

# Practical collision attack on 40-step RIPEMD-128

Gaoli Wang

School of Computer Science and Technology, Donghua University,  
Shanghai 201620, China  
wanggaoli@dhu.edu.cn

**Abstract.** RIPEMD-128 is an ISO/IEC standard cryptographic hash function proposed in 1996 by Dobbertin, Bosselaers and Preneel. There are two different and independent parallel lines called *line1* operation and *line2* operation, and each operation has 64 steps. The results of two line operations are combined at the end of every application of the compression function. In this paper, we present collision differential characteristics for both *line1* operation and *line2* operation by choosing a proper message difference. By using message modification technique seriously, we improve the probabilities of the differential characteristics so that we can give a collision attack on 40-step RIPEMD-128 with a complexity of  $2^{35}$  computations.

**Keywords:** Hash function, collisions, RIPEMD-128, differential characteristic, message modifications.

## 1 Introduction

The cryptographic hash function RIPEMD-128 [1] was proposed in 1996 by Hans Dobbertin, Antoon Bosselaers and Bart Preneel. It was standardized by ISO [2] and was used in HMAC in RFC [3]. The design philosophy of RIPEMD-128 adopts the experience gained by evaluating MD4 [9], MD5 [10], and RIPEMD [8] etc.. RIPEMD-128 is a double-branch hash function, where the compression function consists of two parallel operations denoted by *line1* operation and *line2* operation, respectively. The combination of  $H_{i-1}$ ,  $line1(H_{i-1}, M_{i-1})$  and  $line2(H_{i-1}, M_{i-1})$  generates the output  $H_i$ , where  $H_{i-1}$  is the standard initial value or the output of the message block  $M_{i-2}$ .

As far as we know, the published cryptanalysis of RIPEMD-128 includes collision attack [5, 12, 6], (second) preimage attack [7, 13], distinguishing attack [11], and the attack in [4]. As for the practical collision attack on step reduced RIPEMD-128, Wang et al. presented an example of collision on 32-step RIPEMD-128 in 2008 [12], Mendel et al. presented an example of collision on 38-step RIPEMD-128 in 2012 [5]. In the work [5], finding differential characteristic and performing message modification in the first round are achieved by an automatic search tool.

It is widely believed that it is difficult to construct a differential characteristic including the first round of *line1* operation because the absorption property of the boolean function  $X \oplus Y \oplus Z$  does not hold. Thus, in the collision attack on 32-step RIPEMD-128 [12], the difference of messages is chosen as  $\Delta m_{14} \neq 0, \Delta m_i = 0 (0 \leq i \leq 15, i \neq 14)$  such that the differential characteristic of *line1* operation almost keeps away from the boolean function  $X \oplus Y \oplus Z$ . Inspired by Mendel's work [5], we were motivated to find a differential characteristic of *line1* operation, which takes advantage of the property of the boolean function  $X \oplus Y \oplus Z$ .

In this paper, we use the bit tracing method to propose a collision attack on 40-step RIPEMD-128 with a complexity of  $2^{35}$ . The bit tracing method is proposed by Wang and formalized in [15, 16]. It is very powerful to break most of the dedicated hash functions such as MD4 [15, 20], RIPEMD [15], HAVAL [14, 19], MD5 [16], SHA-0 [17] and SHA-1 [18]. However, in the double-branch hash functions, two state words are updated using a single message word. Therefore, the application of bit tracing method to RIPEMD-128 is far from being trivial. In this paper, constructing differential characteristic, deducing the sufficient conditions and performing message modification are all fulfilled by hand. The previous results and our results are summarized in Table 1.

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

**Table 1.** Summary of the Attacks on RIPEMD-128

Attack	Steps	Generic	Complexity	Reference
collision	32	$2^{64}$	$2^{28}$	[12]
collision	38	$2^{64}$	$2^{14}$	[5]
collision	40	$2^{64}$	$2^{35}$	Ours
near collision	44	$2^{47.8}$	$2^{32}$	[5]
free-start collision	48	$2^{64}$	$2^{40}$	[5]
preimage	33	$2^{128}$	$2^{124.5}$	[7]
preimage	35*	$2^{128}$	$2^{121}$	[7]
preimage	36*	$2^{128}$	$2^{126.5}$	[13]
distinguishing	48	$2^{76}$	$2^{70}$	[5]
distinguishing	45	$2^{42}$	$2^{27}$	[11]
distinguishing	47	$2^{42}$	$2^{39}$	[11]
distinguishing	48	–	$2^{53}$	[11]
distinguishing	52	–	$2^{107}$	[11]
	64			[4]

\* The attack starts from an intermediate step.

## 2 Description of RIPEMD-128

The hash function RIPEMD-128 compresses any arbitrary length message into a message with length of 128 bit. Firstly the algorithm pads any given message into a message with length of 512 bit multiple. For the description of the padding method we refer to [1]. Then, for each 512-bit message block, RIPEMD-128 compresses it into a 128-bit hash value by a compression function, which is composed of two parallel operations: *line1* and *line2*. Each operation has four rounds, and each round has 16 steps. The initial value is  $(a, b, c, d) = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$ . The nonlinear functions in each round are as follows:

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

Here  $X, Y, Z$  are 32-bit words. The four boolean functions are all bitwise operations.  $\neg$  represents the bitwise complement of  $X$ .  $\wedge$ ,  $\oplus$  and  $\vee$  are bitwise AND, XOR and OR respectively. In each step of both *line1* operation and *line2* operation, one the four chaining variables  $a, b, c, d$  is updated.

$$\begin{aligned}
 \phi_0(a, b, c, d, x, s) &= (a + F(b, c, d) + x) \lll s, \\
 \phi_1(a, b, c, d, x, s) &= (a + G(b, c, d) + x + 0x5a827999) \lll s, \\
 \phi_2(a, b, c, d, x, s) &= (a + H(b, c, d) + x + 0x6ed9eba1) \lll s, \\
 \phi_3(a, b, c, d, x, s) &= (a + I(b, c, d) + x + 0x8f1bbcdc) \lll s, \\
 \psi_0(a, b, c, d, x, s) &= (a + I(b, c, d) + x + 0x50a28be6) \lll s, \\
 \psi_1(a, b, c, d, x, s) &= (a + H(b, c, d) + x + 0x5c4dd124) \lll s, \\
 \psi_2(a, b, c, d, x, s) &= (a + G(b, c, d) + x + 0x6d703ef3) \lll s, \\
 \psi_3(a, b, c, d, x, s) &= (a + F(b, c, d) + x) \lll s.
 \end{aligned}$$

$\lll s$  represents the circular shift  $s$  bit positions to the left.  $+$  denotes addition modulo  $2^{32}$ .

