LNCS 3810

Yvo G. Desmedt
Huaxiong Wang
Yi Mu
Yongqing Li (Eds.)

Cryptology and Network Security

4th International Conference, CANS 2005
Xiamen, China, December 2005
Proceedings

Springer

Yvo G. Desmedt Huaxiong Wang
Yi Mu Yongqing Li (Eds.)

Cryptology and Network Security

4th International Conference, CANS 2005
Xiamen, China, December 14-16, 2005
Proceedings

Springer

Volume Editors

Yvo G. Desmedt
University College London, Department of Computer Science
Gower Street, London WC1E 6BT, UK
E-mail: y.desmedt@cs.ucl.ac.uk

Huaxiong Wang
Macquarie University, Department of Computing
NSW 2109, Australia
E-mail: hwang@ics.mq.edu.au

Yi Mu
University of Wollongong, School of Information Technology and Computer Science
Wollongong, NSW 2522, Australia
E-mail: ymu@uow.edu.au

Yongqing Li
Fujian Normal University, School of Mathematics and Computer Science
Fujian, Fuzhou 350007, China
E-mail: yqli@fjnu.edu.cn

Library of Congress Control Number: 2005936808

CR Subject Classification (1998): E.3, D.4.6, F.2.1, C.2, J.1, K.4.4, K.6.5

ISSN 0302-9743
ISBN-10 3-540-30849-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-30849-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11599371 06/3142 5 4 3 2 1 0

Preface

The 4th International Conference on Cryptology and Network Security (CANS 2005) was held in Xiamen, Fujian Province, China, December 14–16, 2005. The conference was sponsored by the Fujian Normal University and Fujian Digital Certificate Authority Co. Ltd and was organized in cooperation with the International Association for Cryptologic Research (IACR).

The first International Workshop on Cryptology and Network Security was in Taipei, Taiwan, 2001. The second one was in San Francisco, California, USA, September 26–28, 2002, and the third in Miami, Florida, USA, September 24–26, 2003. CANS 2005 was the first CANS with proceedings published in the *Lecture Notes in Computer Science* series by Springer.

The Program Committee received 118 submissions, and accepted 28 papers from which 1 withdrew and thus 27 papers were included in the proceedings. The reviewing process took eight weeks, each paper was carefully evaluated by at least three members from the Program Committee. We appreciate the hard work of the members of the Program Committee and external referees who gave many hours of their valuable time. Thanks to Carl Ellison, Goce Jakimoski, Bart Preneel, Yongge Wang, Christopher Wolf and Shouhuai Xu, who acted as the shepherds of 6 papers included in the proceedings.

In addition to the contributed papers, there were two invited talks: Wenbo Mao spoke on "Research Issues in Network Security" — a practical viewpoint; and Matt Franklin on "Research Issues in Network Security" — a foundations viewpoint.

The best paper award was given to Hongbo Yu, Gaoli Wang, Guoyan Zhang and Xiaoyun Wang for their joint paper: The Second-Preimage Attack on MD4.

We would like to thank all the people involved in organizing this conference. In particular we would like to thank the Chair of the Organizing Committee, Xu Li, and people from the School of Mathematics and Computer Science, Fujian Normal University, for their time and efforts, as well as Vijayakrishnan Pasupathinathan and Qingsong Ye for their excellent work on maintaining the submission/reviewing software.

December 2005

Yvo Desmedt
Huaxiong Wang
Yi Mu
Yongqing Li

4TH International Conference on Cryptology and Network Security (CANS 2005)

Sponsored by
Fujian Normal University
Fujian Digital Certificate Authority Co. Ltd.

