Pseudo-Cryptanalysis of Luffa

Keting Jia[1,2], Yvo Desmedt[3], Lidong Han[1], Xiaoyun Wang[1,2]*

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education,
Shandong University, China
{ktjia, hanlidong}@mail.sdu.edu.cn
[2] Institute for Advanced Study, Tsinghua University, China
xiaoyunwang@mail.tsinghua.edu.cn
[3] Department of Computer Science, University College London, UK
y.desmedt@cs.ucl.ac.uk

Abstract. In this paper, we present the pseudo-collision, pseudo-second-preimage and pseudo-preimage attacks on the SHA-3 candidate algorithm Luffa. The pseudo-collisions and pseudo-second-preimages can be found easily by computing the inverse of the message injection function at the beginning of Luffa. We explain in details the pseudo-preimage attacks. For Luffa-224/256, given the hash value, only 2 iteration computations are needed to get a pseudo-preimage. For Luffa-384, finding a pseudo-preimage needs about 2^{64} iteration computations with 2^{67} bytes memory by the extended generalized birthday attack. For Luffa-512, the complexity is 2^{128} iteration computations with 2^{132} bytes memory.
It is noted that, we can find the pseudo-collision pairs and the pseudo-second images only changing a few different bits of initial values. That is directly converted to the forgery attack on NMAC in related key cases.

Key words: Luffa, pseudo-collision, pseudo-second-preimage, pseudo-preimage, generalized birthday attack

1 Introduction

A cryptographic hash function is defined as a function that computes a fixed size message digest from arbitrary size messages. It has been widely used as a fundamental primitive in many cryptographic schemes and protocols, such as electronic signature, authentication of messages, electronic commerce and bit commitment, etc. In the past years, the cryptanalysis of hash functions has achieved tremendous progress with the construction of collisions. In particular, Wang et al. proposed new techniques to find efficiently collisions on the main hash functions from the MD4 family (e.g., MD4 [8], RIPEMD [8], MD5 [11], SHA-0 [9] and SHA-1 [10]). Moreover the techniques can be applied to explore the second-preimage of MD4 [12], forgery and partial key-recovery attacks on HMAC and NMAC [3,4]. Kelsey and Schneier [5] provided a second preimage attack on the iterated hash functions with Merkle-Damgård strengthening, which shows a vulnerability of the Merkle-Damgård construction. Responding to advances in the cryptanalysis of hash functions, NIST held two hash workshops to evaluate the

[*] Corresponding author

security of its approved hash functions and to solicit public comments on its cryptographic hash function policy and standard. Finally, NIST opened a public competition to develop a new hash function called “SHA-3”, similar to the development process of the Advanced Encryption Standard (AES). There are 64 new proposals for hash functions have been submitted to the SHA-3 project, of which 51 submissions have come into the first round. In July, 2009, NIST has selected 14 second round candidates of the SHA-3. Luffa [2] is one of them, proposed by De Cannière, Sato and Watanabe.

In this paper, we give some cryptanalytic results of Luffa with free initial values. The pseudo-collision and pseudo-second-preimage can be obtained easily by the message injection function of Luffa, which only changes a few bits of the initial values. This paper shows a pseudo-collision and pseudo-second-preimage example for Luffa-256 and gives the actual attacks. For Luffa-224/256, only 2 iteration computations are needed to get the pseudo-preimage. A pseudo preimage example for Luffa-256 is shown in this paper, which only changes 2 256-bit words of the initial values with 3 256-bit words. We use the extended generalized birthday attack [7] to compute the pseudo-preimage of Luffa-384 with 2^{64} iteration computations and 2^{64} table lookups. The time complexity and data complexity are both 2^{128} to get the pseudo-preimage for Luffa-512.

This paper is organized as follows. In Section 2, we list some notations and give a brief description of Luffa. Section 3 shows the pseudo-collision and pseudo-second-preimage attacks on Luffa. The pseudo-preimage attacks for Luffa is given in Section 4. The improved pseudo-preimage attacks for Luffa-384/512 are shown in Section 5. Finally, we summarize our results in Section 6.

2 Preliminaries and Notations

In this section, we first list some notations used in this paper, and then give a brief description of Luffa.

2.1 Notations

$X\|Y$: the concatenation of two messages X and Y.
$h_w(X)$: the w most significant bits of X.
$l_w(X)$: the w least significant bits of X.
$\lfloor a \rfloor$: the greatest integer less than or equal to a.
$(b_0, b_1, \ldots, b_m)^T$: the transposed matrix of $(b_0, b_1, \ldots, b_m)$, where $b_i (1 \leq i \leq m)$ are column vectors.
$a \lll j$: left rotation of a by j bits.

2.2 Description of Luffa

Luffa [2], a candidate algorithm for the second round of the SHA-3, was proposed by De Cannière et al. The chaining of Luffa is a variant of a sponge function. Fig.1 depicts the basic structure. For any message, Luffa can produce the hash values with 224, 256, 384 or 512 bits, which are denoted as Luffa-224/256/384/512 respectively. The message padding method consists of appending a single bit ‘1’ followed by the minimum bits of

