5x_4x_3x_1 + x_5x_4x_3x_2 \\
& + x_6x_4x_2x_0 + x_6x_4x_2x_1 + x_6x_4x_3x_1 + x_6x_4x_3x_2 + x_6x_5x_3x_1 + x_6x_5x_3x_2 \\
& + x_6x_5x_4x_1 + x_6x_5x_4x_3 + x_7x_4x_3x_0 + x_7x_4x_3x_1 + x_7x_5x_2x_0 + x_7x_5x_2x_1 \\
& + x_7x_5x_4x_0 + x_7x_5x_4x_1 + x_7x_5x_4x_2 + x_7x_6x_2x_0 + x_7x_6x_3x_0 + x_7x_6x_4x_0 \\
& + x_7x_6x_4x_1 + x_7x_6x_4x_2 + x_7x_6x_5x_0 + x_7x_6x_5x_1 + x_7x_6x_5x_2 + x_7x_6x_5x_4 \\
& + x_4x_3x_1 + x_5x_2x_1 + x_5x_3x_2 + x_5x_4x_0 + x_5x_4x_1 + x_6x_2x_1 + x_6x_3x_0 + x_6x_3x_1 \\
& + x_6x_4x_0 + x_6x_4x_2 + x_6x_4x_3 + x_6x_5x_1 + x_6x_5x_2 + x_6x_5x_3 + x_6x_5x_4 + x_7x_2x_0 \\
& + x_7x_2x_1 + x_7x_3x_0 + x_7x_3x_1 + x_7x_4x_0 + x_7x_4x_2 + x_7x_5x_1 + x_7x_6x_2 + x_7x_6x_4 \\
& + x_7x_6x_5 + x_1x_0 + x_2x_0 + x_2x_1 + x_3x_2 + x_4x_0 + x_4x_1 + x_4x_3 + x_5x_2 + x_5x_4 \\
& + x_6x_1 + x_6x_2 + x_6x_3 + x_6x_4 + x_7x_2 + x_7x_3 + x_7x_4 + x_7x_6 + x_2 + x_3 + x_5 + x_6 \\
& + x_7.
\end{aligned}
$$

B Fast Method for Computing Coefficients of $k_{i_1}k_{i_2}k_{i_3}$

In Section 4.2, it is described that in order to compute all coefficients, the computation of

$$\forall x_L \in V^{(17)} + a, \quad F[e_j](y_R(x_L)) \tag{5}$$

for $(M+1)$ e_j is required. However, there is another method of computing the coefficients of the terms of degree 3, $k_{i_1}k_{i_2}k_{i_3}$ with less complexity. The point is the coefficients of the terms of degree 3, $k_{i_1}k_{i_2}k_{i_3}$ is linear to the input of F. Let

$$\sum_{x_L \in V^{(16)}+a} \mathcal{C}_{i_1 i_2 i_3}(y_R(x_L))k_{i_1}k_{i_2}k_{i_3} \tag{6}$$

be a term of degree 3 in equation (2). The degree of $\mathcal{C}_{i_1 i_2 i_3}$ is 1 with respect to the input of F, since the degree of F is 4. Therefore, we have

$$\mathcal{C}_{i_1 i_2 i_3}(x) = \mathcal{A}_{i_1 i_2 i_3}x + \mathcal{B}_{i_1 i_2 i_3}. \tag{7}$$

The coefficient of (6), which is what we want, is rewritten as follows.

$$\begin{aligned}\sum_{x_L \in V^{(16)}+a} \mathcal{C}_{i_1 i_2 i_3}(y_R(x_L)) &= \sum_{x_L \in V^{(16)}+a} (\mathcal{A}_{i_1 i_2 i_3} y_R(x_L) + \mathcal{B}_{i_1 i_2 i_3}) \\ &= \mathcal{A}_{i_1 i_2 i_3} \sum_{x_L \in V^{(16)}+a} y_R(x_L) \quad \left(\because \sum \mathcal{B}_{i_1 i_2 i_3} = 0\right)\end{aligned} \tag{8}$$

Here we define x_s as $x_s = \sum_{x_L \in V^{(16)}+a} y_R(x_L)$. From equation (5) we have:

$$\begin{aligned}(8) &= \mathcal{A}_{i_1 i_2 i_3} x_s \\ &= \mathcal{C}_{i_1 i_2 i_3}(x_s) + \mathcal{B}_{i_1 i_2 i_3}\end{aligned}$$

The first and second terms are computed as follows:

$$\begin{aligned}\mathcal{C}_{i_1 i_2 i_3}(x_s) = {} & F[e_{i_1 i_2 i_3}](x_s) + F[e_{i_1 i_2}](x_s) + F[e_{i_1 i_3}](x_s) + F[e_{i_2 i_3}](x_s) \\ & + F[e_{i_1}](x_s) + F[e_{i_2}](x_s) + F[e_{i_3}](x_s) + F[\mathbf{0}](x_s), \\ \mathcal{B}_{i_1 i_2 i_3} = {} & F[e_{i_1 i_2 i_3}](\mathbf{0}) + F[e_{i_1 i_2}](\mathbf{0}) + F[e_{i_1 i_3}](\mathbf{0}) + F[e_{i_2 i_3}](\mathbf{0}) \\ & + F[e_{i_1}](\mathbf{0}) + F[e_{i_2}](\mathbf{0}) + F[e_{i_3}](\mathbf{0}) + F[\mathbf{0}](\mathbf{0}),\end{aligned}$$

$$\begin{aligned}\text{where} \quad e_{i_1 i_2 i_3} &= (0, \ldots, \overset{i_1}{\check{1}}, \ldots, \overset{i_2}{\check{1}}, \ldots, \overset{i_3}{\check{1}}, \ldots, 0) \in GF(2)^{32}, \\ e_{i_1 i_2} &= (0, \ldots, \overset{i_1}{\check{1}}, \ldots, \overset{i_2}{\check{1}}, \ldots, 0) \in GF(2)^{32}, \\ e_{i_1} &= (0, \ldots, \overset{i_1}{\check{1}}, \ldots, 0) \in GF(2)^{32}, \\ \mathbf{0} &= (0, \ldots, 0) \in GF(2)^{32}.\end{aligned}$$

The complexity required for this method is $(12+1) \times (M+1)$ times the computation of the round function F. When we use this method, the total complexity is $(5+144+1) \times 2^{17} + 13 \times (368+1) < 2^{25}$ times the computation of the round function F.