In Cooperation with
The International Association for Cryptologic Research (IACR)

General Chairs

Yongqing Li	Fujian Normal University, China
Yi Mu	University of Wollongong, Australia

Program Chairs

Yvo G. Desmedt	University College London, UK & Florida State Univ., USA
Huaxiong Wang	Macquarie University, Australia

Program Committee

Farooq Anjum	Telcordia, USA
Amos Beimel	Ben Gurion University, Israel
John Black	University of Colorado, USA
Carlo Blundo	University of Salerno, Italy
Jung Hee Cheon	Seoul Natl. Univ., South Korea
Cunsheng Ding	Hong Kong Univ. Sci. Tech., China
Carl Ellison	Microsoft, USA
Helena Handschuh	Gemplus, France
Thomas Johansson	University of Lund, Sweden
Antoine Joux	Université de Versailles, France
Kaoru Kurosawa	Ibaraki University, Japan
Xuejia Lai	Shanghai Jiao Tong University, China
Tanja Lange	Technical University of Denmark, Denmark
Pil Joong Lee	Pohang University, South Korea
Arjen Lenstra	Lucent, USA & Tech. Univ. Eindhoven, The Netherland
Radia Perlman	Sun Microsystems, USA
Josef Pieprzyk	Macquarie University, Australia
David Pointcheval	École Normale Supérieure, France
Bart Preneel	Katholieke Universiteit Leuven, Belgium

C. Pandu Rangan	Indian Institute of Technology, India
Kazue Sako	NEC, Japan
Berry Schoenmakers	Tech. Univ. Eindhoven, The Netherlands
Willy Susilo	University of Wollongong, Australia
Xiaoyun Wang	Shandong Univ. Australia & Tsinghua Univ., China
Yongge Wang	University of North Carolina, USA
Duncan Wong	City University of Hong Kong, China
Susanne Wetzel	Stevens Inst. of Technology, USA
Chuan-Kun Wu	Australian Natl. Univ., Australia & SKLOIS, China
Chaoping Xing	National Univ. of Singapore, Singapore
Shouhuai Xu	University of Texas, USA
Sung-Ming Yen	National Central University, Taiwan

Organizing Committee

Li Xu (Chair)	Fujian Normal University, China
Vijayakrishnan Pasupathinathan	Macquarie University, Australia
Xuan Hui Yan	Fujian Normal University, China
Zhi Qiang Yao	Fujian Normal University, China
Qingsong Ye	Macquarie University, Australia
Sheng Yuan Zhang	Fujian Normal University, China

External Referees

Michel Abdalla
Masayuki Abe
Roberto Avanzi
Joonsang Baek
Bruno Blanchet
An Braeken
Benoit Chevallier-Mames
Kuo-Zhe Chiou
Kookrae Cho
Mathieu Ciet
Christophe Clavier
Scott Contini
Ingemar Cox
Nora Dabbous
Serge Fehr
Pierre-Alain Fouque
Chandana Gamage
Rob Granger
Goichiro Hanaoka
Swee-Huay Heng
Shoichi Hirose
Yong Ho Hwang
Sortiris Ioannidis
Goce Jakimoski
Shaoquan Jiang
Masaru Kamada
Charlie Kaufman
Tom Kevenaar
Hyungshin Kim
Seungjoo Kim
Yongdae Kim
Takeshi Koshiba
Taekyung Kwon
Heung-Kyu Lee
Jung Wook Lee
Hsi-Chung Lin
Pin Lin
Phil MacKenzie
Kengo Mori
Kathleen Moriarty
Kenny Nguyen-Qk
Svetla Nikova
M. Paramasivam
Duong Hieu Phan
Raphael C.-W. Phan
Michael Quisquater
Nicholas Sheppard
Jong Hoon Shin
Igor Shparlinski
Martijn Stam
Ron Steinfeld
Po-Chyi Su
Dongvu Tonien
Isamu Teranishi
Duong Quang Viet
Guilin Wang
Liming Wang
Enav Weinreb
Christopher Wolf
Tao Xu
Yeon Hyeong Yang
Jeong Il Yoon

Table of Contents

The Second-Preimage Attack on MD4

Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang*

School of Mathematics and System Sciences,
Shandong University, Jinan 250100, China
yhb@mail.sdu.edu.cn, xywang@sdu.edu.cn

Abstract. In Eurocrypt'05, Wang et al. presented new techniques to find collisions of Hash function MD4. The techniques are not only efficient to search for collisions, but also applicable to explore the second-preimage of MD4. About the second-preimage attack, they showed that a random message was a weak message with probability 2^{-122} and it only needed a one-time MD4 computation to find the second-preimage corresponding to the weak message. A weak message means that there exits a more efficient attack than the brute force attack to find its second-preimage. In this paper, we find another new collision differential path which can be used to find the second-preimage for more weak messages. For any random message, it is a weak message with probability 2^{-56}, and it can be converted into a weak message by message modification techniques with about 2^{27} MD4 computations. Furthermore, the original message is close to the resulting message (weak message), i.e, the Hamming weight of the difference for two messages is about 44.