'0' such that the length of the result is a multiple of 256. Let $M = M_0 \| \cdots \| M_{m-1}$ be a message after padding, where $M_i (0 \leq i < m)$ are 256-bit blocks. The *iteration function* of Luffa is a composition of a message injection function MI and a permutation P with w 256-bit inputs, where $w = 3, 4$ or 5 for Luffa-224/256, Luffa-384 and Luffa-512 respectively. The permutation P includes w permutations Q_0, Q_1, ..., Q_{w-1}, where Q_j is the permutation with 256-bit input, $j = 0, 1, \ldots, w-1$. Let the input of the $i-$th iteration be $(H_0^{(i-1)}, \ldots, H_{w-1}^{(i-1)}, M_{i-1})$, the $i-$th iteration is computed as follows,

$$\begin{aligned} X_0 \| \cdots \| X_{w-1} &= MI(H_0^{(i-1)}, \ldots, H_{w-1}^{(i-1)}, M_{i-1}), \\ H_j^{(i)} &= Q_j(X_j), j = 0, 1, \ldots, w-1, \end{aligned}$$

where $(H_0^i, \ldots, H_{w-1}^i)$ is the i-th iteration output, and $(H_0^0, \ldots, H_{w-1}^0)$ is the initial value. Final operations, called a *finalization* are used to the chaining value $(H_0^{(m-1)}, \ldots, H_{w-1}^{(m-1)})$. For Luffa-224/256, the finalization consists of a blank iteration and a XOR operation OF, where the blank iteration means an iteration with a fixed message $M_m = \mathbb{0}$, where $\mathbb{0}$ denotes 256-bit zeros, the operation OF XORs w 256-bit values and outputs the result 256-bit value. For Luffa-384/512, the finalization includes two blank iterations and two XOR operations, see Fig. 1. The output of Luffa-256 is Z_0, the output of Luffa-512 is $Z_0 \| Z_1$. The outputs of Luffa-224 and Luffa-384 are the truncation of the Luffa-256 and Luffa-512 respectively. Here

$$Z_i = \bigoplus_{j=0}^{w-1} H_j^{(m+i)}, \quad i = 0, 1.$$

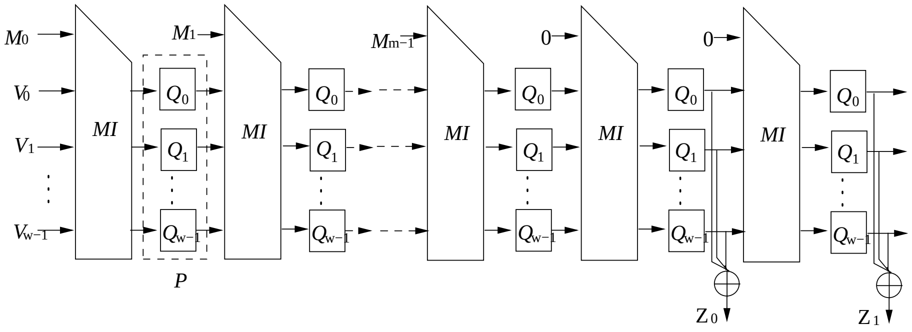

Fig. 1. The Structure of Luffa Hash Function

Message Injection Function MI The message injection functions MI can be represented by the matrix over a field $GF(2^8)$. The multiplication over the field $GF(2^8)$ is modulo $\phi(x) = x^8 + x^4 + x^3 + x + 1$, corresponding to "0x11b". The map from 8 32-bit words $(h_0,\ldots,h_7)$ to 32 8-bit elements of the field is defined by $(\Sigma_{0\leq k<8}h_{k,l}x^k)_{0\leq l<32}$. Let $A_{w\times(w+1)}=(a_0,a_1,\ldots,a_{w-1},a_w)$ represent the matrix of MI, where $a_i(0\leq i\leq w)$ are column vectors. Then $(X_0,X_1,\ldots,X_{w-1})^T=A_{w\times(w+1)}\circ(H_0,H_1,\ldots,H_{w-1},M)^T$. For Luffa-224/256, w=3,

$$A_{w\times(w+1)}=\begin{pmatrix}0x3,0x2,0x2,0x1\\0x2,0x3,0x2,0x2\\0x2,0x2,0x3,0x4\end{pmatrix},$$

where the elements 0x1, 0x2, 0x3, 0x4 correspond to polynomials $1,x,x+1,x^2$ respectively.

For Luffa-384,

$$A_{w\times(w+1)}=\begin{pmatrix}0x4,0x6,0x6,0x7,0x1\\0x7,0x4,0x6,0x6,0x2\\0x6,0x7,0x4,0x6,0x4\\0x6,0x6,0x7,0x4,0x8\end{pmatrix}.$$

For Luffa-512,

$$A_{w\times(w+1)}=\begin{pmatrix}0x0F,0x08,0x0A,0x0A,0x08,0x01\\0x08,0x0F,0x08,0x0A,0x0A,0x02\\0x0A,0x08,0x0F,0x08,0x0A,0x04\\0x0A,0x0A,0x08,0x0F,0x08,0x08\\0x08,0x0A,0x0A,0x08,0x0F,0x10\end{pmatrix}.$$

The Permutation Q_j The permutation Q_j is defined as a composition of an input tweak and 8 steps. Let $a_0,\ldots,a_7$ be the 256-bit input of the Q_j, $b_0,\ldots,b_7$ be the output of tweak. The tweak is defined as follows:

$$\begin{aligned}&b_i=a_i,\text{for }1\leq i<4;\\&b_i=a_i\lll j,\text{for }4\leq i<8.\end{aligned}$$

After tweak, there are 8 steps in the permutation, and each step consists of the following three functions: SubCrumb, MixWord and AddConstant.