**line1 operation** For a 512-bit block  $M$ ,  $M = (m_0, m_1, \dots, m_{15})$ , *line1* operation is as follows:

1. Let  $(a, b, c, d) = (a_0, b_0, c_0, d_0)$  be the input of *line1* operation for  $M$ . If  $M$  is the first block to be hashed,  $(a_0, b_0, c_0, d_0)$  is the initial value. Otherwise it is the output of compressing the previous block.
2. Perform the following 64 steps (four rounds):
  - For  $j = 0, 1, 2, 3$ ,
  - For  $i = 0, 1, 2, 3$ ,
  - $a = \phi_j(a, b, c, d, m_{ord1(j,16j+4i+1)}, s1_{j,16j+4i+1})$ ,
  - $d = \phi_j(d, a, b, c, m_{ord1(j,16j+4i+2)}, s1_{j,16j+4i+2})$ ,
  - $c = \phi_j(c, d, a, b, m_{ord1(j,16j+4i+3)}, s1_{j,16j+4i+3})$ ,
  - $b = \phi_j(b, c, d, a, m_{ord1(j,16j+4i+4)}, s1_{j,16j+4i+4})$ .

**line2 operation** For a 512-bit block  $M$ ,  $M = (m_0, m_1, \dots, m_{15})$ , *line2* operation is as follows:

1. Let  $(aa, bb, cc, dd) = (a_0, b_0, c_0, d_0)$  be the input of *line2* operation for  $M$ . If  $M$  is the first block to be hashed,  $(a_0, b_0, c_0, d_0)$  is the initial value. Otherwise it is the output of compressing the previous block.
2. Perform the following 64 steps (four rounds):
  - For  $j = 0, 1, 2, 3$ ,
  - For  $i = 0, 1, 2, 3$ ,
  - $aa = \psi_j(aa, bb, cc, dd, m_{ord2(j,16j+4i+1)}, s2_{j,16j+4i+1})$ ,
  - $dd = \psi_j(dd, aa, bb, cc, m_{ord2(j,16j+4i+2)}, s2_{j,16j+4i+2})$ ,
  - $cc = \psi_j(cc, dd, aa, bb, m_{ord2(j,16j+4i+3)}, s2_{j,16j+4i+3})$ ,
  - $bb = \psi_j(bb, cc, dd, aa, m_{ord2(j,16j+4i+4)}, s2_{j,16j+4i+4})$ .

The output of compressing the block  $M$  is obtained by combining the initial value with the outputs of *line1* and *line2* operations:  $a = b_0 + cc + ddd$ ,  $b = c_0 + dd + aaa$ ,  $c = d_0 + aa + bbb$ ,  $d = a_0 + bb + ccc$ . If  $M$  is the last message block,  $a \parallel b \parallel c \parallel d$  is the hash value, where  $\parallel$  denotes the bit concatenation. Otherwise repeat the compression process for the next 512-bit message. The ordering of message words and the details of the shift positions can be seen in Table 2.

**Table 2.** Order of the Message Words and Shift Positions in RIPEMD-128

	Step $i$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<i>line1</i>	$ord1(0, i)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	$s1_{0,i}$	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
<i>line2</i>	$ord2(0, i)$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
	$s2_{0,i}$	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
	Step $i$	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
<i>line1</i>	$ord1(1, i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
	$s1_{1,i}$	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
<i>line2</i>	$ord2(1, i)$	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
	$s2_{1,i}$	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
	Step $i$	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
<i>line1</i>	$ord1(2, i)$	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
	$s1_{2,i}$	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
<i>line2</i>	$ord2(2, i)$	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
	$s2_{2,i}$	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
	Step $i$	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
<i>line1</i>	$ord1(3, i)$	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
	$s1_{3,i}$	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
<i>line2</i>	$ord2(3, i)$	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
	$s2_{3,i}$	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8

### 3 Some basic conclusions and notations

In this section we will recall some 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:

1.  $F(0, y, z) = 0$  and  $F(1, y, z) = 1 \iff y = z$ .  
 $F(0, y, z) = 1$  and  $F(1, y, z) = 0 \iff y \neq z$ .  
 $F(x, 0, z) = 0$  and  $F(x, 1, z) = 1 \iff x = z$ .  
 $F(x, 0, z) = 1$  and  $F(x, 1, z) = 0 \iff x \neq z$ .  
 $F(x, y, 0) = 0$  and  $F(x, y, 1) = 1 \iff x = y$ .  
 $F(x, y, 0) = 1$  and  $F(x, y, 1) = 0 \iff x \neq y$ .
2.  $F(x, y, z) = F(\neg x, \neg y, z) = F(x, \neg y, \neg z) = F(\neg x, y, \neg z)$ .

**Proposition 2.** For the nonlinear function  $G(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ , there are the following properties:

1.  $G(x, y, z) = G(\neg x, y, z) \iff y = z$ .  
 $G(0, y, z) = 0$  and  $G(1, y, z) = 1 \iff y = 1$  and  $z = 0$ .  
 $G(0, y, z) = 1$  and  $G(1, y, z) = 0 \iff y = 0$  and  $z = 1$ .
2.  $G(x, y, z) = G(x, \neg y, z) \iff x = 0$ .  
 $G(x, 0, z) = 0$  and  $G(x, 1, z) = 1 \iff x = 1$ .
3.  $G(x, y, z) = G(x, y, \neg z) \iff x = 1$ .  
 $G(x, y, 0) = 0$  and  $G(x, y, 1) = 1 \iff x = 0$ .

**Proposition 3.** For the nonlinear function  $H(x, y, z) = (x \vee \neg y) \oplus z$ , there are the following properties:

1.  $H(x, y, z) = H(\neg x, y, z) \iff y = 0$ .  
 $H(0, y, z) = 0$  and  $H(1, y, z) = 1 \iff y = 1$  and  $z = 0$ .  
 $H(0, y, z) = 1$  and  $H(1, y, z) = 0 \iff y = 1$  and  $z = 1$ .
2.  $H(x, y, z) = H(x, \neg y, z) \iff x = 1$ .  
 $H(x, 0, z) = 0$  and  $H(x, 1, z) = 1 \iff x = 0$  and  $z = 1$ .  
 $H(x, 0, z) = 1$  and  $H(x, 1, z) = 0 \iff x = 0$  and  $z = 0$ .
3.  $H(x, y, 0) = 0$  and  $H(x, y, 1) = 1 \iff x = 0$  and  $y = 1$ .  
 $H(x, y, 0) = 1$  and  $H(x, y, 1) = 0 \iff x = 1$  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:

1.  $I(x, y, z) = I(\neg x, y, z) \iff z = 0$ .  
 $I(0, y, z) = 0$  and  $I(1, y, z) = 1 \iff z = 1$ .
2.  $I(x, y, z) = I(x, \neg y, z) \iff z = 1$ .  
 $I(x, 0, z) = 0$  and  $I(x, 1, z) = 1 \iff z = 0$ .
3.  $I(x, y, z) = I(x, y, \neg z) \iff x = y$ .  
 $I(x, y, 0) = 0$  and  $I(x, y, 1) = 1 \iff x = 1$  and  $y = 0$ .  
 $I(x, y, 0) = 1$  and  $I(x, y, 1) = 0 \iff x = 0$  and  $y = 1$ .

