ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.9, September 2008, pp.2442−2448
DOI: 10.3724/SP.J.1001.2008.02442
© 2008 by *Journal of Software*. All rights reserved.

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

缩减 RIPEMD-128 分析*

王高丽 [1,2+]，王美琴 [1,2]

[1](山东大学 数学与系统科学学院,山东 济南 250100)
[2](山东大学 密码技术与信息安全教育部重点实验室,山东 济南 250100)

Cryptanalysis of Reduced RIPEMD-128

WANG Gao-Li[1,2+], WANG Mei-Qin[1,2]

[1](School of Mathematics and System Sciences, Shandong University, Ji'nan 250100, China)
[2](Laboratory of Cryptographic Technology and Information Security Ministry of Education, Shandong University, Ji'nan 250100, China)
+ Corresponding author: E-mail: wanggaoli@mail.sdu.edu.cn

Wang GL, Wang MQ. Cryptanalysis of reduced RIPEMD-128. *Journal of Software*, 2008,19(9):2442−2448. http://www.jos.org.cn/1000-9825/19/2442.htm

Abstract: RIPEMD-128 is a cryptographic hash function proposed in 1996 by Hans Dobbertin, Antoon Bosselaers and Bart Preneel. It consists of two different and independent parallel parts, with which the results in each application of the compression function. This paper presents a practical attack for finding collisions for the first 32-step reduced RIPEMD-128 with complexity of 2^{28} 32-step reduced RIPEMD-128 operations. This is the first published analysis for the first 32-step reduced RIPEMD-128.

Key words: hash function; collision; RIPEMD-128; differential path; message modification

摘 要: Hans Dobbertin, Antoon Bosselaers 和 Bart Preneel 在 1996 年提出 hash 函数 RIPEMD-128,它包含两个独立并行的部分,每一部分的输出组合成 RIPEMD-128 的输出结果.给出前 32 步 RIPEMD-128 的碰撞实例,其计算复杂度是 2^{28} 次 32-步 RIPEMD-128 运算.本文是对前 32 步 RIPEMD-128 分析的第一次公开.

关键词: 杂凑函数;碰撞;RIPEMD-128;差分路经;明文修改
中图法分类号: TP309 **文献标识码**: A

1 Introduction

MD4[1] is an early-appeared hash function designed by using basic arithmetic and Boolean operations. After the publication of MD4, several hash functions have been proposed, including MD5[2], HAVAL[3], RIPEMD[4], RIPEMD-128[5], RIPEMD-160[5], SHA-0[6] and SHA-1[7], etc., most of which are based on the design principles of MD4. RIPEMD was devised in the framework of the EU project RIPE. RIPEMD-128 was proposed in 1996 by

* Supported by the National Natural Science Foundation of China under Grant No.90604036 (国家自然科学基金); the National Outstanding Young Scientist of China under Grant No.60525201 (国家杰出青年基金); the National Basic Research Program of China under Grant No.2007CB807902 (国家基础研究发展计划(973))

Received 2007-07-23; Accepted 2007-11-20

Hans Dobbertin, Antoon Bosselaers and Bart Preneel as a substitute for RIPEMD with a 128-bit result[5]. H. Dobbertin[8] gave a collision attack on MD4 which found a collision with probability 2^{-22} in 1996. Dobbertin[8] found a collision of RIPEMD reduced to two rounds with 2^{31} RIPEMD operations. Wang, *et al*.[9] found collisions on MD4 and RIPEMD with complexity less than 2^8 MD4 operations and 2^{18} RIPEMD operations respectively.

In this paper, we use the method of modular differential to analyze the hash function RIPEMD-128. This method was presented early in 1997 by Wang, and formalized in Eurocrypt'05[9,10]. The modular differential method is very efficient, by which the most prevailing hash functions such as MD4[9], MD5[10], HAVAL[11,12], SHA-0[6], SHA-1[7] etc. have been broken. Furthermore, we use the message modification proposed by X.Y. Wang to improve our collision probability. We show a cryptanalysis on reduced RIPEMD-128 which can find a collision of 32-step RIPEMD-128.

The rest of the paper is organized as follows: In Section 2, we describe the RIPEMD-128 algorithm. In Section 3, we recall some properties of the nonlinear functions in RIPEMD-128 and some notations. Section 4 presents the detailed descriptions of the attacks on reduced RIPEMD-128. Finally, we summarize the paper in Section 5.