SubCrumb is defined as:

$$\begin{aligned}x_{3,l}\|x_{2,l}\|x_{1,l}\|x_{0,l}&=S(b_{3,l}\|b_{2,l}\|b_{1,l}\|b_{0,l}),0\leq l<32,\\x_{4,l}\|x_{7,l}\|x_{6,l}\|x_{5,l}&=S(b_{4,l}\|b_{7,l}\|b_{6,l}\|b_{5,l}),0\leq l<32,\end{aligned}$$

where S denotes a S-box with 4-bit input and 4-bit output. MixWord is defined as:

$$\begin{aligned}y_{k+4}&=x_{k+4}\oplus x_k, & y_k&=x_k\lll 2,\\y_k&=y_k\oplus y_{k+4}, & y_{k+4}&=y_{k+4}\lll 14,\\y_{k+4}&=y_{k+4}\oplus y_k, & y_k&=y_k\lll 10,\\y_k&=y_k\oplus y_{k+4}, & y_{k+4}&=y_{k+4}\lll 1.\end{aligned}$$

We do not give the description for AddConstant since it has no impact on our cryptanalysis. For more details about Luffa, consult [2].

3 Pseudo-Collision and Pseudo-Second-Preimage Attacks on Luffa

In this section, we give some cryptanalysis for Luffa when the initial value *IV* is free. Flipping 5 bits of *IV* for Luffa-256 is enough to get a pseudo-collision or pseudo-second-preimage. For Luffa-384, 7 bits of *IV* are needed to be changed to get a pseudo-collision or pseudo-second-preimage. There is a 12-bit difference in the *IV* to get a pseudo-collision or pseudo-second-preimage for Luffa-512. This can be used to construct the related key attack for the corresponding MACs using the secret key as initial value.

For the message injection function *MI*, the input is $(w+1)$ 256-bit words, and the output is w 256-bit words. So, it is a many-to-one function. It is easy to know that, any w columns of the *MI* matrix consists of an invertible matrix. So there are exactly 2^{256} inputs corresponding to any given output of *MI*. Given any *MI* output $(X_0, X_1, \ldots, X_{w-1})$, if one entry of $H_0, H_1, \ldots, H_{w-1}$ and M is fixed, we can easily compute the solution to other entries. Any pair of inputs with the same output of *MI* consists of a pseudo-collision of Luffa, that is the output difference of *MI* is zero.

For Luffa-224/256, the input difference is (ΔH_0, ΔH_1, ΔH_2, ΔM), and the output difference of *MI* is $(\mathbb{0}, \mathbb{0}, \mathbb{0})$, here $\mathbb{0}$ denotes 256-bit zeros. They satisfy the following equations.

$$\begin{aligned}
3 \circ \Delta H_0 \oplus 2 \circ \Delta H_1 \oplus 2 \circ \Delta H_2 \oplus \Delta M &= \mathbb{0},\\
2 \circ \Delta H_0 \oplus 3 \circ \Delta H_1 \oplus 2 \circ \Delta H_2 \oplus 2\Delta M &= \mathbb{0},\\
2 \circ \Delta H_0 \oplus 2 \circ \Delta H_1 \oplus 3 \circ \Delta H_2 \oplus 4\Delta M &= \mathbb{0}.
\end{aligned}$$

From the equations, it is easy to get $\Delta H_0 = 0xf2 \circ \Delta M$, $\Delta H_1 = 0xf1 \circ \Delta M$ and $\Delta H_2 = 0xf7 \circ \Delta M$. Let *IV* be the standard initial value, given a message M, the message $M' = M \oplus \Delta M$, with another initial value $IV' = IV \oplus (\Delta H_0, \Delta H_1, \Delta H_2)$ is the pseudo-preimage of M, i.e. Luffa-256(IV, M)=Luffa-256(IV', M'). There are only 5 bits different between *IV* and IV', which is minimum, when the message difference $\Delta M = (2^i, 2^i, 0, 0, 0, 0, 0, 0)$, $(0, 2^i, 2^i, 0, 0, 0, 0, 0)$, $(0, 0, 2^i, 2^i, 0, 0, 0, 0)$, $(0, 0, 0, 2^i, 2^i, 0, 0, 0)$ or $(0, 0, 0, 0, 2^i, 2^i, 0, 0)$ for $(0 \leq i < 32)$.

Let the input difference be (ΔH_0, ΔH_1, ΔH_2, ΔH_3, ΔM), and the output difference of *MI* be $(\mathbb{0}, \mathbb{0}, \mathbb{0}, \mathbb{0})$ for Luffa-384 such that

$$\begin{aligned}
4 \circ \Delta H_0 \oplus 6 \circ \Delta H_1 \oplus 6 \circ \Delta H_2 \oplus 7 \circ \Delta H_3 \oplus \Delta M &= \mathbb{0},\\
7 \circ \Delta H_0 \oplus 4 \circ \Delta H_1 \oplus 6 \circ \Delta H_2 \oplus 6 \circ \Delta H_3 \oplus 2\Delta M &= \mathbb{0},\\
6 \circ \Delta H_0 \oplus 7 \circ \Delta H_1 \oplus 4 \circ \Delta H_2 \oplus 6 \circ \Delta H_3 \oplus 4\Delta M &= \mathbb{0},\\
6 \circ \Delta H_0 \oplus 6 \circ \Delta H_1 \oplus 7 \circ \Delta H_2 \oplus 4 \circ \Delta H_3 \oplus 8\Delta M &= \mathbb{0}.
\end{aligned}$$

By the system of equations, we can deduce $\Delta H_0 = 8 \circ \Delta M$, $\Delta H_1 = 0xa \circ \Delta M$, $\Delta H_2 = 8 \circ \Delta M$ and $\Delta H_3 = 0xf \circ \Delta M$. There is a 7-bit difference in the initial values when $\Delta M = (2^i, 0, 2^i, 2^i, 0, 0, 0, 2^i)(0 \leq i < 32)$. The message $M' = M \oplus \Delta M$ with $IV' = IV \oplus (\Delta H_0, \Delta H_1, \Delta H_2, \Delta H_3)$ is the pseudo-preimage of the given message M, that is to say Luffa-384(IV, M)=Luffa-384(IV', M').

