Higher Order Differential Attack on Step-Reduced Variants of *Luffa*

Yasuo Hatano[1] and Dai Watanabe[1]

Systems Development Laboratory, Hitachi, Ltd.,
292 Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan
dai.watanabe.td@hitachi.com

Abstract. In this paper, a higher order differential attack on the hash function *Luffa* is discussed. We confirmed that the algebraic degree of the permutation Q_j which is an important non-linear component of *Luffa* grows slower than an ideal case both by the theoretical estimate and the experiments. According to our estimate, we can construct a distinguisher for step-reduced variants of *Luffa* up to 7 out of 8 steps by using a block message. The attack for 7 steps requires 2^{216} messages. This attack does not pose any threat to the security of the full-step of *Luffa*, and this doesn't contradict any security claim of *Luffa*.

Keywords. Hash function, *Luffa*, Higher order differential attack, Non-randomness

1 Introduction

Luffa [4][1] is a family of hash functions submitted to NIST SHA-3 Competition and was selected as one of the second round candidates. The self-security evaluations are found in the supporting document [5], and they mainly discuss generic attacks and differential cryptanalysis. Besides, analyses based on algebraic approach is not discussed seriously in the supporting document.

An application of a higher order difference to cryptanalysis is suggested by Lai [8] and firstly applied to a block cipher by Knudsen [7]. The higher order differential attack is a tool to analyze the algebraic property of the target function, especially its algebraic degree. The application to the stream cipher is proposed by Dinur and Shamir [6] and Aumasson *et al.* proposed cube tester [1] which intends to detect the non-randomness of the target function. The cube tester has been applied not only to stream ciphers, but also to several hash functions submitted to SHA-3 Competition such as MD6 and Hamsi.

In the case of *Luffa*, Yamada and Kaneko pointed out that the 32-th order difference can distinguish five step functions of Q_j [9][2]. Starting from their result, we developed the higher order differential attack on step-reduced variants of

[1] Throughout this document, we discuss the algorithm submitted to Round 1.
[2] Their early results and a part of our discussion will be published in [10].

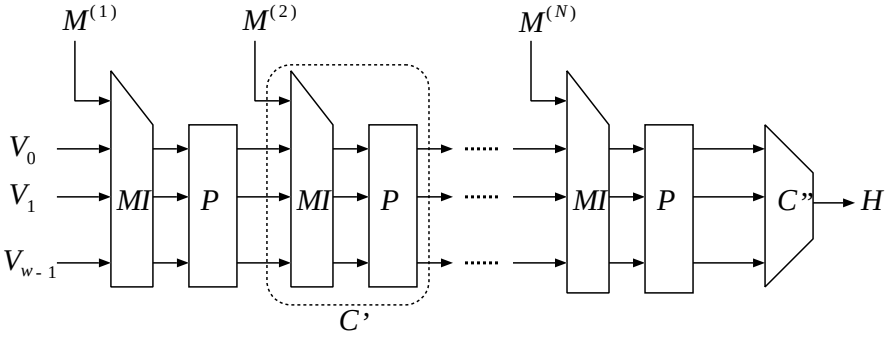

Fig. 1. A generic construction of a hash function based on a permutation

Luffa. In this paper, firstly we confirm that the algebraic degree of Q_j grows slower than an ideal case both by the theoretical estimate and the experiments. According to our estimate, we can construct a distinguisher for reduced step *Luffa* up to 7 out of 8 steps by using a block message. The attack for 7 steps requires 2^{216} messages. This attack does not pose any threat to the security of the full-step of *Luffa*, and this doesn't contradict any security claim of *Luffa*.

The rest of this paper is organized as follows: Firstly the specification of *Luffa* is briefly introduced in Section 2. Secondly the definition of the higher order difference and its basic property is introduced in Section 3. The increase of the algebraic degree by the iteration of the step function is investigated in Section 4. Then the higher order differential attack on step-reduced variant of the permutation Q_j and its extension to the hash function is given in Section 5. We conclude the discussion in Section 6.

2 Specification of *Luffa*

In this section, we introduce a part of the specification of *Luffa* which is needed to describe the attack. Please refer to [4] for the detail of the specification.

2.1 Chaining

The chaining of *Luffa* is a variant of a sponge function [2, 3]. Figure 1 shows the basic structure of the chaining. The chaining of a hash function consists of the intermediate mixing C' (called a round function) and the finalization C''.

Round Function The round function is a composition of a message injection function MI and a permutation P of $w \cdot n_b$ bits input. The permutation is divided into plural sub-permutation Q_j of n_b bits input (See Figure 2). Let the input

of the i-th round be $(H_0^{(i-1)}, \ldots, H_{w-1}^{(i-1)})$, then the output of the i-th round is given by

$$H_j^{(i)} = Q_j(X_j), \quad 0 \le j < w,$$
$$X_0 || \cdots || X_{w-1} = MI(H_0^{(i-1)}, \ldots, H_{w-1}^{(i-1)}, M^{(i)}),$$

where $H_j^{(0)} = V_j$.

In the specification of *Luffa*, the input length of the sub-permutation Q_j is fixed to $n_b = 256$ bits, and the number of the sub-permutations w is 3, 4 and 5 for the hash length 256, 384 and 512 bits respectively.