2 Description of RIPEMD-128

The hash function RIPEMD-128 compresses any arbitrary length message into a message with a length of 128 bit. Firstly the algorithm will pad any given message into a message with a length of 512 bit multiple. We don't describe the padding process because it has little relation with our attack. For each 512-bit message block, RIPEMD-128 compresses it into a 128-bit hash value by a compression function, which has two parallel operations: Line1 and Line2. Each Line has four rounds. The nonlinear functions in each round are as follows:

$$F(X,Y,Z)=X\oplus Y\oplus Z,\qquad G(X,Y,Z)=(X\wedge Y)\vee(\neg X\wedge Z)$$
$$H(X,Y,Z)=(X\vee\neg Y)\oplus Z,\qquad I(X,Y,Z)=(X\wedge Z)\vee(Y\wedge\neg Z)$$

Here X, Y, Z are 32-bit words. The operations of four functions are all bitwise. $\neg$ represents the bitwise complement of X, $\wedge$, $\oplus$ and $\vee$ are bitwise AND, XOR and OR respectively. Each round of the compression function is composed of 16 step operations.

$$FF(a,b,c,d,x,s):a=(a+F(b,c,d)+x)<<<s,$$
$$GG(a,b,c,d,x,s):a=(a+G(b,c,d)+x+0x5a827999)<<<s$$
$$HH(a,b,c,d,x,s):a=(a+H(b,c,d)+x+0x6ed9eba1)<<<s$$
$$II(a,b,c,d,x,s):a=(a+I(b,c,d)+x+0x8f1bbcdc)<<<s,$$
$$FFF(a,b,c,d,x,s):a=(a+F(b,c,d)+x)<<<s$$
$$GGG(a,b,c,d,x,s):a=(a+G(b,c,d)+x+0x6d703ef3)<<<s$$
$$HHH(a,b,c,d,x,s):a=(a+H(b,c,d)+x+0x5c4dd124)<<<s$$
$$III(a,b,c,d,x,s):a=(a+I(b,c,d)+x+0x50a28be6)<<<s$$

The initial value of RIPEMD-128 is: (a,b,c,d) = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476).

The compression function of RIPEMD-128 consists of Line1 operation and Line2 operation.

Line1 operation process For a 512-bit block $M=(m_0,m_1,\ldots,m_{15})$, Line 1 operation process is as follows:

(1) Let (aa,bb,cc,dd) be the input of Line1 process for M. If M is the first block to be hashed, (aa,bb,cc,dd) is the initial value. Otherwise it is the output of the previous block compressing. (2) Perform the following 64 steps: For j=0,1,2,3, for i=0,1,2,3, $a=FF(a,b,c,d,w_{j,4i},s_{j,4i})$, $d=GG(a,b,c,d,w_{j,4i+1},s_{j,4i+1})$, $c=HH(a,b,c,d,w_{j,4i+2},s_{j,4i+2})$, $b=II(a,b,c,d,w_{j,4i+3},s_{j,4i+3})$. $S_{j,4i+k}$ (k=0,1,2,3) are step-dependent constants. $<<<s$ represents the circular shift s bit to the left. + denotes addition modulo 2^{32}.

Line2 operation process For a 512-bit block $M=(m_0,m_1,\ldots,m_{15})$, Line2 operation process is as follows:

(1) Let (aaa,bbb,ccc,ddd) be the input of Line2 for M. If M is the first block to be hashed, (aaa,bbb,ccc,ddd) is

the initial value. Otherwise it is the output of the previous block compressing. (2) Perform the following 64 steps: For j=0,1,2,3, for i=0,1,2,3, $a=III(a,b,c,d,w_{j,4i},s_{j,4i})$, $d=HHH(a,b,c,d,w_{j,4i+1},s_{j,4i+1})$, $c=GGG(a,b,c,d,w_{j,4i+2},s_{j,4i+2})$, $b=FFF(a,b,c,d,w_{j,4i+3},s_{j,4i+3})$. Add the output of Line1 to the output of Line2. $a=b+cc+ddd$, $b=c+dd+aaa$, $c=d+aa+bbb$, $d=a+bb+ccc$. If M is the last message block, $H(MM)=a*b*c*d$ is the hash value for the message MM. Otherwise repeat the compression process for the next 512-bit message block and (a,b,c,d) as inputs.

3 Some Basic Conclusions and Notations

In this section we will recall some useful properties of the four nonlinear functions in our attack.