Keywords: Hash function, collision differential path, second-preimage, weak message.

1 Introduction

In 1990[1], Rivest introduced the hash function MD4 which is the first dedicated hash function. After MD4, many hash functions such as MD5[2], HAVAL[3], RIPEMD [4], SHA-0[5], SHA-1[6], SHA-256[7] were designed subsequently.

For a hash function h with inputs x, x' and outputs y, y', three potential security properties should be satisfied:

1. **Preimage resistance:** for any pre-specified output y, it is computationally infeasible to find an input x such that $h(x) = y$.
2. **Second-preimage resistance:** for any input x, it is computationally infeasible to find another input x' such that $h(x) = h(x')$
3. **Collision resistance:** it is computationally infeasible to find any two distinct inputs x, x' with the same output, i.e., $h(x) = h(x')$.

* Supported by the National Natural Science Foundation of China(NSFC Grant No.90304009) and 973 Project(No.2004CB318000).

The original design purpose of MD4 is that there is no better collision attack than the birthday attack which should take about 2^{64} MD4 computations to find a collision, and no better attack than brute force attack which should take 2^{128} MD4 computations to find a preimage corresponding to a pre-specified hash-value or the second preimage corresponding to a given message. The existing attack reveals that MD4 fails to reach the designer's goals both on collision resistance and second-preimage resistance. In 1996, Dobbertin presented a successful attack on MD4 which find a collision with probability 2^{-22}[8]. In 1998, H.Doberrtin[9] showed that the first two (out of the total three) rounds of MD4 are not one-way. This means it is possible to find the preimage and the second-preimage for the first two rounds of MD4. Wang et al. [11] described a new kind of collision attack on the hash function MD4 and RIPEMD which is also applied to break MD5[10], HAVAL-128[14], SHA-0[12] and SHA-1[13]. Simultaneously, the collision attack on MD4 [11] can be used to explore the second-preimage attack on MD4, and the main results are as follows:

1. A random message is a weak message with probability 2^{-122}. For a weak message, it only needs a one-time MD4 computation to find a second-preimage of the resulting hash value.
2. Any message M can be modified with the basic message modification techniques. The resulting message M' is a weak message with probability 2^{-23}. M and M' are close and the Hamming weight of the difference for two messages is 50 on average.
3. Under the advanced message modification, any message M can be modified into M' which is a weak message with probability 2^{-2} to 2^{-6}. However, the Hamming weight of their difference increases quickly up to 110.

In this paper, we give a further research on the second-preimage attack on MD4. Our results are as follows:

1. We find a new differential path which is efficient to find more weak messages. Utilizing this path, any message M is a weak message with probability 2^{-56}. For a weak message, it only needs a one-time MD4 computation to find a second-preimage.
2. For any message, we apply message modification techniques to convert it into a weak message with 2^{27} MD4 computations, the Hamming weight for their difference is about 44.

The paper is organized as follows: In section 2, we describe MD4 details. In section 3, we give some basic properties of nonlinear round functions for MD4 and some notations. Our main results are introduced in section 4. We summarize the paper in section 5.

2 Description of MD4

The message digest algorithm MD4 takes a message of length less than 2^{64} bits and produces a 128-bit hash value. The input message is padded and then processed in 512-bit blocks by Damgard/Merkle iterative structure. Each iteration invokes a compression function which takes a 128-bit chaining value and a 512-bit message block and outputs another 128-bit chaining value. The initial chaining value (called IV) is a set of fixed constants, and the final chaining value is the hash value of the message.