The message injection functions can be represented by the matrix over a ring $\mathrm{GF}(2^8)^{32}$. The map from an 8 words value $(a_0, \ldots, a_7)$ to an element of the ring is defined by $(\sum_{0 \le k < 8} a_{k,l} x^k)_{0 \le l < 32}$. Note that the least significant word a_7 is the coefficient of the heading term x^7 in the polynomial expression.

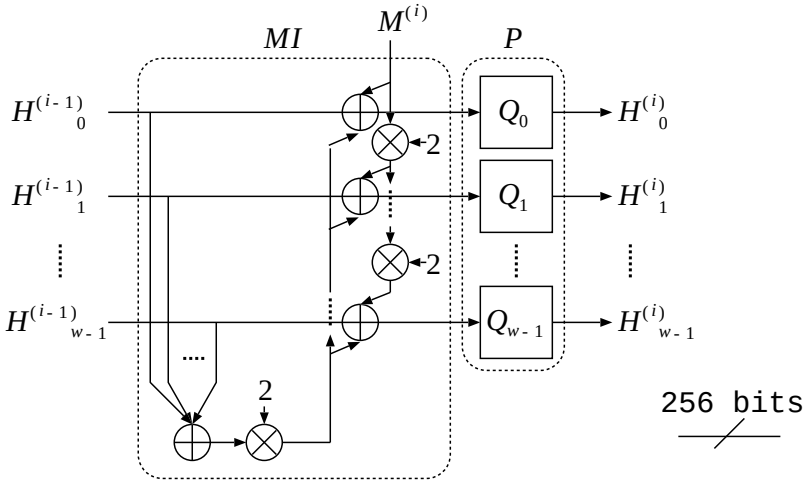

Fig. 2. The round function ($w = 3$)

Finalization The finalization consists of iterations of an output function OF and a round function with a fixed message `0x00...0`. If the number of (padded) message blocks is more than one, a blank round with a fixed message block `0x00...0` is applied at the beginning of the finalization.

The output function OF XORs all block values and outputs the resultant 256-bit value. Let the output at the i-th iteration be Z_i, then the output function

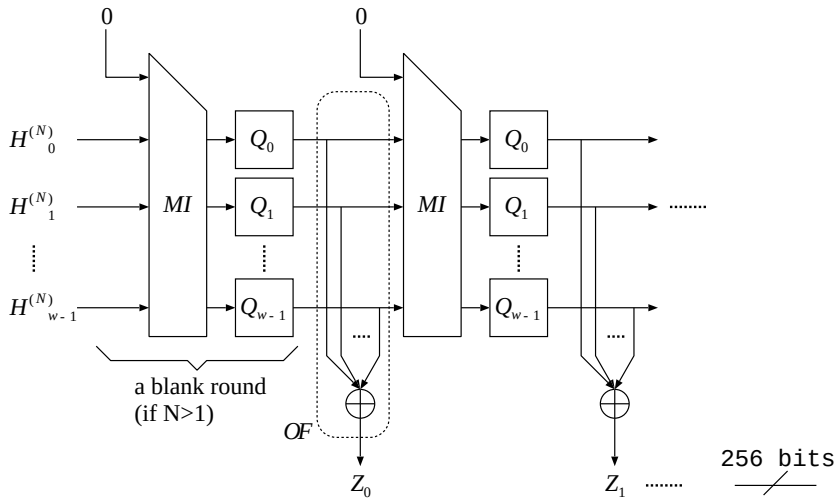

Fig. 3. The finalization function

is defined by

$$Z_i = \bigoplus_{j=0}^{w-1} H_j^{(N+i')},$$

where $i' = i$ if $N = 1$ and $i' = i + 1$ otherwise.

2.2 Non-Linear Permutation

The *Luffa* hash function uses a non-linear permutation Q_j whose input and output length is 256 bits. The permutation Q_j is defined as a composition of an input tweak and iterations of a step function `Step`. The number of iterations of a step function is 8 and the tweak is applied only once per a permutation.

At the beginning of the step function process, the 256 bits data stored in 8 32-bit registers is denoted by $a_k^{(r)}$ for $0 \leq k < 8$. The data before applying the permutation Q_j is denoted by b_k and the data after the tweak is denoted by $a_k^{(0)}$. The step function consists of the following three functions; `SubCrumb`, `MixWord`, `AddConstant`. The pseudo code for Q_j is given by

```
Permute(a[8], j){ //Permutation Q_j
    Tweak(a);
    for (r = 0; r < 8; r++){
        SubCrumb(a[0],a[1],[2],a[3]);
        SubCrumb(a[4],a[5],[6],a[7]);
        for (k = 0; k < 4; k++)
```