Proposition 1. For the nonlinear function $F(X,Y,Z)=X\oplus Y\oplus Z$, there are the following properties:

$$F(X,Y,Z)=\neg F(\neg X,Y,Z)=\neg F(X,\neg Y,Z)=\neg F(X,Y,\neg Z)$$
$$F(X,Y,Z)=F(\neg X,\neg Y,Z)=F(X,\neg Y,\neg Z)=F(\neg X,Y,\neg Z)$$

Proposition 2. For the nonlinear function $G(X,Y,Z)=(X\wedge Y)\vee(\neg X\wedge Z)$, there are the following properties:

$$G(X,Y,Z)=G(X,\neg Y,Z)\Leftrightarrow X=0, G(X,Y,Z)=G(X,Y,\neg Z)\Leftrightarrow X=1, G(X,Y,Z)=G(\neg X,Y,Z)\Leftrightarrow Y=Z,$$
$$G(X,Y,Z)=X, G(\neg X,Y,Z)=\neg X\Leftrightarrow Y=1, Z=0, G(X,Y,Z)=Y, G(X,\neg Y,Z)=\neg Y\Leftrightarrow X=1$$
$$G(X,Y,Z)=\neg X, G(\neg X,Y,Z)=X\Leftrightarrow Y=0, Z=1, G(X,Y,Z)=Z, G(X,Y,\neg Z)=\neg Z\Leftrightarrow X=0$$

Proposition 3. For the nonlinear function $H(X,Y,Z)=(X\vee\neg Y)\oplus Z$, there are the following properties:

$$H(X,Y,Z)=H(\neg X,Y,Z)\Leftrightarrow Y=0, \qquad H(X,Y,Z)=X, H(\neg X,Y,Z)=\neg X\Leftrightarrow Y=1, Z=0$$
$$H(X,Y,Z)=\neg X, H(\neg X,Y,Z)=X\Leftrightarrow Y=0, Z=1, \qquad H(X,Y,Z)=H(X,\neg Y,Z)\Leftrightarrow X=1$$
$$H(X,Y,Z)=Y, H(X,\neg Y,Z)=\neg Y\Leftrightarrow X=0, Z=1, \qquad H(X,Y,Z)=\neg Y, H(X,\neg Y,Z)=Y\Leftrightarrow X=1, Z=0$$
$$H(X,Y,Z)=Z, H(X,Y,\neg Z)=\neg Z\Leftrightarrow X=0, Y=1, \qquad H(X,Y,Z)=\neg Z, H(X,Y,\neg Z)=Z\Leftrightarrow X=1 or Y=0$$

Proposition 4. For the nonlinear function $I(X,Y,Z)=(X\wedge Z)\vee(Y\wedge\neg Z)$, there are the following properties:

$$I(X,Y,Z)=I(\neg X,Y,Z)\Leftrightarrow Z=0, I(X,Y,Z)=I(X,\neg Y,Z)\Leftrightarrow Z=1, I(X,Y,Z)=I(X,Y,\neg Z)\Leftrightarrow X=Y$$
$$I(X,Y,Z)=X, I(\neg X,Y,Z)=\neg X\Leftrightarrow Z=1, I(X,Y,Z)=Y, I(X,\neg Y,Z)=\neg Y\Leftrightarrow Z=0$$
$$I(X,Y,Z)=Z, I(X,Y,\neg Z)=\neg Z\Leftrightarrow X=1, Y=0, I(X,Y,Z)=\neg Z, I(X,Y,\neg Z)=Z\Leftrightarrow X=0, Y=1$$

Notations. In order to describe our attack conveniently, we use the following notations. Some of them are defined in Refs.[6,7,10,11,12].

$M=(m_0,m_1,\ldots,m_{15})$ represents 512-bit messages. a_i, d_i, c_i, b_i denote the outputs of the $(4i-3)$-*th*, $(4i-2)$-*th*, $(4i-1)$-*th*, $4i$-*th* steps for compressing M, where $1\le i\le 16$. $\Delta m_i=m_i'-m_i$ denotes the modular difference of m_i and m_i'. $a_{i,j}$ represents the *j-th* bit of a_i where the least significant bit is the 1-st bit, and the most significant bit is 32-th bit. $x_i[j]$,$x_i[-j]$ 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.