MD4 has three rounds, and every round employs a round function. The three round functions are defined as follows:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z) \\ G(X,Y,Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\ H(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

Here X, Y, Z are 32-bit words. The operations of three functions are all bitwise. $\neg X$ is the bitwise complement of X, $\wedge$, $\oplus$ and $\vee$ are respectively the bitwise AND, XOR and OR.

Each round of the compression function repeats 16 similar step operations, and in each step, one of the four chaining variables a, b, c, d is updated. "$\lll s$" is the circularly left-shift by s bit positions and "$\ggg s$" is the circularly right-shift by s bit positions.

$$\begin{aligned} \phi_0(a,b,c,d,m_k,s) &= ((a + F(b,c,d) + m_k) \ mod \ 2^{32}) \lll s \\ \phi_1(a,b,c,d,m_k,s) &= ((a + G(b,c,d) + m_k + 0x5a827999) \ mod \ 2^{32}) \lll s \\ \phi_2(a,b,c,d,m_k,s) &= ((a + H(b,c,d) + m_k + 0x6ed9eba1) \ mod \ 2^{32}) \lll s \end{aligned}$$

The initial value for MD4 is defined as:

$$(a,b,c,d) = (0x67452301,\ 0xefcdab89,\ 0x98badcfe,\ 0x10325476)$$

MD4 Compression Function. For one 512-bit block M of the padded message $\overline{M}$, $M = (m_0, m_1, ..., m_{15})$, the compressing process is as follows:

1. Let (aa, bb, cc, dd) be input of compressing process for M. If M is the first message block to be hashed, (aa, bb, cc, dd) is selected as the initial value. Otherwise it is the output for compressing the previous message block. a, b, c, d are the chaining variables which are initialized by the initial values.
2. Perform the following 48 steps (three rounds):

```
For j=0, 1, 2
    For i=0, 1, 2, 3
        $a = \phi_j(a, b, c, d, w_{j,4i}, s_{j,4i})$
        $d = \phi_j(d, a, b, c, w_{j,4i+1}, s_{j,4i+1})$
        $c = \phi_j(c, d, a, b, w_{j,4i+2}, s_{j,4i+2})$
        $b = \phi_j(b, c, d, a, w_{j,4i+3}, s_{j,4i+3})$
```

$s_{j,4i+k}$ ($k = 0,\ 1,\ 2,\ 3$) are step-dependent constants. $w_{j,4i+k}$ is a message word. The details of the message order and shift positions can be referred to Table 3.

3. Add a, b, c and d respectively to the chaining variables in the input value.

$$\begin{aligned} aa &= (a + aa) \ mod \ 2^{32} \\ bb &= (b + bb) \ mod \ 2^{32} \\ cc &= (c + cc) \ mod \ 2^{32} \\ dd &= (d + dd) \ mod \ 2^{32} \end{aligned}$$

If M is the last message block, $H(\overline{M}) = aa||bb||cc||dd$ is the hash value for the message $\overline{M}$. Otherwise repeat the above compression function with the next 512-bit message block and (aa, bb, cc, dd) as inputs.

3 Preliminaries

3.1 Some Basic Conclusions of the Three Nonlinear Functions

The collision differential path and its sufficient conditions are closely related to the following properties of the three round functions.

Proposition 1. For the nonlinear function $f(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ in the first round, the following properties hold:

1. $f(x, y, z) = f(\neg x, y, z)$ if and only if $y = z$.
   $f(x, y, z) = x$ and $f(\neg x, y, z) = \neg x$ if and only if $y = 1$ and $z = 0$.
   $f(x, y, z) = \neg x$ and $f(\neg x, y, z) = x$ if and only if $y = 0$ and $z = 1$.

2. $f(x, y, z) = f(x, \neg y, z)$ if and only if $x = 0$.
   $f(x, y, z) = y$ and $f(x, \neg y, z) = \neg y$ if and only if $x = 1$.

3. $f(x, y, z) = f(x, y, \neg z)$ if and only if $x = 1$.
   $f(x, y, z) = z$ and $f(x, y, \neg z) = \neg z$ if and only if $x = 0$.