**Notations** In order to describe our attack conveniently, we define some notations in the following.

1.  $M = (m_0, m_1, \dots, m_{15})$  and  $M' = (m'_0, m'_1, \dots, m'_{15})$  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  $M$  in *line1* operation, where  $1 \leq i \leq 16$ .
3.  $aa_i, dd_i, cc_i, bb_i$  respectively denote the outputs of the  $(4i - 3)$ -th,  $(4i - 2)$ -th,  $(4i - 1)$ -th and  $4i$ -th steps for compressing  $M$  in *line2* operation, where  $1 \leq i \leq 16$ .
4.  $a'_i, d'_i, c'_i, b'_i$  respectively denote the outputs of the  $(4i - 3)$ -th,  $(4i - 2)$ -th,  $(4i - 1)$ -th and  $4i$ -th steps for compressing  $M'$  in *line1* operation.
5.  $aa'_i, dd'_i, cc'_i, bb'_i$  respectively denote the outputs of the  $(4i - 3)$ -th,  $(4i - 2)$ -th,  $(4i - 1)$ -th and  $4i$ -th steps for compressing  $M'$  in *line2* operation.
6.  $\Delta m_i = m'_i - m_i$  denotes the difference of two words  $m_i$  and  $m'_i$ . It is noted that  $\Delta m_i$  is a modular difference and not a XOR difference.
7.  $x_{i,j}$  represent the  $j$ -th bit of  $x_i$ , where the least significant bit is the 1-st bit, and the most significant bit is 32-nd bit.
8.  $x_i[j], x_i[-j]$  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.
9.  $x_i[\pm j_1, \pm j_2, \dots, \pm j_l]$  is the value by change  $j_1$ -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 Collision Attack against 40-step RIPEMD-128

The collision consists of a pair of two 512-bit blocks  $(N \parallel M, N \parallel M')$ . As stated below, in order to implement the message modification, we have to add some conditions on  $b_0$ , which leads to the hash value of the first block  $N$  satisfies  $b_{0,i} = 1 (i = 1, 2, 3, 27), b_{0,i} = 0 (i = 7, \dots, 10, 13, \dots, 24)$ . We search the second block in the following three parts:

1. Choose proper differences of message words and find two concrete differential characteristics for *line1* and *line2* operations respectively in which  $M$  and  $M'$  produces a collision. The differential characteristics without round 1 must hold with high probability.
2. Derive two sets of sufficient conditions which ensure the two differential characteristics hold, respectively.
3. Modify the message to fulfill most of the chaining variable conditions.

### 4.1 Differential Characteristics for 40-step RIPEMD-128

Choosing proper differences of message words plays an important role in constructing differential characteristics which contain as many steps as possible and hold with high probabilities after message modification. Let  $M = (m_0, m_1, \dots, m_{15})$ , we select  $\Delta M = M' - M$  as follows:  $\Delta m_i = 0 (0 \leq i \leq 15, i \neq 2, 12)$ ,  $\Delta m_2 = 2^8$  and  $\Delta m_{12} = -2$ . It forms a local collision from step 25 to step 29 in *line1* operation. Although in the same round, there are the same circular shift values corresponding to the same message words between *line1* operation and *line2* operation, e.g. in step 25 (29) of *line1* operation, the shift value is 7 (11) corresponding to the message word  $m_{12}$  ( $m_2$ ), and in step 28 (32) of *line2* operation, the shift value is also 7 (11) corresponding to the message word  $m_{12}$  ( $m_2$ ), it can not form a local collision from step 28 to step 32 in *line2* operation. The reason is that the property of the boolean function  $(X \vee \neg Y) \oplus Z$  make it need at least three message words to form a local collision. Therefore, the differential characteristic of *line2* operation consists of one long local collision between step 6 to step 32. In round 3, the message differences first appear at step 41 of *line1* operation and at step 43 of *line2* operation. Thus, we can get a collision attack on 40-step RIPEMD-128 by using this message differences.

The boolean function  $X \oplus Y \oplus Z$  make it more difficult to construct a differential characteristic in *line1* operation. Hence, the differential characteristic of *line1* operation we presented in Table 8 is dense. The differential characteristic for *line2* operations is presented in Table 9, which makes the probability after round 1 hold as high as possible.

## 4.2 Deriving Conditions on Chaining Variables of *line1* and *line2* Operations

In this section, we derive two sets of sufficient conditions presented in Table 10 and Table 11, which ensure the differential characteristics in Table 8 and Table 9 hold, respectively. We describe how to derive a set of sufficient conditions that guarantee the difference in steps 3-7 of table 8 hold. Other conditions can be derived similarly.

1. In step 3, the message difference  $\Delta m_2 = 2^8$  produces  $c_1[-1, -2, 3, -24, \dots, -32]$ .
2. In step 4,  $(b_0, a_1, d_1, c_1[-1, -2, 3, -24, \dots, -32])$   
 $\implies (a_1, d_1, c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23])$ .  
 According to Proposition 1, the conditions  $d_{1,i} = a_{1,i} (i = 1, 2, 3, 31)$  ensure that the change of  $c_{1,i} (i = 1, 2, 3, 31)$  result in  $\Delta b_1 = -2^{12} - 2^{13} + 2^{14} - 2^{10}$ , meanwhile, the conditions  $d_{1,i} \neq a_{1,i} (i = 24, \dots, 30, 32)$  ensure that the change of  $c_{1,i} (i = 24, \dots, 30, 32)$  result in  $\Delta b_1 = 2^3 + \dots + 2^9 + 2^{11}$ . Combined with the conditions  $b_{1,i} = 0 (i = 4, \dots, 10, 12, 23)$  and  $b_{1,i} = 1 (i = 11, 13, \dots, 22)$ , we can get  $b'_1 = b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23]$ .
3. In step 5,  $(a_1, d_1, c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23])$   
 $\implies (d_1, c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23], a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32])$ .  
 From Proposition 1, the conditions  $b_{1,i} = d_{1,i} (i = 1, 2, 24, \dots, 27, 29, \dots, 32)$  and  $b_{1,i} \neq d_{1,i} (i = 3, 28)$  ensure that the change of  $c_1$  result in  $\Delta a_2 = 1 - 2 - 2^2 - \dots - 2^7 - 2^{28} - \dots - 2^{31}$ , meanwhile, the conditions  $c_{1,i} = d_{1,i} (i = 7, \dots, 10, 12, 17, \dots, 22)$  and  $c_{1,i} \neq d_{1,i} (i = 4, 5, 6, 11, 13, \dots, 16, 23)$  ensure that the change of  $b_1$  result in  $\Delta a_2 = -2^8 - 2^9 - 2^{10} + 2^{11} + \dots + 2^{20} - 2^{21} - \dots - 2^{27}$ . Combined with the conditions  $a_{2,i} = 0 (i = 1, 12, \dots, 21)$  and  $a_{2,i} = 1 (i = 2, \dots, 11, 22, \dots, 32)$ , we can obtain  $a'_2 = a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32]$ .
4. In step 6,  
 $(d_1, c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23], a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32])$   
 $\implies (c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23], a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32], d_2)$ .  
 From Proposition 1, it is easy to get  $a'_2 = a_2$  without no condition.
5. In step 7,  
 $(c_1[-1, -2, 3, -24, \dots, -32], b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23], a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32], d_2)$   
 $\implies (b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23], a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32], d_2, c_2)$ .  
 From Proposition 1, the conditions  $d_{2,i} = b_{1,i} (i = 1, 3)$  and  $d_{2,i} \neq b_{1,i} (i = 2, 24, \dots, 32)$  result in  $F(d'_2, a'_2, b'_1) - F(d_2, a_2, b_1) = 1 + 2 - 2^2 + 2^{23} + \dots + 2^{31}$ . Combined with  $c'_1 = c_1[-1, -2, 3, -24, \dots, -32]$  we get  $c'_2 = c_2$ .