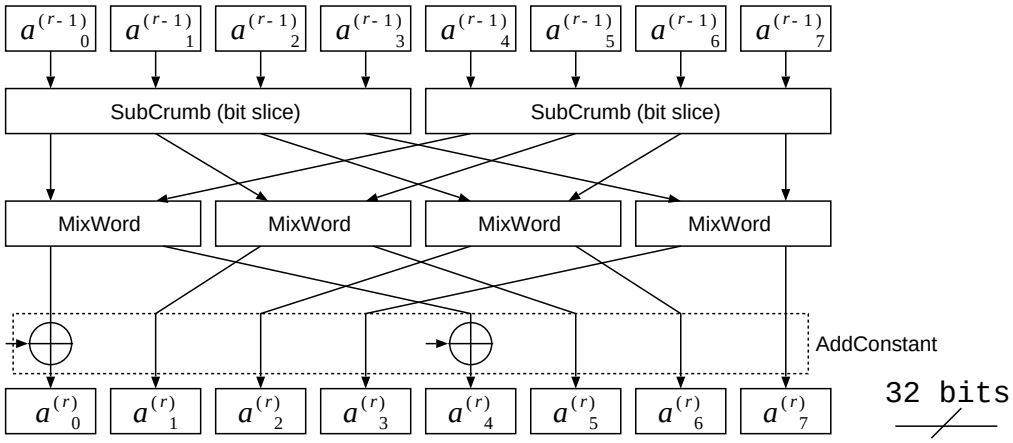

Fig. 4. The step function

```
            MixWord(a[k],a[k+4]);
        AddConstant(a, j, r);
    }
}
```

Each function is described below in turn and the tweaks are described in Section 2.2. We omit the description of `AddConstant` because it is not needed in this paper.

`SubCrumb` `SubCrumb` substitutes l-th bits of a_0, a_1, a_2, a_3 (or a_4, a_5, a_6, a_7) by an Sbox S defined by

$$S[16] = \{7, 13, 11, 10, 12, 4, 8, 3, 5, 15, 6, 0, 9, 1, 2, 14\}.$$

Let the output of `SubCrumb` be x_0, x_1, x_2, x_3 (or x_4, x_5, x_6, x_7). Then the substitution by `SubCrumb` is given by

$$x_{3,l}||x_{2,l}||x_{1,l}||x_{0,l} = S[a_{3,l}||a_{2,l}||a_{1,l}||a_{0,l}], \quad 0 \le l < 32,$$
$$x_{7,l}||x_{6,l}||x_{5,l}||x_{4,l} = S[a_{7,l}||a_{6,l}||a_{5,l}||a_{4,l}], \quad 0 \le l < 32.$$

`MixWord` `MixWord` is a linear permutation of two words. Let the output words be y_k and y_{k+4} where $0 \leq k < 4$. Then `MixWord` is given by the following equations:

$$\begin{aligned} y_{k+4} &= x_{k+4} \oplus x_k, \\ y_k &= x_k \lll \sigma_1, \\ y_k &= y_k \oplus y_{k+4}, \\ y_{k+4} &= y_{k+4} \lll \sigma_2, \\ y_{k+4} &= y_{k+4} \oplus y_k, \\ y_k &= y_k \lll \sigma_3, \\ y_k &= y_k \oplus y_{k+4}, \\ y_{k+4} &= y_{k+4} \lll \sigma_4. \end{aligned}$$

The parameters σ_i are given by $\sigma_1 = 2, \sigma_2 = 14, \sigma_3 = 10, \sigma_4 = 1$.

Tweaks For each permutation Q_j, the least significant four words of a 256-bit input are rotated by j bits to the left in 32-bit registers. Let the j-th block, k-th word input be $b_{j,k}$ and the tweaked word (namely the input to the first step function) be $a^{(0)}_{j,k}$, then the tweak is defined by

$$\begin{aligned} a^{(0)}_{j,k,l} &= b_{j,k,l}, \quad 0 \leq k < 4, \\ a^{(0)}_{j,k,l} &= b_{j,k,(l-j \bmod 32)}, \quad 4 \leq k < 8. \end{aligned}$$

3 Higher Order Differential Attack

3.1 Distinguishing Attack Based on A Hash Function

A hash function does not have an additional input other than a message so that it is not a pseudorandom function. The random distribution of the output is not a necessary condition for a collision resistance. However, non-random distribution of the output of the hash function is considered an undesirable property for applications such as a deterministic random bit generator.

We use the terminology *distinguisher* for functions which detect a kind of non-randomness according to [1]. Note that a distinguisher is usually a terminology for a function which distinguishes two random variables.

3.2 Higher Order Difference

Let $Y = E(X; K)$ be a function where $X \in \mathrm{GF}(2)^n$, $Y \in \mathrm{GF}(2)^m$ and $K \in \mathrm{GF}(2)^s$. Let $\{A_1, \ldots, A_i\}$ be a set of linearly independent vectors in $\mathrm{GF}(2)^n$ and $V^{(i)}$ be the sub-space spanned by these vectors. The i-th order difference is defined by

$$\Delta_{V^{(i)}} E(X; K) = \sum_{A \in V^{(i)}} E(X + A; K).$$

In the following, $\Delta^{(i)}$ denotes $\Delta_{V^{(i)}}$ if the choice of $V^{(i)}$ does not matter in the discussion. The basic fact of the higher order difference is that $\Delta^{(D+1)}E(X;K) = 0$ if the algebraic degree of E with respect to X is D. Therefore the higher order difference is used as the tool to evaluate the algebraic degree of the target function.

4 Algebraic Degree of Non-linear Permutation Q_j

It is pointed out in [5] that the Boolean polynomial expressions of the Sbox of *Luffa* are sparse, especially at the highest degree. The first step of the theoretical estimate is to observe how this property affects the growth of the algebraic degree throughout the iterations of the step functions. In the following, the r iterations of the step function is denoted by $Q_j^{(r)}$. The original permutation of *Luffa* is given by $Q_j = Q_j^{(8)}$.