Proposition 2. For the nonlinear function $g(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$, the following properties hold:

1. $g(x, y, z) = g(\neg x, y, z)$ if and only if $y = z$.
   $g(x, y, z) = x$ and $g(\neg x, y, z) = \neg x$ if and only if $y = \neg z$.

2. $g(x, y, z) = g(x, \neg y, z)$ if and only if $x = z$.
   $g(x, y, z) = y$ and $g(x, \neg y, z) = \neg y$ if and only if $x = \neg z$.

3. $g(x, y, z) = g(x, y, \neg z)$ if and only if $x = y$.
   $g(x, y, z) = z$ and $g(x, y, \neg z) = \neg z$ if and only if $x = \neg y$.

Proposition 3. For the nonlinear function $h(x, y, z) = x \oplus y \oplus z$, the following properties hold:

1. $h(x, y, z) = \neg h(\neg x, y, z) = \neg h(x, \neg y, z) = \neg h(x, \neg y, z)$
2. $h(x, y, z) = h(\neg x, \neg y, z) = h(x, \neg y, \neg z) = h(\neg x, y, \neg z)$

Here, x, y, $z \in \{0, 1\}$ and $\neg$ is the bit complement operation.

3.2 Basic Notations

1. $M = (m_0, m_1, ..., m_{15})$ and $M' = (m'_0, m'_1, ..., m'_{15})$ represent two 512-bit messages.
2. a_i, d_i, c_i, b_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th, $(4i-1)$-th and $4i$-th steps for compressing M, where $1 \leq i \leq 16$.
3. a'_i, b'_i, c'_i, d'_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th, $(4i-1)$-th and $4i$-th steps for compressing M'.
4. $\Delta m_i = m'_i - m_i$ denotes the difference of two words m_i and m'_i. It is noted that Δm_i is an modular difference and not a XOR difference.
5. $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$ represent respectively the j-th bit of a_i, b_i, c_i, d_i, where the least significant bit is the 1-st bit, and the most significant bit is 32-nd bit.
6. $x_i[j]$, $x_i[-j]$ (x can be a, b, c, d) are the resulting values by only changing the $j-th$ bit of the word x_i. $x_i[j]$ is obtained by changing the j-th bit of x_i from 0 to 1. $x_i[-j]$ is obtained by changing the j-th bit of x_i from 1 to 0.
7. $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ is the value by changing j_1-th, j_2-th, ..., j_l-th bits of x_i. The " $+$ " sign means that the bit is changed from 0 to 1, and the " $-$ " sign means that the bit is changed from 1 to 0.

4 The Second-Preimage Attack on MD4

In this section, we describe a second-preimage attack to find more weak messages and the corresponding second-preimages. The collision differential path in [11] is efficient to find collisions of MD4, but it isn't efficient to find weak messages and second-preimages because the path has too many conditions. Our purpose is to find another collision differential path with fewer conditions which is easily used to find weak messages and second-preimages.

4.1 Constructing the Specific Collision Differential Path

In order to find such a path, we select $\Delta M = M' - M$ as:

$$M = (m_0, m_1, \ldots, m_{15})$$
$$\Delta M = (0, 0, 0, 0, e2^i, 0, \ldots, 0), where\ e = \pm 1\ and\ 0 \leq i \leq 31$$

We find a collision differential path when $e = 1$ and $i = 22$ with 62 variable conditions which are showed in Table 3.

The following description shows how to construct such a path. The main idea of constructing a valid collision differential path is to cancel out the propagations of message difference $\Delta m_4 = 2^{22}$ which occurs in step 5, 18 and 35 respectively.

1. In step 5, the message difference can cause the chaining variable difference $a_2[26]$.
2. In step 18, the message difference can be cancelled out by previous chaining variable difference or cause new difference $d_5[28]$.
3. In step 35, we select a chaining variable difference $\Delta c_8 = -2^{22}$ to cancel out the difference $\Delta m_4 = 2^{22}$, so we set $c_8' = c_8[-23]$.