## 4.3 Message Modification

As demonstrated in Table 10 of *line1* operation, there is no constraint on the message words  $m_i (i = 0, 9, 11, \dots, 15)$ , and there is freedom on the message words  $m_i (i = 1, 5, 7, 8, 10)$ . Thus, all the freedom of these message words can be utilized to fulfill the conditions in Table 11, which are imposed by the differential characteristic of *line2* operation.

We modify  $M$  so that all the conditions in the first round of Table 10 and most of the conditions in Table 11 hold. The outline of the modification is described as follows. Taking into consideration the fact that in Table 11 of *line2* operation, the conditions first appear in the chaining variable  $bb_1$ , and the message words  $m_5, m_{14}, m_7$  are involved in steps 1-3, we first modify  $m_i (i = 1, \dots, 7)$  such that all the conditions of  $d_1, c_1, b_1, a_2, d_2, c_2$  and  $b_2$  in Table 10 are satisfied. Then we correct the conditions of  $bb_1$  in Table 11. The message word involved in  $bb_1$  is  $m_0$ , which is also involved in the first step of *line1* operation. Therefore, if the conditions of  $bb_1$  are corrected by  $m_0$ , it will probably lead to the correction of  $d_1, c_1, b_1, a_2, d_2, c_2, b_2$  being invalid. As stated below, the condition  $bb_{1,4} = 0$  is corrected by  $m_0$ , and all the other conditions of  $bb_1$  are corrected by the change of  $dd_1$ . For example, if the condition  $bb_{1,24} = 0$  does not hold, we flip the bit  $dd_{1,13}$  by changing  $m_{14}$ . However, we need to add the condition  $b_{0,13} = 0$  such that the change of  $dd_{1,13}$  does not disturb  $cc_1$ . Meanwhile, we also need to add the condition  $aa_{1,13} = 0$  such that the change of  $dd_{1,13}$  will invert  $bb_{1,24}$ . Similarly, we need to add some conditions on the chaining variables of *line2* operation, especially on the chaining variables  $aa_1, dd_1$  and  $cc_1$ . Hence, we correct the conditions of *line2* operation from  $aa_1$ , and the process of modification is in the following. It is noted that in most cases, the conditions are corrected from low bit to high bit. Sometimes, the correction order is adjusted.

1. Modify  $m_i (i = 1, 2, 3, 4)$  such that the conditions of  $d_1, c_1, b_1$  and  $a_2$  in Table 10 hold.

2. Firstly, modify  $m_5$  such that the conditions of  $d_2$  in Table 10 hold. Secondly, if there is no overlap between the conditions on  $d_2$  in Table 10 and  $aa_1$  in Table 11, i.e. the conditions on  $aa_1$  lies in  $aa_{1,i}(i \neq 1, 2, 3, 24, \dots, 32)$ , then it is easy to correct them. For example, if the condition  $aa_{1,13} = 0$  does not hold, we flip the bit  $d_{2,13}$  by changing  $m_5$ , then  $aa_{1,13}$  is inverted, i.e.,  $aa_{1,13} = 0$  is satisfied. Thirdly, if the conditions on  $aa_1$  lies in  $aa_{1,i}(i = 1, 2, 3, 24, \dots, 32)$ , we present an example below to illustrate how to correct them. For example, if the condition  $aa_{1,1} = 0$  does not hold, we correct it by changing  $m_5$ , which will also flip the bit  $d_{2,1}$ . In order to fulfill the condition  $d_{2,1} = b_{1,1}$ ,  $b_{1,1}$  is flipped by changing  $m_3$ . Similarly,  $m_0$ ,  $m_1$  and  $m_4$  are modified in order to ensure the conditions on  $d_1$ ,  $c_1$ ,  $b_1$  and  $a_2$ , especially,  $b_{1,1} = d_{1,1}$  and  $d_{1,1} = a_{1,1}$  hold. The modification of  $m_0$ ,  $m_1$ ,  $m_3$  and  $m_4$  ensures that the differential characteristic of *line1* operation is not disturbed by the change of  $m_5$ . The detail of correcting the condition  $aa_{1,1} = 0$  is described in the following steps and illustrated in Table 3.
- Modify  $m_0$  such that  $a_{1,1}$  in Table 10 is flipped and all the other bits of  $a_1$  are unchanged. Without loss of generality, we suppose  $aa_{1,1} = 0$ , then  $a_1$  becomes  $a_1[1]$  after flipping  $a_{1,1}$ .
  - Modify  $m_1$  such that  $d_{1,1}$  in Table 10 is flipped and all the other bits of  $d_1$  are unchanged, which ensures the condition  $d_{1,1} = a_{1,1}$  in Table 10 hold.
  - The change of  $a_{1,1}$  and  $d_{1,1}$  does not disturb  $c_1$  according to Proposition 1.
  - Modify  $m_3$  such that  $b_{1,1}$  in Table 10 is flipped and all the other bits of  $b_1$  are unchanged, which ensures the condition  $b_{1,1} = d_{1,1}$  in Table 10 hold.
  - Modify  $m_4$  such that  $a_2$  in Table 10 is unchanged.
  - Modify  $m_5$  such that  $d_{2,1}$  in Table 10 is flipped and all the other bits of  $d_2$  are unchanged, which ensures the condition  $d_{2,1} = b_{1,1}$  hold. Meanwhile,  $aa_{1,1}$  is flipped by the change of  $m_5$  and the condition  $aa_{1,1} = 0$  is satisfied.

It is noted that combined with the conditions  $c_{1,1} = 1$  and  $a_{2,1} = 0$ , the flips of  $d_{1,1}$  and  $b_{1,1}$  have no impact on  $d_2$ . Hence, the modification of  $m_5$  does not need to offset the flips of  $d_{1,1}$ ,  $b_{1,1}$ , and only flips  $d_{2,1}$ . Consequently, the change of  $m_5$  is only likely to flip  $aa_{1,1}$  and  $aa_{1,i}(i = 2, \dots, 8)$  by carry. Since the conditions of  $aa_1$  are corrected from low bit to high bit, i.e., the order of modification is  $9, \dots, 32, 1, \dots, 8$ , the correction of  $aa_{1,1}$  does not disturb the conditions which have been corrected. Therefore, the condition  $aa_{1,1} = 0$  is corrected successfully with probability 1.