4.1 Boolean Expressions of Sbox

Let the inputs and outputs of the Sbox be x_0, x_1, x_2, x_3 and y_0, y_1, y_2, y_3. Then the polynomial expressions of the relations between the input and output bits are given by

$$
\begin{array}{lllllllll}
y_0 = 1 & & +x_2 & +x_0x_1 & & & & +x_1x_3 + x_2x_3 + x_0x_1x_3 \\
y_1 = 1 & +x_0 & +x_2 & +x_0x_1 & +x_0x_2 & & +x_3 & +x_1x_3 + x_2x_3 + x_0x_1x_3 \\
y_2 = 1 & +x_1 & & & & & & +x_1x_3 + x_2x_3 + x_0x_1x_3 \\
y_3 = & x_0 & +x_1 & +x_2 & +x_0x_1 & +x_1x_2 & +x_0x_1x_2 & +x_1x_3
\end{array}
$$

4.2 Basic Facts

`MixWord()` is the function which sums up $y_{k,l}$ over the subscript l. In other words, $z_{k,l} = \texttt{MixWord}(y_k, y_{k+4})_{k,l} = \sum_{l \in \Omega} y_{k,l}$. Therefore it preserves the algebraic properties of the polynomial expressions of y_k. `AddConstant()` is the function which adds the step constants to the state, so that it also preserves the algebraic property of the polynomial expressions of y_k. These two facts make it easier to estimate the increase of algebraic degree of $Q_j^{(r)}$. Namely, we formally have to consider the iterations of the Sbox instead of the iterations of whole step function in the estimate of the algebraic degree.

It is clear from the simple observation of the Boolean expressions of the Sbox, the terms whose degrees are more than one and which has monomial x_3 in y_0, y_1, y_2 are equal. Let $\eta \cdot x_3$ be the common part in y_0, y_1, y_2 and ξ_k be the remainders. Then the multiplication of y_k and y'_k for $k \neq k'$ is given by

$$y_k \cdot y_{k'} = (\xi_k + \eta x_3)(\xi_{k'} + \eta x_3) = \xi_k \xi_{k'} + (\xi_k + \xi_{k'} + 1)\eta x_3. \tag{1}$$

Therefore, we get $\deg y_k \cdot y_{k'} < \deg y_k + \deg y_{k'}$ so that the algebraic degree does not increase ideally (namely times 3) at `SubCrumb`().

4.3 Recurrence Relations about Algebraic Degree

In the following, we identify the iterations of the Sboxes as the iterations of the step functions. Let us denote the inputs to the r-th Sbox by $(x_0^{(r-1)}, x_1^{(r-1)}, x_2^{(r-1)}, x_3^{(r-1)})$ and denote η, ξ by

$$\begin{aligned}
\eta^{(r)} &= \eta(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}) = x_1^{(r)} + x_2^{(r)} + x_0^{(r)} x_1^{(r)},\\
\xi_0^{(r)} &= \xi_0(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}) = 1 + x_2^{(r)} + x_0^{(r)} x_1^{(r)},\\
\xi_1^{(r)} &= \xi_1(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}) = 1 + x_0^{(r)} + x_2^{(r)} + x_0^{(r)} x_1^{(r)} + x_0^{(r)} x_2^{(r)},\\
\xi_2^{(r)} &= \xi_2(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}) = 1 + x_1^{(r)},\\
\xi_3^{(r)} &= \xi_3(x_0^{(r)}, x_1^{(r)}, x_2^{(r)}) = x_0^{(r)} + x_1^{(r)} + x_2^{(r)} + x_0^{(r)} x_1^{(r)} + x_1^{(r)} x_2^{(r)} + x_0^{(r)} x_1^{(r)} x_2^{(r)}.
\end{aligned}$$

In other words, $\eta \cdot x_3$ denotes the common terms of the polynomial expressions and ξ_k denotes the different terms which do not have the variable x_3. In addition, we denote the terms of degree d in $\eta^{(r)}$, $\xi_k^{(r)}$ by $\eta_d^{(r)}$, $\xi_{k,d}^{(r)}$ respectively.

Now we are going to estimate the algebraic degree of $x_k^{(r)}$, $\eta^{(r)}$, $\xi^{(r)}$ by the recurrence relations. We approximate the relations in order to simplify their expressions and the relation 1 is applied once for each variable in the estimation. Let us denote $\delta^{(r)} = \deg \eta^{(r-1)} + \deg x_3^{(r-1)}$, $\epsilon_{k,k'}^{(r)} = \deg \xi_k^{(r)} + \deg \xi_{k'}^{(r)}$. Then we have the following relations:

$$\deg \eta^{(r)} \sim \max(\epsilon_{0,1}^{(r-1)}, \deg \max(\xi_0^{(r-1)}, \xi_1^{(r-1)}) + \delta^{(r-1)}), \tag{2}$$

$$\deg \xi_0^{(r)} \sim \deg \eta^{(r)}, \tag{3}$$

$$\deg \xi_1^{(r)} \sim \max(\deg \xi_1^{(r-1)}, \deg \xi_2^{(r-1)}) + \max(\deg \xi_0^{(r-1)}, \delta^{(r-1)}), \tag{4}$$