In order to guarantee M and M' consist of a collision in step 35, the differences Δb_8, Δa_9, Δd_9 must remain zero. According to Proposition 3, the difference Δc_8 results in the nonzero differences Δa_9 and Δd_9. In order to cancel out these two differences, we set $d_8' = d_8[23]$. From $a_2[26]$, we get two simple lines reaching $d_8[23]$ and $c_8[23]$. They are expressed as follows:

$$a_2[26] \to a_3[-29,30] \to a_4[32] \to a_5[3] \to d_5[8] \to d_6[13] \to d_7[18] \to d_8[23]$$

$$a_2[26]\to a_3[-29]\to c_3[-8]\to c_4[-19]\to c_5[-28]\to c_6[5,-6]\to c_7[-14]\to c_8[-23]$$

In addition, the difference $a_2[26]$ produces $a_3[29]$, we expand $a_3[29]$ to a_3 $[-29,30]$ by bit carry so that the bit difference $a_3[-29]$ can produce $c_3[-8]$. Similarly, the difference $c_5[-28]$ produces $c_6[-5]$, and we expand $c_6[-5]$ to $c_6[5,-6]$ by bit carry so that $c_6[-6]$ can offset $a_6[6]$.

Finally, the message difference $\Delta m_4 = 2^{22}$ in step 18 produces $d_5[28]$ which can be cancelled out by bit difference of $c_5[-28]$. The whole route can be expressed in table 3 where the first column defines the operating step. The second is the chaining variable in each step for M. The third denotes the message word of M'. The fourth is shift rotation. The fifth is the message word difference. The sixth is the chaining variable difference. The seventh is the chaining variable for M' and the last column is the sufficient condition that guarantee the differential path to hold.

4.2 Deriving Conditions on Chaining Variable

From the differential path in Table 3, we derive a set of sufficient conditions on chaining variables from the Boolean function properties and the bit carry. For example,

The condition $b_{1,26} = c_{1,26}$ guarantees that the difference $a_2[26]$ results in no bit change in d_2.

The conditions $d_{2,26} = 0$ and $c_{2,26} = 1$ guarantee that $a_2[26]$ causes no bit change in c_2 and b_2 respectively.

The conditions $a_{3,29} = 1$ and $a_{3,30} = 0$ guarantee that the difference in a_3 has 1-bit carry.

Similarly, we can derive all the other conditions that are showed in the 8-th column of Table 3. They are also listed in Table 4.

What deserves particularly to mention is that constructing the path and deriving the conditions go on simultaneously. On one hand, we derive the sufficient conditions according to the differential path. On the other hand, we can adjust the path to avoid the contradictory conditions.

4.3 How to Verify Whether a Message Is a Weak Message

From the conditions in Table 4, we know that if M satisfies all the 62 conditions, $M' = M + \Delta M$ is the second-preimage of $h(M)$. In fact, for every $\Delta m_4 = \pm 2^i$, $0 \le i \le 31$, we can find a differential path similar to Table 3 and derive the corresponding 62 conditions which guarantee the path to hold. So any message M is a weak message with probability about $2^{-62} \times 2^5 \times 2 = 2^{-56}$. The probability can be further improved as long as we can find better differential path with less conditions.

4.4 Modifying Any Message into a Weak Message

Given message M_0, we use the **basic message modification, advanced message modification** and **bit searching** techniques to modify M_0 into a weak message M.

Basic message modification. The basic message modification technique is a kind of simple message modification used to ensure all the conditions in the first round to hold. A condition for chaining variable from compressing M_0 which isn't consistent with the condition in Table 4 is called a wrong condition. Usually correcting a wrong condition in the first round needs about single bit message modification.

For example, we can correct the condition $a_{2,26} = 1$ to $a_{2,26} = 0$ by changing the 23-rd bit of m_4, i.e,

$$m_4 \leftarrow m_4 \oplus 0x400000.$$

This can be easily seen from the following equation:

$$a_2 = (a_1 + f(b_1, c_1, d_1) + m_4) \lll 3.$$

Using a similar technique, we can correct all the wrong conditions by modifying their corresponding message words in the first 16 steps. If more than one condition need to be corrected in a step, we can correct them from the lower bit to the higher bit in order to avoid influencing the corrected conditions. For example, in step 9, we first correct the condition $b_{2,29} = c_{2,29}$, then $b_{2,30} = c_{2,30}$.