**Table 3.** Message Modification for Correcting  $aa_{1,1}$

	step	$m_i$	Shift	Modify $m_i$	Chaining values before modifying $m_i$	Chaining values after modifying $m_i$
<i>line1</i>	1	$m_0$	11	Modify $m_0$	$a_1$	$a_1[1]$
<i>line1</i>	2	$m_1$	14	Modify $m_1$	$d_1$	$d_1[1]$
<i>line1</i>	3	$m_2$	15		$c_1$	$c_1$
<i>line1</i>	4	$m_3$	12	Modify $m_3$	$b_1$	$b_1[1]$
<i>line1</i>	5	$m_4$	5	Modify $m_4$	$a_2$	$a_2$
<i>line1</i>	6	$m_5$	8	Modify $m_5$	$d_2$	$d_2[1]$
<i>line2</i>	1	$m_5$	8	Modify $m_5$	$aa_1$	$aa_{1,1}$ is flipped

- Modify  $m_{14}$  and  $m_6$  such that the conditions on  $dd_1$  in Table 11 and  $c_2$  in Table 10 hold, respectively.
- Firstly, modify  $m_7$  such that the conditions on  $b_2$  in Table 10 hold. Secondly, similar to the modification of  $aa_{1,i}(i \neq 1, 2, 3, 24, \dots, 32)$ , the conditions on  $cc_{1,i}(i \neq 2, \dots, 12)$  can be corrected by the change of  $m_7$ . Thirdly, the other conditions on  $cc_1$  are corrected by the change of  $dd_1$ . For example, if the condition  $cc_{1,10} = 0$  does not hold, we flip  $dd_{1,1}$  by changing  $m_{14}$ . Then  $cc_{1,10}$  is flipped if the extra condition  $b_{0,1} = 1$  is added according to Proposition 4. The detail of correcting the condition  $cc_{1,10} = 0$  is illustrated in Table 4.
- Firstly, the condition  $bb_{1,4} = 0$  is corrected by the change of  $m_0$ . If  $bb_{1,4} = 0$  does not hold, we flip  $bb_{1,4}$  by modifying  $m_0$ , which will change  $a_1$  in Table 10. On one hand, there is no constraint on  $a_1$ , so the change of  $a_1$  does not disturb the differential characteristic. On the other hand,  $d_1$ ,  $c_1$ ,  $b_1$  and  $a_2$  are unchanged by modifying

**Table 4.** Message Modification for Correcting  $cc_{1,10}$ 

step	$m_i$	Shift	Modify $m_i$	flipped bit	additional condition
2	$m_{14}$	9	Modify $m_{14}$	$dd_{1,1}$	
3	$m_7$	9		$cc_{1,10}$	$b_{0,1} = 1$

$m_1, m_2, m_3$  and  $m_4$  respectively. Therefore, the change of  $m_0$  does not disturb the differential characteristic of *line1* operation. The procedure of correcting  $bb_{1,4} = 0$  is illustrated in Table 5. Secondly, all the other conditions on  $bb_1$  are corrected by the change of  $dd_1$ . For example, if the condition  $bb_{1,24} = 0$  does not hold, we flip  $dd_{1,13}$  by changing  $m_{14}$ . Then  $cc_1$  is unchanged if the extra condition  $b_{0,13} = 0$  is added, and  $bb_{1,24}$  is flipped if the extra condition  $aa_{1,13} = 0$  is added according to Proposition 4.

**Table 5.** Message Modification for Correcting  $bb_{1,4}$ 

	step	$m_i$	Shift	Modify $m_i$	Chaining values before modifying $m_i$	Chaining values after modifying $m_i$
<i>line2</i>	4	$m_0$	11	Modify $m_0$	$bb_1$	$bb_{1,4}$ is flipped
<i>line1</i>	1	$m_0$	11	Modify $m_0$	$a_1$	$a_1$ is changed
<i>line1</i>	2	$m_1$	14	Modify $m_1$	$d_1$	$d_1$
<i>line1</i>	3	$m_2$	15	Modify $m_2$	$c_1$	$c_1$
<i>line1</i>	4	$m_3$	12	Modify $m_3$	$b_1$	$b_1$
<i>line1</i>	5	$m_4$	5	Modify $m_4$	$a_2$	$a_2$

6. Modify  $m_9$  such that the conditions on  $aa_2$  in Table 11 hold.
7. The conditions on  $dd_2$  in Table 11 are corrected through the following four approaches. All the conditions on  $dd_2$  are fulfilled after message modification except  $dd_{2,29} = 1$ . We present examples to illustrate the approaches of modification.
  - (a) The condition  $dd_{2,16} = 0$  is corrected by the change of  $m_7$ . In order not to disturb the condition  $b_{2,2} = 0$  which has been corrected, we modify  $m_7$  such that only  $b_{2,1}$  is flipped and the other bits of  $b_2$  are unchanged. The modification of  $m_7$  flips  $cc_{1,1}$  definitely, and is likely to flip  $cc_{1,i}(i = 2, \dots, 9)$  by carry. Hence, according to Proposition 4,  $bb_1$  in all probability is unchanged if the extra conditions  $aa_{1,1} = 0$  and  $aa_{1,2} = 0$  are added, and  $dd_{2,16}$  is flipped because the condition  $aa_{2,1} \neq bb_{1,1}$  is hold yet. Furthermore,  $aa_2$  is unchanged by modifying  $m_9$ . The success probability of correcting  $dd_{2,16} = 0$ , i.e., the probability that  $dd_{2,16} = 0$  is satisfied and all the other conditions which have been corrected are not disturbed, is very close to 1.
  - (b) The condition  $dd_{2,24} = 1$  is corrected by the change of  $m_{14}$ . Firstly,  $m_{14}$  is changed such that  $dd_{1,9}$  is flipped and all the other bits of  $dd_1$  are unchanged. Then, according to Proposition 4,  $cc_1$  will remain unchanged if the extra condition  $b_{0,9} = 0$  is added, and  $bb_1$  will be unchanged if the extra condition  $aa_{1,9} = 1$  is added. Furthermore,  $aa_2$  remains unchanged by modifying  $m_9$ , and  $dd_{2,24}$  is flipped by the change of  $dd_{1,9}$ .
  - (c) The condition  $dd_{2,26} = 1$  is corrected by the change of  $m_9$ . Furthermore,  $m_9$  is changed such that only  $aa_{2,11}$  is flipped and the other bits of  $aa_2$  are unchanged, which does not make the differential characteristic invalid because there is no constraint on  $aa_{2,11}$ . The change of  $aa_{2,11}$  will flip  $dd_{2,26}$  if the extra condition  $cc_{1,11} = 1$  is added.
  - (d) The condition  $dd_{2,19} = 1$  is corrected by the change of  $m_2$ . However, the change of  $m_2$  disturbs the conditions on  $c_1$ , which is compensated by modifying  $m_1$  and  $m_6$ . Firstly, we modify  $m_1$  such that  $d_{1,19}$  is flipped and all the other bits of  $d_1$  are unchanged. Then we modify  $m_2$  such that  $c_{1,19}$  is flipped and all the other bits of  $c_1$  are unchanged. According to Proposition 1, we can get  $b_1$  and  $a_2$  are unchanged, meanwhile,  $d_2$  is also unchanged because of the conditions  $c_{1,19} = d_{1,19}$ ,  $b_{1,19} = 1$  and  $a_{2,19} = 0$ . Thirdly, we modify  $m_6$  such

that  $c_2$  is unchanged. Therefore,  $b_1$ ,  $a_2$ ,  $d_2$  and  $c_2$  are unchanged, and all the conditions in Table 10 are not disturbed. Obviously, the change of  $m_2$  will flip  $dd_{2,19}$ , however, it is also likely to change  $dd_{2,2}$ . Fortunately, the conditions on  $dd_2$  are corrected from low bit to high bit and  $dd_{2,2} = 1$  is not corrected yet. So the success probability of correcting  $dd_{2,19} = 1$  is 1. The procedure of correction  $dd_{2,19}$  is illustrated in Table 6.