Given the input difference of Luffa-512 (ΔH_0, ΔH_1, ΔH_2, ΔH_3, ΔH_4, ΔM) and the output difference of *MI* $(\mathbb{0}, \mathbb{0}, \mathbb{0}, \mathbb{0}, \mathbb{0})$, we can compute that $\Delta H_0 = 0xbe \circ \Delta M$,

$\Delta H_1 = 0x3c \circ \Delta M$, $\Delta H_2 = 0x25 \circ \Delta M$, $\Delta H_3 = 0x17 \circ \Delta M$ and $\Delta H_4 = 0x75 \circ \Delta M$ from the following equations.

$$\begin{aligned}
0xf \circ \Delta H_0 \oplus 0x8 \circ \Delta H_1 \oplus 0xa \circ \Delta H_2 \oplus 0xa \circ \Delta H_3 \oplus 0x8 \circ \Delta H_4 \oplus \Delta M &= \mathbb{0},\\
0x8 \circ \Delta H_0 \oplus 0xf \circ \Delta H_1 \oplus 0x8 \circ \Delta H_2 \oplus 0xa \circ \Delta H_3 \oplus 0xa \circ \Delta H_4 \oplus 0x2\Delta M &= \mathbb{0},\\
0xa \circ \Delta H_0 \oplus 0x8 \circ \Delta H_1 \oplus 0xf \circ \Delta H_2 \oplus 0x8 \circ \Delta H_3 \oplus 0xa \circ \Delta H_4 \oplus 0x4\Delta M &= \mathbb{0},\\
0xa \circ \Delta H_0 \oplus 0xa \circ \Delta H_1 \oplus 0x8 \circ \Delta H_2 \oplus 0xf \circ \Delta H_3 \oplus 0x8 \circ \Delta H_4 \oplus 0x8\Delta M &= \mathbb{0},\\
0x8 \circ \Delta H_0 \oplus 0xa \circ \Delta H_1 \oplus 0xa \circ \Delta H_2 \oplus 0x8 \circ \Delta H_3 \oplus 0xf \circ \Delta H_4 \oplus 0x10\Delta M &= \mathbb{0}.
\end{aligned}$$

When $\Delta M = (0, 2^i, 2^i, 2^i, 0, 2^i, 0, 0)$, $(2^i, 2^i, 0, 0, 2^i, 0, 0, 2^i)$ or $(0, 0, 2^i, 0, 0, 2^i, 2^i, 2^i)$ for $(0 \le i < 32)$, the number of bits with difference in the initial value is least, which is 12.

Table 1 shows a pseudo-second-preimage example for the message M_0 = (0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa, 0xaaaaaaaa).

From the above description, only a few bits are needed to be changed to get a collision pair or the second-preimage for a given message. It is obvious that we can directly construct the forgery attack on NMAC based on Luffa in the related key case, for the NMAC replaces the fixed IV in hash function with a secret key[1]. The NMAC function, on input message M and a pair of independent keys (K_1, K_2), is defined as:

$$\mathrm{NMAC}_{(K_1,K_2)}(M) = H(K_1, H(K_2, M)).$$

When H is the Luffa hash function, a forgery message $M \oplus \Delta M$ with the same NMAC value as the message M in the related key case is given:

$$Luffa(K_1, Luffa(K_2, M)) = Luffa(K_1, Luffa(K_2 \oplus \Delta IV, M \oplus \Delta M)).$$

Where ΔM and ΔIV satisfy $MI(\Delta IV, \Delta M) = 0$.

4 The Pseudo-Preimage Attack on Luffa

For Luffa-256, given a hash value Z_0, the adversary can compute a pseudo-preimage with the following process. An example is shown in Table 2 with $Z_0 = \mathbb{0}$.

1. Select Y_0, Y_1 arbitrary, and get $Y_3 = Z_0 \oplus Y_0 \oplus Y_1$.
2. Compute $X_0 = Q_0^{-1}(Y_0)$, $X_1 = Q_1^{-1}(Y_1)$, $X_2 = Q_2^{-1}(Y_2)$.
3. Because the message $M = \mathbb{0}$ for the blank iteration, the adversary can compute $MI^{-1}(X_0, X_1, X_2)$ as follows,

$$\begin{pmatrix} H_0 \\ H_1 \\ H_2 \end{pmatrix} = \begin{pmatrix} 0x3, 0x2, 0x2 \\ 0x2, 0x3, 0x2 \\ 0x2, 0x2, 0x3 \end{pmatrix}^{-1} \circ \begin{pmatrix} X_0 \\ X_1 \\ X_2 \end{pmatrix}.$$

4. For the chaining variables H_0, H_1, H_2, the adversary can obtain $X_0 = Q_0^{-1}(H_0)$, $X_1 = Q_1^{-1}(H_1)$, $X_2 = Q_2^{-1}(H_2)$.
5. For (X_0, X_1, X_2), the adversary computers (IV_1', IV_2', M) with the fixed IV_0 by the following equations,

$$\begin{pmatrix} IV_1' \\ IV_2' \\ M \end{pmatrix} = \begin{pmatrix} 0x2, 0x2, 0x1 \\ 0x3, 0x2, 0x2 \\ 0x2, 0x3, 0x4 \end{pmatrix}^{-1} \circ \begin{pmatrix} X_0 \oplus (3 \circ IV_0) \\ X_1 \oplus (2 \circ IV_0) \\ X_2 \oplus (2 \circ IV_0) \end{pmatrix}.$$

6. Output (IV_0, IV_1', IV_2', M) which is the pseudo-preimage of Z_0, i.e.,

$$\text{Luffa-256}(IV', M) = Z_0.$$

There are $w-1$ 256-bit words changed of the initial value with w 256-bit words.