4 The Practical Attack Against Reduced RIPEMD-128

The collision pair of the first 32-step reduced RIPEMD-128 consist of two 512-bit messages $M_0\|M, M_0\|M'$. We search them in the following 4 parts: (1) Denote the first 32-step reduced RIPEMD-128 by H_{32} and the output of $H_{32}(M_0)$ by $a\times b\times c\times d$. Find a message M_0 such that the outputs of $H_{32}(M_0)$ (i.e. the inputs of $H_{32}(M)$ and $H_{32}(M')$) satisfy $b_2=1$, $b_3=0$, $b_4=0$. (2) Find two near-collision differentials respectively for Line1 and Line2 operations in which M and M'produce a collision. (3) Derive two sets of sufficient conditions which ensure that the collision differentials hold. (4) Modify the message to fulfill most of the variable conditions.

Obviously the first part is easy to be accomplished. We will describe the last three parts in details.

4.1 Collision differential path for the first 32-step reduced RIPEMD-128

We use Wang's method to deduce the differential paths. After deriving the sufficient conditions for the differential paths according to the properties of the nonlinear functions, we must make sure that the sufficient conditions are not contradict each other. All the conditions in the first round and some conditions in the second round can be modified to hold by message modification technique, the other conditions in the last rounds are difficult to be modified to hold. Therefore, we will ensure the sufficient conditions in the last rounds to be as few as possible. We select $\Delta M = M' - M$ as follows: $M = (m_0, m_1, ..., m_{15}), \Delta M = (0, ..., 0, 2^{24}, 0).$ The near-collision differential paths for Line1 and Line2 are showed in Tables 1 and 2 respectively. The output differences are: $\Delta a = \Delta cc + \Delta ddd = 0,\ \Delta d = \Delta bb + \Delta ccc = 0,\ \Delta c = \Delta aa + \Delta bbb = 0,\ \Delta b = \Delta dd + \Delta aaa = 2^{31} + 2^{31} (\mathrm{mod}\, 2^{32}) = 0$.

4.2 Deriving conditions on chaining variables of Line1 and Line2

This section derives all the variable conditions that ensure the differentials in Tables 1 and 2 to hold. For example, we describe how to derive sufficient conditions that guarantee the difference in step 4 of Table 2. The input difference (ccc_1[11,12,−13],ddd_1[−2, −3,4],aaa_1,bbb_0) yields the output difference bbb_1[15,22,24,…,30,−31].

By Proposition 4, the condition $aaa_{1,i}$=1 (i=2,3) ensures that the change of $ddd_{1,i}$=1 (i=2,3) results in no change in bbb_1; $aaa_{1,4}$=0 ensures that the change of $ddd_{1,4}$ results in $\Delta bbb_1=2^{14}$. $bbb_{1,15}$=0 results in bbb_1' =bbb_1[15]. $aaa_{1,12}$=0 ensures that the change of $ccc_{1,12}$ results in no change in bbb_1. $aaa_{1,11}$=1 ensures that the change of $ccc_{1,11}$ results in $\Delta bbb_1 = 2^{21}$ and $bbb_{1,22} = 0$ results in $bbb_1' = bbb_1[22]$. $aaa_{1,13}$=1ensures that the change of $ccc_{1,13}$ results in $\Delta bbb_1 = -2^{23}$ and $bbb_{1,i} = 0 (i = 24,...,30), bbb_{1,31} = 1$ results in $bbb_1' = bbb_1[24,...,30,-31]$.

Table 1 Differential Characteristic for Line1 of 32-step reduced RIPEMD-128

Step	Chaining value	$w_{j,i}$	Shift	Δm_i	The step difference	The output for M'
15	cc_4	m_{14}	9	2^{24}	2	cc_4[2]
16	bb_4	m_{15}	8		-2^9	bb_4[−10]
17	aa_5	m_7	7		0	aa_5
18	dd_5	m_4	6		0	dd_5
19	cc_5	m_{13}	8		2^9	cc_5[10]
20	bb_5	m_1	13		0	bb_5
21	aa_6	m_{10}	11		0	aa_6
22	dd_6	m_6	9		0	dd_6
23	cc_6	m_{15}	7		2^{16}	cc_6 [17]
24	bb_6	m_3	15		0	bb_6
25	aa_7	m_{12}	7		0	aa_7
26	dd_7	m_0	12		0	dd_7
27	cc_7	m_9	15		2^{31}	cc_7 [32]
28	bb_7	m_5	9		0	bb_7
29	aa_8	m_2	11		0	aa_8
30	dd_8	m_{14}	7	2^{24}	2^{31}	dd_8 [32]
31	cc_8	m_{11}	13		0	cc_8
32	bb_8	m_8	12		0	bb_8

4.3 Message modification

We modify M so that most of the conditions of Line2 in Table 3 hold. The modified algorithm is divided into basic modification and advanced message modification techniques.