**Table 6.** Message Modification for Correcting  $dd_{2,19}$

	step	$m_i$	Shift	Modify $m_i$	Chaining values before modifying $m_i$	Chaining values after modifying $m_i$	Conditions
line1	2	$m_1$	14	Modify $m_1$	$d_1$	$d_1[19]$	
line1	3	$m_2$	15	Modify $m_2$	$c_1$	$c_1[19]$	$c_{1,19} = d_{1,19}$
line1	4	$m_3$	12		$b_1$	$b_1$	$b_{1,19} = 1$
line1	5	$m_4$	5		$a_2$	$a_2$	$a_{2,19} = 0$
line1	6	$m_5$	8		$d_2$	$d_2$	
line1	7	$m_6$	7	Modify $m_6$	$c_2$	$c_2$	$c_{2,19} = d_{2,19}$
line2	6	$m_2$	15	Modify $m_2$	$dd_2$	$dd_{2,19}$ is flipped	$dd_{2,19} = 1$

8. Modify  $m_{11}$  to correct the conditions of  $cc_2$  in Table 11.
9. Similar to the procedure of modification above, the conditions of  $bb_{2,i}$  ( $i \neq 1, 4, 8, 16, 23, 24, 25, 26, 29, 31, 32$ ) in Table 11 are corrected by changing  $cc_2$  or  $aa_2$ , corresponding to changing  $m_{11}$  or  $m_9$ , respectively.
10. Modify  $m_{13}$  to correct the conditions of  $aa_3$ .
11. Similar to the procedure of modification above, the conditions of  $dd_{3,i}$  ( $i \neq 2, 5, 7, 23, 25, 26, 30, 31, 32$ ) in Table 11 are corrected by changing  $aa_3$ , corresponding to changing  $m_{13}$ .
12. Modify  $m_{15}$  to correct the conditions of  $cc_3$ .
13. Firstly, modify  $m_8$  such that the conditions on  $a_3$  in Table 10 and  $bb_{3,i}$  ( $i = 23, \dots, 32$ ) in Table 11 hold. Secondly, the condition  $bb_{3,12} = 1$  in Table 11 is corrected by flipping  $cc_{3,1}$  combined with the condition  $aa_{3,1} = 1$  according to Proposition 4. Thirdly, if the condition  $bb_{3,2} = 0$  does not hold, we flip  $cc_{3,22}$ , then  $bb_{3,1}$  is flipped if the extra condition  $aa_{3,22} = 1$  (which is satisfied in step 10) is added according to Proposition 4. Meanwhile, if  $bb_{3,1} \neq cc_{3,22}$ , then the change of  $bb_{3,1}$  will result in the change of  $bb_{3,2}$  by bit carry. Furthermore, the condition  $bb_{3,1} \neq cc_{3,22}$  can be corrected by modifying  $m_8$ .
14. Firstly, the condition on  $aa_{4,5}$  can be corrected by the change of  $cc_{3,23}$  and  $bb_{3,23}$ . Similarly, the condition on  $aa_{4,9}$  can be corrected by the change of  $cc_{3,27}$  and  $bb_{3,27}$ . Secondly, the condition  $aa_{4,25} = 1$  in Table 11 is corrected by flipping  $cc_{3,11}$ . Then  $bb_3$  is unchanged if the extra condition  $aa_{3,11} = 0$  is added, and  $aa_{4,25}$  is changed if the extra condition  $dd_{3,11} = 0$  is added according to Proposition 4. The condition  $aa_{3,11} = dd_{3,11}$  is already corrected in step 11, thus, the extra conditions  $aa_{3,11} = 0$  and  $dd_{3,11} = 0$  hold with a probability of  $2^{-1}$ . Therefore, the success probability of correcting the condition on  $aa_{4,25}$  is about  $2^{-1} + 2^{-1} \times 2^{-1} = 3/4$ . Thirdly, if the condition  $aa_{4,7} = 0$  does not hold, we flip  $cc_{3,24}$ , then  $bb_3$  is unchanged if the extra condition  $aa_{3,24} = 0$  is added, and  $aa_{4,6}$  is changed if the extra condition  $dd_{3,24} = 0$  is added according to Proposition 4. Furthermore, if  $aa_{4,6} \neq cc_{3,24}$ , then the change of  $aa_{4,6}$  will lead to the change of  $aa_{4,7}$  by bit carry. The condition  $aa_{3,24} = dd_{3,24}$  is already corrected in step 11, thus, the extra conditions  $aa_{3,24} = 0$  and  $dd_{3,24} = 0$  hold with a probability of  $2^{-1}$ . Meanwhile, the condition  $aa_{4,6} \neq cc_{3,24}$  holds with a probability of  $2^{-1}$ . Therefore, the success probability of correcting the condition on  $aa_{4,7}$  is about  $2^{-1} + 2^{-1} \times 2^{-1} \times 2^{-1} = 5/8$ .
15. The condition  $dd_{4,9} = 1$  is corrected by flipping  $cc_{3,13}$ . Then  $bb_3$  is unchanged if the extra condition  $aa_{3,13} = 0$  is added, and  $aa_{4,27}$  is flipped if the extra condition  $dd_{3,13} = 0$  is added. The change of  $aa_{4,27}$  will result in the change of  $dd_{4,9}$  if the extra condition  $cc_{3,27} = 1$  is added. The condition  $cc_{3,27} = 1$  has been corrected in step 12. The condition  $dd_{3,13} = aa_{3,13}$  has been corrected in step 11, thus, the extra conditions  $aa_{3,13} = 0$  and  $dd_{3,13} = 0$  hold with a probability of  $2^{-1}$ . Therefore, the success probability of correcting the condition on  $dd_{4,9}$  is about  $2^{-1} + 2^{-1} \times 2^{-1} = 3/4$ .
16. The conditions on  $cc_{4,i}$  ( $i = 7, 9, 12$ ) are corrected by the change of  $dd_{4,i}$  ( $i = 27, 29, 32$ ) respectively with probability 1. The condition  $cc_{4,5} = 1$  is corrected by flipping  $dd_{4,24}$  if the extra condition  $cc_{4,4} \neq dd_{4,24}$  is added, which

holds with a probability of  $2^{-1}$ . Therefore, the success probability of correcting the condition on  $cc_{4,5}$  is about  $2^{-1} + 2^{-1} \times 2^{-1} = 3/4$ .

It is noted that the conditions which are corrected in the first 12 steps hold with a probability of about  $2^{-3}$  by experiment after message modification. Meanwhile, after message modification, in the first round of *line2* operation in Table 11, there are 29 conditions which are not corrected, 3 conditions which hold with a probability of  $3/4$  respectively, and 1 condition which holds with a probability of  $5/8$ . Therefore, all the conditions in steps 2-11 of Table 10 and in steps 4-15 of Table 11 hold with a probability of about  $2^{-35}$ .