For Luffa-384, the hash value consists of Z_0 cascaded with the 128 most significant bits of Z_1, and

$$Z_1 = Z_{1,0} \| Z_{1,1} \| Z_{1,2} \| Z_{1,3} \| Z_{1,4} \| Z_{1,5} \| Z_{1,6} \| Z_{1,7},$$

where $Z_{1,i}$ for $0 \leq i < 8$ are 32-bit words. The adversary randomly chooses (H_0, H_1, H_2), and gets $H_3 = H_0 \oplus H_1 \oplus H_2 \oplus Z_0$, computes Z_1' using the finalization function.

If the equation $Z_{1,0}' \| Z_{1,1}' \| Z_{1,2}' \| Z_{1,3}'$ = $Z_{1,0} \| Z_{1,1} \| Z_{1,2} \| Z_{1,3}$ holds, let $Y_0 = H_0, Y_1 = H_1, Y_2 = H_2$. The adversary can now compute $(IV_0', IV_1', IV_2', IV_3')$ and message M_0 which has the same hash value $Z_0 \| Z_{1,0} \| Z_{1,1} \| Z_{1,2} \| Z_{1,3}$, using the similar method with Luffa-256. The total complexity is 2^{128} iteration computations.

For Luffa-512, the complexity is 2^{255} using a similar attack.

5 Improved Pseudo-Preimage Attacks on Luffa-384/512

In this section, we introduce an algorithm to improve the pseudo-preimage attack on Luffa-384/512 by the extended generalized birthday attack which is used to solve a system of equations, proposed by Schnorr [6]. The k-dimensional generalization of the birthday problem is, given k lists $L_0, L_1, \ldots, L_{k-1}$ independently at random from $\{0,1\}^n$, to find k elements $x_i \in L_i$ for $0 \leq i \leq k-1$ such that $x_0 \oplus x_1 \oplus \cdots \oplus x_{k-1} = 0$. Wagner's algorithm [7] builds a binary tree starting from the input lists $L_0, L_1, \ldots, L_{k-1}$. The time complexity and data complexity are both $t \cdot 2^{\frac{n}{1+t}}$, where $t = \lfloor \log_2 k \rfloor$.

5.1 The Extended Generalized Birthday Attack

We give a brief description of Wagner's generalized birthday attack in the following.
Wagner's Algorithm.

1. The adversary constructs 2^t sets S_0^0, S_1^0, …, $S_{2^t-1}^0$, where $t = \lfloor \log_2 k \rfloor$, $S_i^0 = \{ x_i^j \mid x_i^j \in L_i, j = 0, 1, \ldots, 2^{\frac{n}{1+t}} - 1\}$ for $1 \leq i < 2^t - 1$ and $S_{2^t-1}^0 = \{ x_{2^t-1}^j \oplus x_{2^t} \oplus \cdots \oplus x_k \mid x_{2^t-1}^j \in L_{2^t-1}, j = 0, 1, \ldots, 2^{\frac{n}{1+t}} - 1\}$, where $x_l \in L_l$ for $l = 2^t, \ldots, k-1$.
2. The adversary searches $2^{\frac{n}{1+t}}$ element pairs $x_{2i}^j \in S_{2i}^0$, $x_{2i+1}^k \in S_{2i+1}^0$ with the same low $\frac{n}{1+t}$ bits by the birthday attack. Construct 2^{t-1} new sets S_i^1, $i = 0, 1, \ldots, 2^{t-1} - 1$, where S_i^1= $\{$ $x_{2i}^j \oplus x_{2i+1}^k \mid$ the low $\frac{n}{1+t}$ bits are zeros$\}$.
3. For $m = 2$ to $t-1$, the adversary searches $2^{\frac{n}{1+t}}$ pairs $x_{2i}^j \in S_{2i}^{m-1}$ and $x_{2i+1}^k \in S_{2i+1}^{m-1}$ with the m-th low $\frac{n}{1+t}$ bits same. Construct 2^{t-m} new sets S_i^m, $i = 0, 1, \ldots, 2^{t-m} - 1$, where S_i^m= $\{$ $x_{2i}^j \oplus x_{2i+1}^k \mid$ the low $m \cdot \frac{n}{1+t}$ bits are zeros$\}$.
4. The adversary searches a pair $x_0^j \in S_0^{t-1}$, $x_1^k \in S_1^{t-1}$, s.t. $x_0^j \oplus x_1^k = 0$.

The above algorithm can find one solution $x_0, x_1, \ldots, x_{k-1}$ such that $x_0 \oplus x_1 \cdots \oplus x_{k-1} = 0$ with time complexity and data complexity being both $t \cdot 2^{\frac{n}{1+t}}$.