Basic Modification All the conditions in the first round (step 1−32) of Line2 can be modified to hold by the basic modification which is a simple message modification. For example, if the condition $aaa_{1,4}$=0 does not hold, we set aaa_1=aaa_1⊕0x8, then update m_5 as: m_5=(aaa_1>>>8)-aaa_0-I(bbb_0,ccc_0,ddd_0)-0x50a28be6.

Table 2 Differential characteristic for Line2 of 32-step reduced RIPEMD-128

Step	Chaining value	$w_{j,i}$	Shift	Δm_i	The step difference	The output for M'
1	aaa_1	m_5	8			aaa_1
2	ddd_1	m_{14}	9	2^{24}	2	$ddd_1[-2,-3,4]$
3	ccc_1	m_7	9		-2^{10}	$ccc_1[11,12,-13]$
4	bbb_1	m_0	11		$2^{14}+2^{21}-2^{23}$	$bbb_1[15,22,24,...,30,-31]$
5	aaa_2	m_9	13		2^8+2^{10}	$aaa_2[9,-11,-12,13]$
6	ddd_2	m_2	15		-2^9+2^{16}	$ddd_2[-10,17]$
7	ccc_2	m_{11}	15		$-2^{10}-2^{12}+2^{23}$	$ccc_2[11,-12,13,-14,24]$
8	bbb_2	m_4	5		$-2^{16}+2^{19}+2^{21}+2^{26}$	$bbb_2[-17,20,22,27]$
9	aaa_3	m_{13}	7		2^{15}	$aaa_3[16]$
10	ddd_3	m_6	7		-2^{16}	$ddd_3[17,-18]$
11	ccc_3	m_{15}	8		$2^2-2^{18}-2^{20}+2^{31}$	$ccc_3[3,-19,-21,32]$
12	ddd_3	m_8	11		$1+2^5+2^{30}$	$bbb_3[1,6,31]$
13	aaa_4	m_1	14		-2^2+2^{29}	$aaa_4[-3,30]$
14	ddd_4	m_{10}	14		$-1-2^2-2^{30}$	$ddd_4[-1,-3,-31]$
15	ccc_4	m_3	12		$-1+2^{11}-2^{30}$	$ccc_4[-1,12,-31]$
16	bbb_4	m_{12}	6		2^{11}	$bbb_4[12]$
17	aaa_5	m_6	9		2^6	$aaa_5[7]$
18	ddd_5	m_{11}	13		-2^{15}	$ddd_5[-16]$
19	ccc_5	m_3	15		$-2^{13}-2^{15}$	$ccc_5[-14,-16]$
20	bbb_5	m_7	7		$-2^{13}+2^{18}$	$bbb_5[-14,-19]$
21	aaa_6	m_0	12		2^{18}	$aaa_6[19]$
22	ddd_6	m_{13}	8		0	ddd_6
23	ccc_6	m_5	9		-2^{24}	$ccc_6[25,-26]$
24	bbb_6	m_{10}	11		-2^{24}	$bbb_6[-25]$
25	aaa_7	m_{14}	7	2^{24}	2^{25}	$aaa_7[26]$
26	ddd_7	m_{15}	7		0	ddd_7
27	ccc_7	m_8	12		0	ccc_7
28	bbb_7	m_{12}	7		2^{31}	$bbb_7[32]$
29	aaa_8	m_4	6		2^{31}	$aaa_8[32]$
30	ddd_8	m_9	15		0	ddd_8
31	ccc_8	m_1	13		0	ccc_8
32	bbb_8	m_2	11		0	bbb_8

It is easy to rectify all the conditions from step 1 to step 32 of the Line2 differential path in Table 3.

Advanced Message Modification Some more conditions in round 2 of Line2 can be rectified by the advanced message modification. If the condition on $aaa_{i,j}$ does not hold, we change the *j-th* bit of the corresponding message m to rectify it, and change some other message words to produce a partial collision in the first round of Line2. A sample for correcting $aaa_{5,7}$ is given in Table 4.