#### 4.4 Collision Search Algorithm

From the above technique details, we present an overview of the collision search algorithm to get the second block  $M = m_0 \parallel m_1 \parallel \dots \parallel m_{15}$ .

1. Randomly choose  $m_i$  ( $0 \leq i \leq 15, i \neq 12$ ), and modify them by the above modification techniques such that all the conditions in steps 2-11 of Table 10 are satisfied and all the conditions in steps 4-15 of Table 11 hold with a probability of  $2^{-35}$ .
2. If all the conditions in steps 4-15 of Table 11 are satisfied, then goto Step 3. Otherwise, go to Step 1.
3. Randomly choose  $m_{12}$  and compute the hash values of  $M$  and  $M'$  under 40-step RIPEMD-128. If the two hash values are equal, then output  $M$  and  $M'$ . Otherwise, goto Step 1.

There are 4 conditions in steps 24-27 of Table 10 and 17 conditions in steps 16-29 of Table 11. It is easy to make the 21 conditions hold by exhaustively search  $m_{12}$  when the other conditions of Table 10 and Table 11 hold by our experiment. Therefore, the time complexity of the collision attack is about  $2^{35} + 2^{21}$  40-step RIPEMD-128 computations. We give an example in Table 7.

**Table 7.** Collision for 40-step of RIPEMD-128.

$N$	664504b6	d6e949ba	2176407d	85426fc1	5ec28995	c3d318b	787db431	ae2c13fb	cee9d90	c5078e4b	84bae5bc	99f3f4ae	d7403dc6	917fa14c	85155db5	fd9311e6
$M$	a7e4a89f	6278156c	2a535118	90eba965	670841b2	ea6f8dcb	800766d9	d0bfa5c6	ffe74d8e	6df2c5f7	a3ffdbfd	53e156d4	54f75d	f0d3a13f	7eef12b9	ef317f76
$M'$	a7e4a89f	6278156c	2a535218	90eba965	670841b2	ea6f8dcb	800766d9	d0bfa5c6	ffe74d8e	6df2c5f7	a3ffdbfd	53e156d4	54f75b	f0d3a13f	7eef12b9	ef317f76
$H$	a76df6ab	43ae1a6e	171d9fda	da03925e												

## 5 Conclusions

In this paper, we present a practical collision attack for 40-step RIPEMD-128 with a complexity of about  $2^{35}$  computations. Firstly, we find two differential characteristics for *line1* operation and *line2* operation respectively. Then, by correcting most of the sufficient conditions that ensure the collision characteristics hold, we can improve the probabilities of the characteristics. Furthermore, our attack can be extended to near-collision attack on more steps than 40.

## References

1. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE. LNCS 1039, pp. 71-82. Springer (1996)

2. International Organization for Standardization: ISO/IEC 10118-3:2004, Information technology-Security techniques-Hash-functions-Part 3: Dedicated hash functions (2004)
3. Kap, J.: Test Cases for HMAC-RIPEDM160 and HMAC-RIPEDM128. Internet Engineering Task Force (IETF), RFC 2286 (1998), <http://www.ietf.org/rfc/rfc2286.txt>
4. Landelle, F., Peyrin, T.: Cryptanalysis of Full RIPEMD-128. Accepted by EUROCRYPT'13.
5. Mendel, F., Nad, T., Schlaffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE. LNCS 7549, pp. 226-243. Springer (2012)
6. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. LNCS 4176, pp. 101-116. Springer (2006)
7. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt. LNCS 6584, pp. 169-186. Springer (2011)
8. RIPE, Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040), LNCS 1007, 1995.
9. Rivest, R.: The MD4 message digest algorithm, Advanced in Cryptology. In: Menezes, A., Vanstone, S. (eds.) CRYPTO. LNCS 537, pp. 303-312. Springer (1990)
10. Rivest, R.: The MD5 message-digest algorithm, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.
11. Sasaki, Y., Wang, L.: Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions. In: Bao, F., Samarati, P., Zhou, J. (eds.): ACNS. LNCS 7341, pp. 275-292. Springer (2012)
12. Wang G., Wang M.: Cryptanalysis of Reduced RIPEMD-128. Ruanjianxuebao/Journal of Software in Chinese, 2008,19(9): 2442-2448.
13. Wang, L., Sasaki, Y., Komatsubara, W., Ohta, K., Sakiyama, K.: (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In: Kiayias, A. (ed.) CT-RSA. LNCS 6558, pp. 197-212. Springer (2011)
14. Wang, X., Feng, D., Yu, X.: An attack on HAVAL function HAVAL-128. Science in China Ser. F Information Sciences 2005 Vol.48, No.5, 1-12.
15. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis for Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT. LNCS 3494, pp. 1-18. Springer (2005)
16. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. LNCS 3494, pp. 19-35. Springer (2005)
17. Wang, X., Yu, B., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO. LNCS 3621, pp. 1-16. Springer (2005)
18. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. LNCS 3621, pp. 17-36. Springer (2005)
19. Yu, H., Wang, X., Yun, A., Park, S.: Cryptanalysis of the full HAVAL with 4 and 5 passes. In: Robshaw, M. (ed.) FSE. LNCS 4047, pp. 89-110. Springer (2006)
20. Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-Preimage Attack on MD4. In: Desmedt, Y.G. et al. (eds.) CANS. LNCS 3810, pp. 1-12. Springer (2005)

**Table 8.** Differential Characteristic for *line1* Operation

Step	Message $M$	$Shift$	$\Delta m_i$	The output for $M'$
1	$m_0$	11		$a_1$
2	$m_1$	14		$d_1$
3	$m_2$	15	$2^8$	$c_1[-1, -2, 3, -24, \dots, -32]$
4	$m_3$	12		$b_1[4, \dots, 10, -11, 12, -13, \dots, -22, 23]$
5	$m_4$	5		$a_2[1, -2, \dots, -11, 12, \dots, 21, -22, \dots, -32]$
6	$m_5$	8		$d_2$
7	$m_6$	7		$c_2$
8	$m_7$	9		$b_2[2, \dots, 10, -11, -12]$
9	$m_8$	11		$a_3[-2, \dots, -11, 12]$
10	$m_9$	13		$d_3$
11	$m_{10}$	14		$c_3$
12	$m_{11}$	15		$b_3$
13	$m_{12}$	6	-2	$a_4$
...	...	...	...	...
25	$m_{12}$	7	-2	$a_7[-9]$
26	$m_0$	12		$d_7$
27	$m_9$	15		$c_7$
28	$m_5$	9		$b_7$
29	$m_2$	11	$2^8$	$a_8$
...	...	...	...	...
40	$m_1$	15		$b_{10}$