Now consider the solution to the following two equations instead of one equation.

$$f_1(x_1) \oplus f_2(x_2) \oplus \cdots \oplus f_k(x_k) = c_1, \tag{1}$$

$$g_1(x_1) \oplus g_2(x_2) \oplus \cdots \oplus g_k(x_k) = c_2, \tag{2}$$

where f_i and g_i $(1 \leq i \leq k)$ are random functions, $f_i : 2^m \to 2^{n_1}$, $g_i : 2^m \to 2^{n_2}$. The equations (1) and (2) can be solved together by the extended generalized birthday attack [6] described in the following.

It is easy to construct the following equation from equations (1) and (2):

$$(f_1(x_1) \| g_1(x_1)) \oplus (f_2(x_2) \| g_2(x_2)) \oplus \cdots \oplus (f_k(x_k) \| g_k(x_k)) = c_1 \| c_2. \tag{3}$$

For the new equation (3), the Wagner's algorithm can be applied to obtain $x_1, \ldots, x_k$. The data and time complexity is $t \cdot 2^{\frac{n_1+n_2}{1+t}}$, where $t = \lfloor \log_2 k \rfloor$ and $m \geq \frac{n_1+n_2}{1+t}$.

It is clear that, the algorithm can be extended to solve more equations.

$$\begin{aligned}
f_1^{(1)}(x_1) \oplus f_2^{(1)}(x_2) \oplus \cdots \oplus f_k^{(1)}(x_k) &= c_1, \\
f_1^{(2)}(x_1) \oplus f_2^{(2)}(x_2) \oplus \cdots \oplus f_k^{(2)}(x_k) &= c_2, \\
&\vdots \\
f_1^{(l)}(x_1) \oplus f_2^{(l)}(x_2) \oplus \cdots \oplus f_k^{(l)}(x_k) &= c_l,
\end{aligned}$$

where $f_j^{(i)} : 2^m \to 2^{n_i}$ are random functions,$0 \leq i \leq l$, and $0 \leq j \leq k$. The data and time complexity is $t \cdot 2^{\frac{n_1+n_2+\cdots+n_t}{1+t}}$, where $t = \lfloor \log_2 k \rfloor$ and $m \geq \frac{n_1+n_2+\cdots+n_t}{1+t}$.

5.2 The Improved Pseudo-Preimage Attack on Luffa-384

Let $(H_0, H_1, H_2, H_3, \mathbb{0})$ be the input of the last blank iteration function, and (X_0, X_1, X_2, X_3) be the output of its MI. The hash value is $Z_0 \| \overline{Z_1}$, where $\overline{Z_1} = Z_{1,0} \| Z_{1,1} \| Z_{1,2} \| Z_{1,3}$. Then

$$h_{128}(Q_0(X_0) \oplus Q_1(X_1) \oplus Q_2(X_2) \oplus Q_3(X_3)) = \overline{Z_1}. \tag{4}$$

From the message injection function MI, we know that $(H_0, H_1, H_2, H_3)^T = A_{4\times4}^{-1}(X_0, X_1, X_2, X_3)^T$, where $A_{4\times4}$ is the first 4 column vectors of the matrix $A_{4\times5}$, i.e.,

$$A_{4\times4} = \begin{pmatrix} 0x4, 0x6, 0x6, 0x7 \\ 0x7, 0x4, 0x6, 0x6 \\ 0x6, 0x7, 0x4, 0x6 \\ 0x6, 0x6, 0x7, 0x4 \end{pmatrix}.$$

It's inverse matrix is

$$A_{4\times4}^{-1} = \begin{pmatrix} 0x20, 0x43, 0x84, 0x11 \\ 0x11, 0x20, 0x43, 0x84 \\ 0x84, 0x11, 0x20, 0x43 \\ 0x43, 0x84, 0x11, 0x20 \end{pmatrix}.$$

From $H_0 \oplus H_1 \oplus H_2 \oplus H_3 = Z_0$, we can prove that,

$$X_0 \oplus X_1 \oplus X_2 \oplus X_3 = Z_0', \tag{5}$$

where $Z_0' = 0x3 \circ Z_0$.

Obviously, it is necessary for us to find the solution (X_0, X_1, X_2, X_3) to make equations (4) and (5) hold together. We search the solution by the extended generalized birthday attack and some specific properties of Luffa. The algorithm is as follows.

1. The adversary constructs four sets such that,

$$\begin{aligned} S_0 &= \{X_0 \mid X_0 \in \{0,1\}^n, l_{192}(X_0) = c_0\}, \\ S_1 &= \{X_1 \mid X_1 \in \{0,1\}^n, l_{192}(X_1) = c_0 \oplus l_{192}(Z_0')\}, \\ S_2 &= \{X_2 \mid X_2 \in \{0,1\}^n, l_{192}(X_2) = c_1\}, \\ S_3 &= \{X_3 \mid X_3 \in \{0,1\}^n, l_{192}(X_3) = c_1\}, \end{aligned}$$

   where c_0, c_1 are two 192-bit constants, and each set includes 2^{64} elements. It is clear that,

$$l_{192}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) = l_{192}(Z_0'),$$

   where $X_i \in S_i$ for $0 \leq i \leq 3$.
2. The adversary searches the solution (X_0, X_1, X_2, X_3) satisfying the following two equations by the extended generalized birthday attack.

$$h_{64}(X_0 \oplus X_1 \oplus X_2 \oplus X_3) = h_{64}(Z_0'),$$

$$h_{128}(Q_0(X_0) \oplus Q_1(X_1) \oplus Q_2(X_2) \oplus Q_3(X_3)) = \overline{Z_1},$$

   where $X_i \in S_i, i = 0,1,2,3$. It is clear that, The solution (X_0, X_1, X_2, X_3) guarantees the equations (4) and (5) hold together.
3. For (X_0, X_1, X_2, X_3), the adversary can calculate $(IV_0, IV_1', IV_2', IV_3')$ and the message M, and get the pseudo-preimage using the similar pseudo-preimage attack on Luffa-256.