In Line2, the rectifiable conditions are as follows: $aaa_{5,i}$ (i=7,12,16), $ddd_{5,i}$ (i=7,14,16), $ccc_{5,i}$ (i=7,14,16,19), $bbb_{5,i}$ (i=14,16,19), $aaa_{6,14}$, $aaa_{6,19}$, $ddd_{6,25}$, $ddd_{6,26}$, $ccc_{6,26}$, $bbb_{6,25}$, $bbb_{6,26}$, $aaa_{7,25}$, $aaa_{7,26}$, $ccc_{7,26}$, $aaa_{8,32}$, $ddd_{8,32}$. There are 21 conditions of Line1 in Table 3. For a 512-bit messages M, after the two types of modifications, there are 6 remaining conditions of Line2 in Table 3 that need to be satisfied. Therefore $M, M'(M'=M+\Delta M)$ consist of a collision with probability 2^{-27}. It is easy to see that the complexity of finding (M,M') does not exceed 2^{28} 32-step reduced RIPEMD-128 computations. We give a 1024-bit collision $(M_0\|M, M_0\|M')$ for the first 32-step reduced RIPEMD-128 in Table 5. M_0, M, M' are all hashed by the first 32-step reduced RIPEMD-128.

Table 3 A set of sufficient conditions for collision of 32-step reduced RIPEMD-128

Step		Line 1	Line 2
			$b_{0,2}=1$, $b_{0,i}=0$ (i=3,4)
1	a_1		$aaa_{1,i}=1$ (i=2,3,11,13), $aaa_{1,i}=0$ (I=4,12)
2	d_1		$ddd_{1,i}=1$ (i=2,3,11,12,13,28,30), $ddd_{1,i}=0$ (I=4,5,12,24,…,27,29,31)
3	c_1		$ccc_{1,i}=1$ (I=13,15,22,24,25,26), $ccc_{1,i}=0$ (i=9,11,12,27,…,31)
4	b_1		$bbb_{1,i}=0$ (i=9,…,13,15,17,22,24,…,30), $bbb_{1,31}=1$, $bbb_{1,i}=ccc_{1,i}$ (I=2,3,4)
5	a_2		$aaa_{2,i}=1$ (I=10,11,12,24,28,30), $aaa_{2,i}=0$ (i=9,13,14,17)
6	d_2		$ddd_{2,i}=1$ (i=10,13,14,24), $ddd_{2,i}=0$ (i=11,12,17,20,22,27,28,30), $ddd_{2,i}=aaa_{2,i}$ (I=15,22,24,…,27,29,31)
7	c_2		$ccc_{2,i}=1$ (i=12,14,20,22,27), $ccc_{2,i}=0$ (i=11,13,16,17,24), $ccc_{2,9}=ddd_{2,9}$
8	b_2		$bbb_{2,i}=1$ (I=16,17), $bbb_{2,10}=ccc_{2,10}$, $bbb_{2,i}=0$ (i=18,20,22,27)
9	a_3		$aaa_{3,18}=1$, $aaa_{3,i}=bbb_{2,i}$ (i=11,…,14,24) $aaa_{3,i}=0$ (I=3,16,17,19,21,27,32)
10	d_3		$ddd_{3,i}=1$ (I=3,18,19,27,32), $ddd_{3,i}=aaa_{3,i}$ (i=20,22) $ddd_{3,i}=0$ (i=1,6,17,21,31)
11	c_3		$ccc_{3,i}=1$ (i=1,6,19,21,31), $ccc_{3,i}=0$ (i=3,30,32) $ccc_{3,16}=ddd_{3,16}$
12	b_3		$bbb_{3,30}=1$, $ccc_{3,i}=bbb_{3,i}$ (I=17,18), $bbb_{3,i}=0$ (i=1,3,6,19,21,31)
13	a_4		$aaa_{4,i}=1$ (i=3,19,21), $aaa_{4,i}=0$ (i=1,12,30,31) $aaa_{4,32}=bbb_{3,32}$
14	d_4	$dd_{4,2}=aa_{4,2}+1$	$ddd_{4,6}=aaa_{4,6}$, $ddd_{4,i}=1$ (I=1,3,31)
15	c_4	$cc_{4,2}=0$, $cc_{4,10}=dd_{4,10}$	$ccc_{4,i}=1$ (i=1,31), $ccc_{4,i}=0$ (I=3,12), $ccc_{4,30}=ddd_{4,30}$
16	b_4	$bb_{4,10}=1$, $bb_{4,2}=0$	$bbb_{4,i}=0$ (I=1,7,12,31)
17	a_5	$aa_{5,10}=0$, $aa_{5,2}=1$	$aaa_{5,7}=0$, $aaa_{5,12}=0$, $aaa_{5,16}=0$
18	d_5	$dd_{5,10}=1$	$ddd_{5,7}=1$, $ddd_{5,16}=1$, $ddd_{5,14}=0$
19	c_5	$cc_{5,10}=0$	$ccc_{5,i}=1$ (I=7,14,16), $ccc_{5,19}=0$
20	b_5	$bb_{5,10}=0$	$bbb_{5,16}=0$, $bbb_{5,19}=0$, $bbb_{5,14}=1$
21	a_6	$aa_{6,10}=1$	$aaa_{6,14}=0$, $aaa_{6,19}=0$
22	d_6	$aa_{6,17}=dd_{6,17}$	$ddd_{6,i}=0$ (I=19,25,26)
23	c_6	$cc_{6,17}=0$	$ccc_{6,25}=0$, $ccc_{6,26}=1$
24	b_6	$bb_{6,17}=0$	$bbb_{6,25}=1$, $bbb_{6,26}=1$
25	a_7	$aa_{7,17}=1$	$aaa_{7,25}=0$, $aaa_{7,26}=0$
26	d_7	$dd_{7,32}=aa_{7,32}$	$ddd_{7,25}=1$, $ddd_{7,26}=1$
27	c_7	$cc_{7,32}=0$	$ccc_{7,26}=0$, $ccc_{7,32}=0$
28	b_7	$bb_{7,32}=0$	$bbb_{7,32}=0$
29	a_8	$aa_{8,32}=1$	$aaa_{8,32}=0$
30	d_8	$dd_{8,32}=0$	$ddd_{8,32}=0$
31	c_8	$cc_{8,32}=0$	