**Table 9.** Differential Characteristic for *line2* Operation

Step	Message $M$	Shift	$\Delta m_i$	The output for $M'$
1	$m_5$	8		$aa_1$
2	$m_{14}$	9		$dd_1$
3	$m_7$	9		$cc_1$
4	$m_0$	11		$bb_1$
5	$m_9$	13		$aa_2$
6	$m_2$	15	$2^8$	$dd_2[-1, -2, -3, 4, -24, \dots, -32]$
7	$m_{11}$	15		$cc_2[17, 18 - 19]$
8	$m_4$	5		$bb_2[8, \dots, 15, -16, -24]$
9	$m_{13}$	7		$aa_3[-31]$
10	$m_6$	7		$dd_3[8, -23, 26, \dots, 31, -32]$
11	$m_{15}$	8		$cc_3[7, 8, -25]$
12	$m_8$	11		$bb_3[2, 5]$
13	$m_1$	14		$aa_4[7, -9, -12]$
14	$m_{10}$	14		$dd_4[-5, 7, -9]$
15	$m_3$	12		$cc_4[-5]$
16	$m_{12}$	6	-2	$bb_4$
17	$m_6$	9		$aa_5[-21]$
18	$m_{11}$	13		$dd_5[-20, -21]$
19	$m_3$	15		$cc_5[-20]$
20	$m_7$	7		$bb_5$
21	$m_0$	12		$aa_6$
22	$m_{13}$	8		$dd_6[-29]$
23	$m_5$	9		$cc_6[-29]$
24	$m_{10}$	11		$bb_6$
25	$m_{14}$	7		$aa_7$
26	$m_{15}$	7		$dd_7$
27	$m_8$	12		$cc_7[-9]$
28	$m_{12}$	7	-2	$bb_7[-9]$
29	$m_4$	6		$aa_8$
30	$m_9$	15		$dd_8$
31	$m_1$	13		$cc_8$
32	$m_2$	11	$2^8$	$bb_8$
...	...	...	...	...
40	$m_9$	14		$bb_{10}$

**Table 10.** A Set of Sufficient Conditions for the Differential Characteristic in Table 8

Step	Chaining Variable	Conditions on the Chaining Variable
2	$d_1$	$d_{1,i} = a_{1,i}(i = 1, 2, 3, 31), d_{1,i} \neq a_{1,i}(i = 24, \dots, 30, 32)$
3	$c_1$	$c_{1,3} = 0, c_{1,i} = 1(i = 1, 2, 24, \dots, 32), c_{1,i} = d_{1,i}(i = 7, \dots, 10, 12, 17, \dots, 22), c_{1,i} \neq d_{1,i}(i = 4, 5, 6, 11, 13, \dots, 16, 23)$
4	$b_1$	$b_{1,i} = 0(i = 4, \dots, 10, 12, 23), b_{1,i} = 1(i = 11, 13, \dots, 22), b_{1,i} = d_{1,i}(i = 1, 2, 24, \dots, 27, 29, \dots, 32), b_{1,i} \neq d_{1,i}(i = 3, 28)$
5	$a_2$	$a_{2,i} = 0(i = 1, 12, \dots, 21), a_{2,i} = 1(i = 2, \dots, 11, 22, \dots, 32)$
6	$d_2$	$d_{2,i} = b_{1,i}(i = 1, 3), d_{2,i} \neq b_{1,i}(i = 2, 24, \dots, 32)$
7	$c_2$	$c_{2,i} = d_{2,i}(i = 1, \dots, 10, 13, \dots, 21, 24), c_{2,i} \neq d_{2,i}(i = 11, 12, 22, 23, 25, \dots, 32)$
8	$b_2$	$b_{2,i} = 0(i = 2, \dots, 10), b_{2,i} = 1(i = 11, 12)$
9	$a_3$	$a_{3,12} = 0, a_{3,i} = 1(i = 2, \dots, 11)$
11	$c_3$	$c_{3,i} = d_{3,i}(i = 2, \dots, 10, 12), c_{3,11} \neq d_{3,11}$
24	$b_6$	$b_{6,9} = c_{6,9}$
25	$a_7$	$a_{7,9} = 1$
26	$d_7$	$d_{7,9} = 0$
27	$c_7$	$c_{7,9} = 1$

**Table 11.** A Set of Sufficient Conditions for the Differential Characteristic in Table 9

Step	Chaining Variable	Conditions on the Chaining Variable
4	$bb_1$	$bb_{1,i} = 0(i = 1, 3, 4, 24, \dots, 32), bb_{1,2} = 1$
5	$aa_2$	$aa_{2,i} = 0(i = 3, 17, 18), aa_{2,i} = 1(i = 1, 2, 4, 19, 24, \dots, 32)$
6	$dd_2$	$dd_{2,i} = 0(i = 4, 8, \dots, 16), dd_{2,i} = 1(i = 1, 2, 3, 17, 18, 19, 24, \dots, 32)$
7	$cc_2$	$cc_{2,i} = 0(i = 16, 17, 18, 24, 26, \dots, 32), cc_{2,i} = 1(i = 8, \dots, 15, 19)$
8	$bb_2$	$bb_{2,i} = 0(i = 8, \dots, 15, 19, 23, 26, \dots, 32), bb_{2,i} = 1(i = 16, 24), bb_{2,i} = cc_{2,i}(i = 1, 2, 3, 4, 25)$
9	$aa_3$	$aa_{3,i} = 0(i = 7, 23, 27), aa_{3,i} = 1(i = 8, 19, 25, 26, 28, \dots, 32), aa_{3,i} = bb_{2,i}(i = 17, 18)$
10	$dd_3$	$dd_{3,i} = 0(i = 2, 5, 8, 25, \dots, 31), dd_{3,i} = 1(i = 7, 23, 32), dd_{3,i} = aa_{3,i}(i = 9, \dots, 16, 24)$
11	$cc_3$	$cc_{3,i} = 0(i = 7, 8, 12), cc_{3,i} = 1(i = 2, 5, 9, 25, 26, 30, 31)$
12	$bb_3$	$bb_{3,i} = 0(i = 2, 5, 8, 25, 26, 30, 31), bb_{3,i} = 1(i = 7, 12), bb_{3,i} = cc_{3,i}(i = 23, 27, 28, 29), bb_{3,32} \neq cc_{3,32}$
13	$aa_4$	$aa_{4,i} = 0(i = 5, 7), aa_{4,i} = 1(i = 8, 9, 12, 25)$
14	$dd_4$	$dd_{4,7} = 0, dd_{4,i} = 1(i = 5, 9), dd_{4,2} = aa_{4,2}$
15	$cc_4$	$cc_{4,i} = 0(i = 7, 9), cc_{4,5} = 1, cc_{4,12} = dd_{4,12}$
16	$bb_4$	$bb_{4,i} = 0(i = 5, 21)$
17	$aa_5$	$aa_{5,20} = 0, aa_{5,21} = 1$
18	$dd_5$	$dd_{5,i} = 1(i = 20, 21)$
19	$cc_5$	$cc_{5,21} = 0, cc_{5,20} = 1$
20	$bb_5$	$bb_{5,20} = 0$
21	$aa_6$	$aa_{6,29} = 0$
22	$dd_6$	$dd_{6,29} = 1$
23	$cc_6$	$cc_{6,29} = 1$
24	$bb_6$	$bb_{6,29} = 0$
26	$dd_7$	$dd_{7,9} = 0$
27	$cc_7$	$cc_{7,9} = 1$
28	$bb_7$	$bb_{7,9} = 1$
29	$aa_8$	$aa_{8,9} = 0$