There are 2^{64} Q_0, Q_1, Q_2, Q_3 computations and 2^{64} table lookups in the above algorithm. So the total complexity is about 2^{64} iteration computations and 2^{67} bytes memory.

5.3 The Improved Pseudo-Preimage Attack on Luffa-512

For Luffa-512, let $(H_0, H_1, H_2, H_3, H_4, \mathbb{0})$ be the input of the last blank iteration function, and $(X_0, X_1, X_2, X_3, X_4)$ be the output of MI. Then

$$Q_0(X_0) \oplus Q_1(X_1) \oplus Q_2(X_2) \oplus Q_3(X_3) \oplus Q_4(X_4) = Z_1. \tag{6}$$

For the message injection function MI, we know that, $(H_0, H_1, H_2, H_3, H_4)^T = A_{5\times5}^{-1}(X_0, X_1, X_2, X_3, X_4)^T$, where $A_{5\times5}$ is the first 5 column vectors of the matrix $A_{5\times6}$, i.e.,

$$A_{5\times5} = \begin{pmatrix} 0xf, 0x8, 0xa, 0xa, 0x8 \\ 0x8, 0xf, 0x8, 0xa, 0xa \\ 0xa, 0x8, 0xf, 0x8, 0xa \\ 0xa, 0xa, 0x8, 0xf, 0x8 \\ 0x8, 0xa, 0xa, 0x8, 0xf \end{pmatrix}.$$

Its inverse matrix is

$$A_{5\times5}^{-1} = \begin{pmatrix} 0xc7, 0x8b, 0xf4, 0xf4, 0x8b \\ 0x8b, 0xc7, 0x8b, 0xf4, 0xf4 \\ 0xf4, 0x8b, 0xc7, 0x8b, 0xf4 \\ 0xf4, 0xf4, 0x8b, 0xc7, 0x8b \\ 0x8b, 0xf4, 0xf4, 0x8b, 0xc7 \end{pmatrix}.$$

Since $H_0 \oplus H_1 \oplus H_2 \oplus H_3 \oplus H_4 = Z_0$, we obtain

$$X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4 = Z_0', \tag{7}$$

where $Z_0' = 0xf \circ Z_0$.

We can search a solution to equations (6) and (7) by the extended generalized birthday attack and some specific properties of Luffa.

1. The adversary constructs four sets such that,

$$\begin{aligned} S_0 &= \{X_0 \mid X_0 \in \{0,1\}^n, l_{128}(X_0) = c_0\}, \\ S_1 &= \{X_1 \mid X_1 \in \{0,1\}^n, l_{128}(X_1) = c_0 \oplus l_{128}(Z_0')\}, \\ S_2 &= \{X_2 \mid X_2 \in \{0,1\}^n, l_{128}(X_2) = c_1\} \\ S_3 &= \{(X_3, X_4) \mid X_3, X_4 \in \{0,1\}^n, l_{128}(X_3 \oplus X_4) = c_1\}, \end{aligned}$$

   where c_0, c_1 are two 128-bit constants, and each set includes 2^{128} elements. It is clear that,

$$l_{128}(X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4) = l_{128}(Z_0').$$

2. The adversary searches a solution $(X_0, X_1, X_2, X_3, X_4)$ satisfying the following two equations by the extended generalized birthday attack.

$$h_{128}(X_0 \oplus X_1 \oplus X_2 \oplus X_3 \oplus X_4) = h_{128}(Z_0'),$$

$$Q_0(X_0) \oplus Q_1(X_1) \oplus Q_2(X_2) \oplus Q_3(X_3) \oplus Q_4(X_4) = \bar{Z_1},$$

   where $X_i \in S_i, i = 0, 1, 2$ and $(X_3, X_4) \in S_3$. It is clear that, The solution $(X_0, X_1, X_2, X_3, X_4)$ guarantees equations (6) and (7) hold.
3. For $(X_0, X_1, X_2, X_3, X_4)$, the adversary can calculate $(IV_0, IV_1', IV_2', IV_3', IV_4')$ and the message M, and get the pseudo-preimage using the similar pseudo-preimage attack on Luffa-256.

The total complexity is about 2^{128} iteration computations and 2^{132} bytes memory.

6 Conclusion

In this paper, we give pseudo-collision, pseudo-second-preimage and pseudo-preimage attacks on Luffa, one of the second round candidates of SHA-3. For any given output of the message injection function MI, it is easy to get the input to MI using the inverse operation of MI. So we can find pseduo-collisions and pseudo-second-preimages easily for Luffa by applying the MI property. It is noted that, the pseudo-collisions and pseudo-second-preimages only with a few different bits are easily searched. The attack can be directly converted to a forgery attack on NMAC with related keys.

Especially, we focus on the the pseudo-preimage attack on Luffa. For Luffa-224/256, the attack can find the the pseudo-preimage only with 2 iteration computations. It takes about 2^{64} iteration computations and 2^{67} bytes memory to search a pseudo-preimage for Luffa-384, and search a pseudo-preimage with 2^{128} iteration computations and 2^{132} bytes memory for Luffa-512 with the extended generalized birthday attack.