$$\deg \xi_2^{(r)} \sim \max(\deg \xi_1^{(r-1)}, \delta^{(r-1)}), \tag{5}$$

$$\deg \xi_{3,2} = \max(\deg \xi_0^{(r-1)}, \deg \xi_2^{(r-1)}) + \max(\deg \xi_1^{(r-1)}, \delta^{(r-1)}), \tag{6}$$

$$\begin{aligned}
\deg \xi_{3,3} \sim \max(\ & \deg \xi_0^{(r-1)} + \deg \xi_1^{(r-1)} + \deg \xi_2^{(r-1)},\\
& \max(\epsilon_{0,1}^{(r-1)}, \epsilon_{0,2}^{(r-1)}, \epsilon_{1,2}^{(r-1)}) + \delta^{(r-1)}),
\end{aligned} \tag{7}$$

$$\deg x_0^{(r)} \sim \max(\epsilon_{0,1}^{(r-2)}, \delta^{(r-1)}), \tag{8}$$

$$\deg x_1^{(r)} \sim \max(\epsilon_{0,1}^{(r-2)}, \epsilon_{0,2}^{(r-2)}, \delta^{(r-1)}), \tag{9}$$

$$\deg x_2^{(r)} \sim \delta^{(r-1)}, \tag{10}$$

$$\begin{aligned}
\deg x_3^{(r)} \sim \max(\ & \deg \xi_0^{(r-2)} + \deg \xi_1^{(r-2)} + \deg \xi_2^{(r-2)},\\
& \max(\epsilon_{0,1}^{(r-2)}, \epsilon_{0,2}^{(r-2)}, \epsilon_{1,2}^{(r-2)}, \deg x_3^{(r-2)}) + \delta^{(r-2)},\\
& 2\deg \xi_1^{(r-2)} + \deg \xi_3^{(r-2)}).
\end{aligned} \tag{11}$$

The detailed calculations to get the relations is given in Appendix A.

4.4 Theoretical Estimate of Algebraic Degrees

Table 1 shows the pace of increase of algebraic degrees of variables x_k, ξ_k, η from the recurrent relations 2 to 11 and the initial values at $r = 0, 1$. The

input/output length of the non-linear permutation $Q_j^{(r)}$ is 256 bits so that the algebraic degrees are at most 256. However we put the estimated degrees as it is, even if it is more than 256, in order to clarify the pace of increase.

Table 1. Pace of increase of algebraic degrees

r	$x_0^{(r)}$	$x_1^{(r)}$	$x_2^{(r)}$	$x_3^{(r)}$	$\xi_0^{(r)}$	$\xi_1^{(r)}$	$\xi_2^{(r)}$	$\xi_3^{(r)}$	$\eta^{(r)}$
0	1	1	1	1	2	2	1	2	2
1	3	3	3	3	5	5	3	7	5
2	8	8	8	7	13	13	8	18	13
3	20	20	20	18	33	33	20	46	33
4	51	51	51	46	84	84	51	117	84
5	130	130	130	117	214	214	130	298	214
6	331	331	331	298	545	545	331	759	545
7	843	843	843	759	1,388	1,388	843	1,933	1,388
8	2,147	2,147	2,147	1,933	3,535	3,535	2,147	4,923	3,535

5 Higher Order Differential Attack on *Luffa*

If the Sbox is an ideal one, the algebraic degree of the permutation $Q_j^{(r)}$ should be about 3^r. However, as shown in the previous section, the degree increases slower than the ideal case. Especially, the high order part $\eta^{(r-1)} \cdot x_3^{(r-1)}$ of the variables $x_k^{(r)}$ are common, so that it can be cancelled by the addition $x_k^{(r)} + x_{k'}^{(r)}$.

5.1 Theoretical Estimate

We propose to use $x_k^{(r)} + x_{k'}^{(r)}$ as the distinguisher in the higher order differential attack in order to eliminate the common part $\eta \cdot x_3$. In this attack, the important variable is not $x_k^{(r)}$, but $\xi_0^{(r-1)}, \xi_1^{(r-1)}, \xi_2^{(r-1)}$.

In addition, there are two techniques to skip the first Sboxes. The first one is to choose a Λ-set in which the inputs to the Sboxes are `constant` or `active`. For example, if the Λ-set takes all values in $x_{k,l}$ for $0 \le k < 8$ and $0 \le l < t$, we can ignore the effect of `SubCrumb`() at the first step. The second technique is to move only a bit per an Sbox. This technique is applicable only if the algebraic degree of the target function is small.

These observations indicate that the number of steps r to be attacked can be estimated by the maximum degree of $\xi_k^{(r-2)}$. In Table 1, $\max_k \deg \xi_k^{(5)}$ is 214 so that $Q_j^{(6)}$ is distinguishable from a random permutation by calculating 214-th order difference. This distinguisher for 6 steps does not depend on the choice of the Λ-set. In addition, $Q_j^{(7)}$ is distinguishable from a random function by choosing the Λ-set such that all possible values of $x_{k,l}$ for $0 \le k < 8$ and

$0 \leq l < 27$ appears once. This attack requires 2^{216} messages. On the other hand, $Q_j^{(8)}$ is not expected to be distinguishable because $\max \deg \xi_k^{(6)} = 545 > 256$.