For any random message M_0, if we only fulfil the basic message modification, the modified message M is a weak message with probability 2^{-38}, and the Hamming weight between M_0 and M is about 12 because there are total 24 conditions in first 16 steps.

Advanced message modification. We can correct part of conditions in round 2 by the advanced message modification which includes various technique details.

1. The correction of $a_{5,3}$, $a_{5,8}$, $a_{5,19}$ and $a_{5,28}$ in Table 4.
   From
   $$a_5 = (a_4 + G(b_4, c_4, d_4) + m_0 + 0x5a827999) \lll 3$$
   we know that these conditions can be corrected by modifying m_0, m_1, m_2, m_3, m_4 which consists of a partial collision from 1-5 steps which ensures

that all the conditions in round 1 unchanged. But one condition correction depends on at least 5 bits of these messages that will increase about 3 Hamming weights for the difference. In order to keep the low Hamming weights, we correct them by only modifying the message words m_{14} and m_{15} mainly. For example, if $a_{5,3} \neq 0$, we correct it as follows:

$$\begin{aligned} b_4 &= b_4 \oplus 0x80000000 \\ m_{15} &= b_4 \gg 19 - b_3 - F(c_4, d_4, a_4) \\ a_5 &= (a_4 + G(b_4, c_4, d_4) + m_0 + 0x5a827999) \ll 3 \end{aligned}$$

Due to $c_{4,32} \neq d_{4,32}$, from the proposition 2, we know that the change of $b_{4,32}$ will cause the change of $a_{5,3}$. Similarly, it's easy to correct the other three conditions of a_5. The computation of the Hamming weights is included in the next part.

2. The correction of $d_{5,3}$, $d_{5,8}$, $d_{5,28}$, $c_{5,3}$, $c_{5,8}$, $b_{5,6}$ and $b_{5,8}$.
These conditions can be corrected by the similar method. We take $c_{5,3}$ for example. If $c_{5,3} \neq d_{5,3}$, we correct it by changing $c_{4,26}$ and keep the conditions in c_4, b_4, a_5, d_5 hold. The modification details are given in Table 1. The conditions $b_{4,26} = d_{4,26}$ and $a_{5,26} = b_{4,26}$ are set in advance to ensure no-change of a_5 and d_5. Indeed, there are many kinds of methods to correct a condition and this kind of advanced message modification is very flexible.

Remark 1. The condition $c_{5,28}$ can't be corrected by the method above since the condition $c_{4,19}$, $b_{4,19}$ and $a_{5,19}$ are all fixed in table 4.

Searching Conditions and Estimating Hamming Weight. There are still 27 conditions undetermined in table 4 after the correction of basic message modification and advanced message modification. We can search them exhaustively by choosing m_{14} and m_{15} randomly. For any couple of m_{14} and m_{15}, we first modify them to ensure the conditions(except $c_{5,28}$) from step 15 to step 20 hold and then check whether all the remaining 27 conditions hold. If all the conditions hold, the resulting message is a weak message. There are 15 conditions from step 15 to step 20 of Table 4 which can be corrected and they depend on about 20 bits of m_{14} and m_{15}. Therefore, there leave about a message space of 2^{40} which is large enough to search the remaining 27 conditions.

According to our analysis, we can estimate the Hamming weight for the difference of the original message M_0 and the resulting message M by counting the

Table 1. The modification for correcting $c_{5,3}$

15	m_{14}	11	$c_4' = c_4 \oplus 2^{25}, d_4, a_4, b_3$	$m_{14} \leftarrow c_4' \gg 11 - c_3 - F(d_4, a_4, b_3)$
16	m_{15}	19	b_4, c_4', d_4, a_4	$m_{15} \leftarrow b_4 \gg 19 - b_3 - F(c_4', d_4, a_4)$
17	m_0	3	a_5, b_4, c_4', d_4	$b_{4,26} = d_{4,26}$
18	m_4	5	d_5, a_5, b_4, c_4'	$a_{5,26} = b_{4,26}$
19	m_8	9	c_5', d_5, a_5, b_4	