Acknowledgements. We would like to thank the anonymous reviewers for their valuable comments on this paper. This work is supported by the National "973" Program of China (Grant No.2007CB807902) and Tsinghua University Initiative Scientific Research Program (2009THZ01002). For the second author, a part of this work was done while funded by EPSRC EP/C538285/1, by BT(as BT Chair of Information Security) and while visiting Tsinghua University funded by 2007CB807902.

References

1. Bellare, M., Canetti R., Krawczyk H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Cannière, C. D., Sato, H., Watanabe, D.: Hash function Luffa Specification. http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Luffa.zip. (2008)
3. Contini, S., Yin, Y. L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.): ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
4. Fouque, P.-A., Leurent, G., Nguyen, P. Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
5. Kelsey, J., Schneier, B.: Second Preimages on n-Bit Hash Functions for Much Less than 2^n Work. In: Cramer, R. (ed.): EUROCRYPT 2005, LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
6. Schnorr, C.P., Enhancing the Security of Perfect Blind DL-Signatures. Information Sciences. 176(i10), pp.1305-1320 (2006)
7. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.): CRYPTO 2002, LNCS, vol 2442, pp. 288–304. Springer, Heidelberg (2002)
8. Wang, X., Lai, X., Feng, D. et al.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.): Eurocrypt 2005, LNCS, vol 3494, pp. 1–18. Springer, Heidelberg (2005)
9. Wang, X., Yu, H., Yin, Y. L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.): CRYPT 2005, LNCS, vol 3621, pp. 1–16. Springer, Heidelberg (2005)
10. Wang, X., Yin, Y. L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.): CRYPT 2005, LNCS, vol 3621, pp. 17–36. Springer, Heidelberg (2005)

11. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.): Eurocrypt 2005, LNCS, vol 3494, pp. 19–35. Springer, Heidelberg (2005)
12. Yu, H., Wang, G., Zhang, G. et al.: The second-preimage attack on MD4. In: Desmedt, Y. et al. (eds.): CANS 2005, LNCS, vol 3810, pp. 1–12. Springer, Heidelberg (2005)

Appendix

In the appendix,we give two examples for the pseudo-second-preimage and pseudo-preimage.

IV_0	0x6d251e69	0x44b051e0	0x4eaa6fb4	0xdbf78465	0x6e292011	0x90152df4	0xee058139	0xdef610bb
IV_1	0xc3b44b95	0xd9d2f256	0x70eee9a0	0xde099fa3	0x5d9b0557	0x8fc944b3	0xcf1ccf0e	0x746cd581
IV_2	0xf7efc89d	0x5dba5781	0x04016ce5	0xad659c05	0x0306194f	0x666d1836	0x24aa230a	0x8b264ae7
M_0	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa
IV_0'	0x6d251e6**8**	0x44b051e0	0x4eaa6fb**5**	0xdbf7846**4**	0x6e292011	0x90152df4	0xee058139	0xdef610bb
IV_1'	0xc3b44b95	0xd9d2f256	0x70eee9a0	0xde099fa**2**	0x5d9b0557	0x8fc944b3	0xcf1ccf0e	0x746cd581
IV_2'	0xf7efc89d	0x5dba578**0**	0x4016ce5	0xad659c05	0x306194f	0x666d1836	0x24aa230a	0x8b264ae7
M_0'	0xaaaaaaa**b**	0xaaaaaaa**b**	0xaaaaaaaa	aaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa	0xaaaaaaaa
X_0	0xe6333b1e	0x96d8e9f6	0x24d83129	0x6aa44be3	0x4da482a5	0x0a0bbb57	0x3d1e5ae2	0x71efd72c
X_1	0x48a26ee2	0xa110e0ea	0x1a9cb73d	0xc5f0fa8f	0xd4bc0d49	0x15d7d210	0x1c0714d5	0xdb751216
X_2	0x7cf9edea	0x2578453d	0xc4d998d2	0xb69cf929	0x208bbbfb	0x56d9243f	0xf7b1f8d1	0x243f8d70

Table 1. A Pseudo-second-preimage for Luffa-256

IV_0	0x6d251e69	0x44b051e0	0x4eaa6fb4	0xdbf78465	0x6e292011	0x90152df4	0xee058139	0xdef610bb
IV_1'	0x6a366118	0x3ee79df6	0x39643181	0x60793777	0x8ddc9066	0x1d50cebd	0xb1cfd39b	0x967da4e4
IV_2'	0x9622ac99	0xb752bbbb	0xd256db58	0x73db6cac	0x9ae49b27	0xeb1666b4	0x805027ed	0x8176bfc6
M	0x7c08aa09	0x52f9e2bf	0x27ce6bb9	0x11af8970	0x22c8478d	0x9eebde0e	0x78ae77ef	0xdafc7fa8
H_0	0xd42f102f	0x94316735	0xec5bb8a2	0xceb338ee	0x6d35036f	0x85d4ba8c	0xc9a85c96	0xed839a52
H_1	0x70238338	0x4461e9a7	0xa3012529	0xb6a10e0f	0xdfdf5bc0	0x2fd50d38	0xe98ddd20	0xf90f4fe9
H_2	0xe0d87b07	0x5704423f	0xb8ba00ed	0xeaa52759	0x8bc1b72b	0xc5720d53	0x41cde665	0x1288c8fc
Z_0	0	0	0	0	0	0	0	0

Table 2. A Pseudo-preimage for Luffa-256