Table 4 Message modification for correcting $aaa_{5,7}$

Step	m_i	Shift	Modify m_i	Chaining values after message modification
17	m_6	7	$m_6 \leftarrow (ddd_3[5]>>>7)-ddd_2-I(aaa_3,bbb_2,ccc_2)-0x50a28be6$	$ddd_3[5],aaa_3,bbb_2,ccc_2$
18	m_{15}	8	$m_{15} \leftarrow (ccc_3>>>8)-ccc_2-I(ddd_3[5],aaa_3,bbb_2)-0x50a28be6$	$ccc_3,ddd_3[5],aaa_3,bbb_2$
19	m_8	11	$m_8 \leftarrow \left(bbb_3>>>11\right)-bbb_2-I\left(ccc_3,ddd_3[5],aaa_3\right)-0x50a28be6$	$bbb_3,ccc_3,ddd_3[5],aaa_3$
20	m_1	14	$m_1 \leftarrow \left(aaa_4>>>14\right)-aaa_3-I\left(bbb_3,ccc_3,ddd_3[5]\right)-0x50a28be6$	$aaa_4,bbb_3,ccc_3,ddd_3[5]$
21	m_{10}	14	$m_{10} \leftarrow (ddd_4>>>14)-ddd_3-I(aaa_4,bbb_3,ccc_3)-0x50a28be6$	ddd_4,aaa_4,bbb_3,ccc_3

Table 5 A collision of the first 32-step reduced RIPEMD-128. *H* is the hash value without message padding

M_0	0x0587ab92,0x2cd3a579,0x7989ca1a,0x1b8148c3,0xdc532138,0xd7c68b2b,0x9569259a,0xb7015533, 0x462354d1,0x59f2c00f,0x5810a92e,0xa4abc9e9,0xb61c35be,0x5eb8bb5b,0xacf5181f,0xc7769005
M	0x848cab86,0x16327e14,0x2d7d37d2,0x74f42427,0xdc33493e,0xd3c48f2b,0x9c7d395e,0xb7fddd32, 0x029e4313,0x90eee605,0x4cb78228,0xd4abd22b,0x75a373e5,0x785710d8,0x10130778,0x67da1c0c
M_0	0x0587ab92,0x2cd3a579,0x7989ca1a,0x1b8148c3,0xdc532138,0xd7c68b2b,0x9569259a,0xb7015533, 0x462354d1,0x59f2c00f,0x5810a92e,0xa4abc9e9,0xb61c35be,0x5eb8bb5b,0xacf5181f,0xc7769005
M'	0x848cab86,0x16327e14,0x2d7d37d2,0x74f42427,0xdc33493e,0xd3c48f2b,0x9c7d395e,0xb7fddd32, 0x029e4313,0x90eee605,0x4cb78228,0xd4abd22b,0x75a373e5,0x785710d8,0x11130778,0x67da1c0c
H	0xe7fe9b03,0x59ceb5a7,0x542a0994,0xc7ca0ca9