5.2 Experimental Inspection

Here we check the theoretical estimates by experiments. As mentioned, there are two techniques to ignore the effect of the first step and we applied the second one, namely "a bit per an Sbox" manner. If we apply the first technique, the active Sboxes are relatively sparse, so that it is possible to skip the `SubCrumb`() in the second step by choosing a good alignment of the active Sboxes. However, our purpose is not to optimize the attack, but to check if the theoretical estimates summarized in Table 1 is reliable so that this kind of "unexpected" skip is not desired. Therefore, we applied t-th order difference by moving the most least t bits of the variable $x_0^{(0)}$. And we calculated each higher order difference for 100 times by generating the initial states randomly.

Table 2 summarizes the experimental results. The numerical values in the table shows the ratio such that $x_0 = x_1 = x_2$ and $x_4 = x_5 = x_6$ hold. Besides, the values in the parentheses shows the ratio such that one of the equations $x_0 = x_1$, $x_0 = x_2$, $x_1 = x_2$, $x_4 = x_5$, $x_4 = x_6$, $x_5 = x_6$ holds. In other words, the values in the parentheses mean the ratio of the distinguishing attack being successful.

Table 3 shows the comparison between the theoretical estimates taken from Table 1.

We calculated the algebraic degree of $Q_j^{(r)}$ from the experimental results by the order. Let t be the lowest number such that the t-th order differential of $x_k^{(r)}$ is equal to zero with probability one. The degree of $Q_j^{(r)}$ is formally estimated at $t-1$. This may cause the contradictions in Table 3 such that the degree of $\xi_k^{(r-2)}$ is larger than that of $x_k^{(r-1)}$ for $r = 1, 2$. In other cases, the Table 3 indicates that the theoretical estimates in Table 1 are very close to the experimental results in Table 2.

We append a note that the $t-1$-th order differentials are rarely constants, so that it might be better to estimate the degree of $Q_j^{(r)}$ by $t-1$.

5.3 Higher Order Differential Attack on The Hash Function

The higher order differential attack on a hash function does not violate the central three requirements for a hash function, namely collision resistance, second preimage resistance, preimage resistance. On the other hand, the distinguishing attacks are useful to check whether or not the target function has pseudo-randomness which is also required to a hash function. Here we consider the higher order differential attack on reduced step *Luffa* hash function.

The first point of *Luffa* is that there is no blank round if the message length is less than 256 bits. In this case, the message is mixed by the message injection function MI, permuted by non-linear permutation Q_j, then the XORed 256-bit value is output. Therefore, it might be possible to construct a distinguisher

Table 2. Experimental results

	Number of steps				
Order	1	2	3	4	5
1	.28 (1.00)	.00 (.39)	.00 (.00)	.00 (.00)	.00 (.00)
2	1.00 (1.00)	1.00 (1.00)	.00 (.12)	.00 (.00)	.00 (.00)
3	1.00 (1.00)	1.00 (1.00)	.25 (.56)	.00 (.00)	.00 (.00)
4	1.00 (1.00)	1.00 (1.00)	.60 (.93)	.00 (.00)	.00 (.00)
5	1.00 (1.00)	1.00 (1.00)	.90 (1.00)	.00 (.00)	.00 (.00)
6	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.01 (.01)	.00 (.00)
7	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.01 (.04)	.00 (.00)
8	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.07 (.16)	.00 (.00)
9	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.32 (.45)	.00 (.00)
10	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.61 (.83)	.00 (.00)
11	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.90 (.97)	.00 (.00)
12	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.97 (1.00)	.00 (.00)
13	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
14	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
15	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
16	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
17	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.01)
18	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
19	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
20	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
21	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.00 (.00)
22	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.02 (.03)
23	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.03 (.04)
24	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.12 (.13)
25	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.20 (.21)
26	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.37 (.38)
27	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.45 (.46)
28	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.69 (.71)
29	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.81 (.83)
30	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.83 (.85)
31	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.91 (.93)
32	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	1.00 (1.00)	.99 (.99)

Table 3. The summary of the algebraic degrees

Number of steps		1	2	3	4	5	6	7	8
Algebraic degree	Theoretical estimate	1	3	8	20	51	130	–	–
($\max_{0\le k\le 2} x_k^{(r-1)}$)	Experimental result	1	1	7	18	–	–	–	–
Distinguisher's degree	Theoretical estimate	–	2	5	13	33	84	214	–
($\max_{0\le k\le 2} \xi_k^{(r-2)}$)	Experimental result	2	2	6	13	–	–	–	–

based on a higher order difference if the algebraic degree of Q_j is smaller than 256 for all j. Because Q_js are only the non-linear component in *Luffa*, and they are different each other only at their tweaks and their step constants, the interference for the attack is the influences by MI and the tweaks. In fact, it is not difficult to ignore these two interferences.

Firstly, the message injection function MI consists of the constant multiplication over $\mathrm{GF}(2^8)^{32}$. This map stabilizes subspaces of $\mathrm{GF}(2^8)^{32}$ given by a natural injection of $\mathrm{GF}(2^8)^t$ where $t \leq 32$. Therefore the influence of the message injection function MI is ignorable by choosing such a subspace as a Λ-set. Secondly, the tweaks rotate the lower 4 words a_4, a_5, a_6, a_7 by j bits to the left in a word. Obviously, the tweaks preserve the properties `active` and `constant`. Therefore the Λ-set which cancels the influence of the message injection function also cancels that of tweaks.

These two facts indicate that the distinguisher for $Q_j^{(7)}$ is also applicable to the reduced step hash function as it is.

5.4 Probabilistic Distinguisher

Table 2 shows that the behavior of the distinguisher is probabilistic if the order is less than the expected algebraic degree. Here we discuss what causes the probabilistic behavior and whether or not this is useful to reduce the complexity of the attack.

One possible reason is that the terms of high degree of $Q_j^{(r)}$ are sparsely distributed. In the experiments, most of the input bits are fixed so that the variables of the polynomial expressions are replaced by randomly chosen constants. This does not matter if the there are lot of terms of high degree because some terms consisting of variables still exist. However, all terms of high degree could include the variables which are replaced by constants, or worse, they might vanishes if the constant is zero.

If the target function is sufficiently random, the probability to eventually find a local collision $x_k = x_{k'}$ for any k, k' is given by $6 \cdot 2^{-(32+1)/2} \sim 2^{-14}$ and it is small[3]. Therefore $x_k + x_{k'}$ can be used as a distinguisher even if the event is probabilistic. For example, Table 2 shows that 3 of 100 trials successfully found the partial collision with the 22-th order difference for 5 steps. In this case, the computational complexity is $2^{22} \times 100 \sim 2^{28.6}$ and it is significantly smaller than 2^{33}, which is the complexity of the attack with the deterministic distinguisher.

So far, we have no idea to estimate the frequency of this event. And it is not clarified yet how much the computational complexity is reduced in the case of 7 steps. In addition, the expected degree of the distinguisher for 8 steps is much larger than 256 so that the distinguisher is expected to include many high order terms. We expect that it is difficult to apply the probabilistic distinguisher for 8 steps.

[3] In [8] Lai pointed out that $\mathrm{Prob}(\Delta_{V_i} f(a) = b)$ is either 0 or at least 2^{i-n} where $f : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$. But this is not our case because the domain of our distinguisher is larger than the range

6 Conclusion

In this paper, a higher order differential attack on the hash function *Luffa* is discussed. We confirmed that the algebraic degree of the underlying non-linear permutation Q_j grows slower than an ideal case both by the theoretical estimate and the experiments. According to our estimate, we can construct a distinguisher for reduced step *Luffa* up to 7 out of 8 steps by using a block message. The attack for 7 steps requires 2^{216} messages. This attack does not pose any threat to the security of the full-step of *Luffa*, and this doesn't contradict any security claim of *Luffa*.

Acknowledgement

We thank to T. Yamada and T. Kaneko informed their result before publishing. Most of the discussions in this paper are started from their experimental results and their observations.

References

1. J.P. Aumasson, I. Dinur, W. Meier and A. Shamir "Cube Testers and Key Recovery Attacks On Reduced-Round MD6 and Trivium," *Fast Software Encryption, FSE 2009*, Lecture Notes in Computer Science, vol. 5665, Springer-Verlag, pp. 1–22, 2009.
2. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "Sponge Functions," Ecrypt Hash Workshop 2007.
3. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "On the Indifferentiability of the Sponge Construction," *Advances in Cryptology, Eurocrypt 2008*, pp. 181–197, 2008.
4. C. De Cannière, H. Sato, D. Watanabe, "Hash Function Luffa: Specification," Submission to NIST SHA-3 Competition, 2008. Available at http://www.sdl.hitachi.co.jp/crypto/luffa/.
5. C. De Cannière, H. Sato, D. Watanabe, "Hash Function Luffa: Supporting Document," Submission to NIST SHA-3 Competition, 2008. Available at http://www.sdl.hitachi.co.jp/crypto/luffa/.
6. I. Dinur and A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials," Cryptology ePrint Archive, Report 2008/385.
7. L. R. Knudsen, "Truncated and Higher Order Differentials," *Fast Software Encryption, FSE '94*, Lecture Note in Computer Science vol. 1008, pp. 196–211, Springer-Verlag, 1994.
8. X. Lai, "Higher order derivatives and differential cryptanalysis," *Proc. Symposium on Communication, Coding and Cryptography*, pp. 227–233, Kluwer Academic Publishers, 1994.
9. T. Yamada and T. Kaneko, Private communication, 2 July 2009.
10. T. Yamada, D. Watanabe, Y. Hatano and T. Kaneko, "A Higher Order Differential Property of the Non-Linear Permutation in Hash Function *Luffa*," *The 12th Computer Security Symposium, CSS2009*, October 26-28, 2009, to be appeared (*In Japanese*).

A Recurrence Relations

The symbol "$\sim$" means the simplification of the expression which (is considered) preserves the algebraic degree.