5 Conclusions

In this paper, we find a pair of collisions on the first 32-step reduced RIPEMD-128 by using Wang's modular differential method. To break the total RIPEMD-128 (64 steps), it is necessary to look for better differential characteristics and to modify most of the sufficient conditions of the differential characteristics.

Acknowledgement We are greatly indebted to our supervisor Professor Xiaoyun Wang who gave us many elaborate instructions.

References:

[1] Rivest RL. The MD4 message digest algorithm. In: Menezes A, Vanstone SA, eds. Proc. of the Advances in Cryptology - CRYPTO'90. LNCS 537, Berlin, Heidelberg: Springer-Verlag, 1991. 303−311.

[2] Rivest RL. The MD5 message-digest algorithm. Request for comments (RFC 1321), Internet activities board, Internet privacy task force, 1992. http: // www.faqs.org/rfcs/rfcs 1321.html

[3] Zheng Y, Pieprzyk J, Seberry J. HAVAL–An one-way hashing algorithm with variable length of output. In: Jennifer S, Zheng YL, eds. Proc. of the Advances in Cryptology, Auscrypto'92. LNCS 718, Berlin, Heidelberg: Springer-Verlag, 1993. 83−104.

[4] Dobbertin H. RIPEMD with two round compress function is not collision-free. Journal of Cryptology, 1997,10(1):51−69.

[5] Dobbertin H, Bosselaers A, Preneel B. RIPEMD-160: A strengthened version of RIPEMD. In: Gollmann D, ed. Proc. of the FSE 1996. LNCS 1039, Berlin, Heidelberg: Springer-Verlag, 1996. 71−82.

[6] Wang XY, Yu HB, Lisa Y. Efficient collision search attacks on SHA-0. In: Shoup V, ed. Proc. of Crypto'05. LNCS 3621, Berlin, Heidelberg: Springer-Verlag, 2005. 1−16.

[7] Wang XY, Lisa Y, Yu HB. Finding collisions on the Full SHA-1. In: Shoup V, ed. Proc. of the Crypto'05. LNCS 3621, Berlin, Heidelberg: Springer-Verlag, 2005. 17-36.

[8] Dobbertin H. Cryptanalysis of MD4. In: Gollmann D, ed. Proc. of the FSE 1996. LNCS 1039, Berlin, Heidelberg: Springer-Verlag, 1996. 53−69.

[9] Wang XY, Lai XJ, Feng DG, Chen H, Yu XY. Cryptanalysis for hash functions MD4 and RIPEMD. In: Cramer R, ed. Proc. of the Eurocrypt2005. LNCS 3494, Berlin, Heidelberg: Springer-Verlag, 2005. 1−18.

[10] Wang XY, Yu HB. How to Break MD5 and other hash functions. In: Cramer R, ed. Proc. of the Eurocrypt2005. LNCS 3494, Berlin, Heidelberg: Springer-Verlag, 2005. 19−35.

[11] Wang XY, Feng DG, Yu XY. An attack on hash function HAVAL-128. Science in China (Series F), 2005,48(5):545−556.

[12] Yu HB, Wang XY, Yun A, Park S. Cryptanalysis of the full HAVAL with 4 and 5 passes. In: Robshaw M, ed. Proc. of the FSE 2006. LNCS 4047, Berlin, Heidelberg: Springer-Verlag, 2006. 89−110.

附中文参考文献:

[11] 王小云,冯登国,于秀源.HAVAL-128 的碰撞攻击.中国科学(E 辑),2005,35(4):405−416. http://www.ilib.cn/I-zgkx-ce.2005.04.html

WANG Gao-Li was born in 1982. She is a Ph.D. candidate at the Shandong University. Her current research areas are cryptography, etc.

WANG Mei-Qin was born in 1974. She is an assistant professor at the Shandong University. Her current research areas are computer network security, cryptography, etc.