A.1 Recurrence Relation of $\eta^{(r)}$

$$\begin{aligned}\eta^{(r)} &= x_1^{(r)} + x_2^{(r)} + x_0^{(r)}x_1^{(r)} \\ &\sim x_0^{(r)}x_1^{(r)} \\ &= (\xi_0^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)})(\xi_1^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)}) \\ &= \xi_0^{(r-1)}\xi_1^{(r-1)} + (\xi_0^{(r-1)} + \xi_1^{(r-1)} + 1)\eta^{(r-1)}x_3^{(r-1)} \\ &\sim \xi_0^{(r-1)}\xi_1^{(r-1)} + (\xi_0^{(r-1)} + \xi_1^{(r-1)})\eta^{(r-1)}x_3^{(r-1)}. \end{aligned} \tag{12}$$

A.2 Recurrence Relation of $\xi_k^{(r)}$

$$\xi_0^{(r)} = \xi_{0,0}^{(r)} + \xi_{0,1}^{(r)} + \xi_{0,2}^{(r)} \sim \xi_{0,2}^{(r)} = x_0^{(r)}x_1^{(r)} = \eta^{(r)}. \tag{13}$$

$$\begin{aligned}\xi_1^{(r)} &= \xi_{1,0}^{(r)} + \xi_{1,1}^{(r)} + \xi_{1,2}^{(r)} \\ &\sim \xi_{1,2}^{(r)} \\ &= x_0^{(r)}x_1^{(r)} + x_0^{(r)}x_2^{(r)} \\ &= (\xi_1^{(r-1)} + \xi_2^{(r-1)})(\xi_0^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)}). \end{aligned} \tag{14}$$

$$\xi_2^{(r)} = \xi_{2,0}^{(r)} + \xi_{2,1}^{(r)} \sim \xi_{2,1}^{(r)} = x_1^{(r)} = \xi_1^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)}. \tag{15}$$

$$\begin{aligned}\xi_{3,2}^{(r)} &= (x_0^{(r)} + x_2^{(r)})x_1^{(r)} \\ &= (\xi_0^{(r-1)} + \xi_2^{(r-1)})(\xi_1^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)}). \end{aligned} \tag{16}$$

$$\begin{aligned}\xi_{3,3}^{(r)} &= x_0^{(r)}x_1^{(r)}x_2^{(r)} \\ &= (\xi_0^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)})(\xi_1^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)})(\xi_2^{(r-1)} + \eta^{(r-1)}x_3^{(r-1)}) \\ &\sim \xi_0^{(r-1)}\xi_1^{(r-1)}\xi_2^{(r-1)} \\ &\quad +(\xi_0^{(r-1)}\xi_1^{(r-1)} + \xi_0^{(r-1)}\xi_2^{(r-1)} + \xi_1^{(r-1)}\xi_2^{(r-1)})\eta^{(r-1)}x_3^{(r-1)}. \end{aligned} \tag{17}$$

A.3 Recurrence Relation of $x_k^{(r)}$

$$\begin{aligned}
x_0^{(r)} &= \xi_{0,0}^{(r-1)} + \xi_{0,1}^{(r-1)} + \xi_{0,2}^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&\sim \xi_{0,2}^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&= x_0^{(r-1)} x_1^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&= (\xi_0^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)})(\xi_1^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)}) + \eta^{(r-1)} x_3^{(r-1)} \\
&\sim \xi_0^{(r-2)} \xi_1^{(r-2)} + \eta^{(r-1)} x_3^{(r-1)}. \qquad (18)
\end{aligned}$$

$$\begin{aligned}
x_1^{(r)} &= \xi_{1,0}^{(r-1)} + \xi_{1,1}^{(r-1)} + \xi_{1,2}^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&\sim \xi_{1,2}^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&= x_0^{(r-1)} x_1^{(r-1)} + x_0^{(r-1)} x_2^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \\
&\sim \xi_0^{(r-2)} (\xi_1^{(r-2)} + \xi_2^{(r-2)}) + \eta^{(r-1)} x_3^{(r-1)}. \qquad (19)
\end{aligned}$$

$$x_2^{(r)} = \xi_{2,0}^{(r-1)} + \xi_{2,1}^{(r-1)} + \eta^{(r-1)} x_3^{(r-1)} \sim \eta^{(r-1)} x_3^{(r-1)}. \qquad (20)$$

$$\begin{aligned}
x_3^{(r)} &= \xi_{3,1}^{(r-1)} + \xi_{3,2}^{(r-1)} + \xi_{3,3}^{(r-1)} + x_1^{(r-1)} x_3^{(r-1)} \\
&\sim \xi_{3,3}^{(r-1)} + x_1^{(r-1)} x_3^{(r-1)} \\
&= x_0^{(r-1)} x_1^{(r-1)} x_2^{(r-1)} + x_1^{(r-1)} x_3^{(r-1)} \\
&= (\xi_0^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)})(\xi_1^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)})(\xi_2^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)}) \\
&\quad + (\xi_1^{(r-2)} + \eta^{(r-2)} x_3^{(r-2)})(\xi_{3,2}^{(r-2)} + \xi_{3,3}^{(r-2)} + x_1^{(r-2)} x_3^{(r-2)}) \\
&\sim \xi_0^{(r-2)} \xi_1^{(r-2)} \xi_2^{(r-2)} \\
&\quad + (\xi_0^{(r-2)} \xi_1^{(r-2)} + \xi_0^{(r-2)} \xi_2^{(r-2)} + \xi_1^{(r-2)} \xi_2^{(r-2)} + \xi_{3,3}^{(r-2)}) \eta^{(r-2)} x_3^{(r-2)} \\
&\quad \xi_1^{(r-2)} (\xi_{3,3}^{(r-2)} + x_1^{(r-2)} x_3^{(r-2)}). \qquad (21)
\end{aligned}$$