Improved Preimage Attacks on RIPEMD-160 and HAS-160

Yanzhao Shen[1, 2] and Gaoli Wang[3]
[1] School of Mathematics, Shandong University, Jinan 250100, Shandong, China
[e-mail: yanzhaoshen@mail.sdu.edu.cn]
[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, Shandong, China
[3] Shanghai Key Laboratory of Trustworthy Computing, School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China
[e-mail: glwang@sei.ecnu.edu.cn]
*Corresponding author: Gaoli Wang

Received August 30, 2017; accepted September 30, 2017; published February 28, 2018

Abstract

The hash function RIPEMD-160 is a worldwide ISO/IEC standard and the hash function HAS-160 is the Korean hash standard and is widely used in Korea. On the basis of differential meet-in-the-middle attack and biclique technique, a preimage attack on 34-step RIPEMD-160 with message padding and a pseudo-preimage attack on 71-step HAS-160 without message padding are proposed. The former is the first preimage attack from the first step, the latter increases the best pseudo-preimage attack from the first step by 5 steps. Furthermore, we locate the linear spaces in another message words and exchange the bicliques construction process and the mask vector search process. A preimage attack on 35-step RIPEMD-160 and a preimage attack on 71-step HAS-160 are presented. Both of the attacks are from the intermediate step and satisfy the message padding. They improve the best preimage attacks from the intermediate step on step-reduced RIPEMD-160 and HAS-160 by 4 and 3 steps respectively. As far as we know, they are the best preimage and pseudo-preimage attacks on step-reduced RIPEMD-160 and HAS-160 respectively in terms of number of steps.

Keywords: Cryptography, Preimage attack, RIPEMD-160, HAS-160, Differential meet-in-the-middle, Hash function

This work was supported by National Basic Research Program of China (973 Program) 2013CB834205, National Natural Science Foundation of China (Nos. 61373142, 61572125, 61632012) and Shanghai High-Tech Field Project of 16511101400.
This work is the extension of “(Pseudo-) Preimage Attacks on Step-Reduced HAS-160 and RIPEMD-160” which appeared in the 17th International Conference on Information Security (ISC 2014).

http://doi.org/10.3837/tiis.2018.02.011 **ISSN : 1976-7277**

1. Introduction

Hash Functions play an important role in modern cryptography. They should satisfy three classical security properties: collision resistance, preimage resistance and second preimage resistance. With the breakthroughs in the collision attacks on MD5 [1], SHA-0 [2] and SHA-1 [3], research on preimage attack is strongly motivated (see [4-11] for example). For an *n*-bit hash function, a generic algorithm requires 2^n computations for finding a preimage. The meet-in-the-middle attack [4, 12] is the basic tool used in preimage attacks which is improved by splice-and-cut technique [4], biclique technique [6], initial structure technique [10] and so on. Knellwolf and Khovratovich [7] introduced the differential meet-in-the-middle attack which is suitable for the hash functions with linear message expansion and weak diffusion properties. Recently, Espitau et al. [13] generalized it to the higher-order differential meet-in-the-middle attack.

RIPEMD-160 [14] was designed by Dobbertin et al. and standardized in ISO/ IEC 10118-3 [15]. It is one of the unbroken hash functions which implemented in numerous security protocols. The first collision attack on RIPEMD-160 was proposed by Mendel et al. [16]. Mendel et al. [17] improved the collision attack using the method of the attacks on MD5 and SHA-1 [1, 3]. They obtained a 48-step semi-free-start near-collision attack and a 36-step semi-free-start collision attack on RIPEMD-160. Later Mendel et al. [18] improved the 36-step semi-free-start collision attack to 42-step. Furthermore, they, using a carefully designed non-linear path search tool, obtained a 36-step semi-free-start collision attack from the first step. As for the preimage attack, Ohtahara et al. [8] presented the first preimage attack on step-reduced RIPEMD-160. They gave a 30-step second-preimage attack and a 31-step preimage attack on RIPEMD-160 using the local-collision and initial structure techniques. Moreover, Sasaki and Wang [19] proposed the distinguishing attacks on RIPEMD-160. They gave a practical 2-dimension sum distinguisher on 42-step RIPEMD-160 and a theoretical one on 51-step RIPEMD-160.

HAS-160 [20] is the Korean hash standard and is widely used in Korea. Its structure is similar to SHA-0 and SHA-1 expect the message expansion. Yun et al. [21] presented the first cryptanalysis on HAS-160. They proposed a collision attack on 45-step HAS-160 by applying techniques introduced by Wang et al. [22]. Later Cho et al. [23] extended the previous work to 53 steps by selecting message differences judiciously and constructing a more complicated first round differential path. Mendel and Rijmen [24] improved the attack on 53-step HAS-160 to a practical one by using a slightly different message modification technique and extended the attack to 59 steps. The best collision attack on HAS-160 is a practical 65-step semi-free-start collision attack which proposed by Mendel et al. [25]. Sasaki and Aoki [9] proposed the first preimage attack on step-reduced HAS-160. They presented a 48-step preimage attack using splice-and-cut and partial-matching techniques and a 52-step preimage attack using local-collision technique. Hong et al. [5] improved the Sasaki and Aoki's work [9] to 68 steps. Moreover, Kircanski et al. [26] proposed a boomerang attack on the full version of HAS-160.

This paper improves the preimage and pseudo-preimage attacks on step-reduced RIPEMD-160 and HAS-160 using differential meet-in-the-middle attack and biclique technique. A preimage attack on 34-step RIPEMD-160 (with message padding) with complexity $2^{158.91}$ and a pseudo-preimage attack on 71-step HAS-160 (without message padding) with complexity $2^{158.13}$ are presented. Both of the attacks are from the first step.

Furthermore, preimage attacks from the intermediate step and satisfying the message padding are also proposed. A preimage attack on 35-step RIPEMD-160 holds with a complexity of $2^{159.38}$ and a preimage attack on 71-step HAS-160 holds with a complexity of $2^{158.97}$. In the attack on RIPEMD-160, we locate the linear spaces in one message word to decrease the error probability. Owing to the linear spaces located in the same message word and the bicliques located in the Left branch and the Right branch, the bicliques are more sophisticated. So we exchange the bicliques construction process and the mask vector search process to make the mask vector match the constant bits of initial values. For this reason, we improve the preimage attack on RIPEMD-160 from 34-step to 35-step. Because the pseudo-preimage attack on 71-step HAS-160 uses the message word m_{15}, it does not satisfy the message padding. However, the preimage attack on 71-step HAS-160 easily satisfies the message padding by fixing the linear spaces in message words m_0, m_{10}, m_6 and m_{12}. The dimension of the linear spaces increases from 3 to 5, which benefits from the linear spaces located in these words. In this case, the complexity of pseudo-preimage attack is improved and a preimage attack on 71-step HAS-160 is obtained. As far as we know, they are the best preimage and pseudo-preimage attacks on step-reduced RIPEMD-160 and HAS-160 in term of number of steps. The previous work and our results about the (pseudo-)preimage attacks are summarized in **Table 1**.

Table 1. Summary of the (pseudo-)preimage attacks on RIPEMD-160 and HAS-160

Hash Function	Steps	Complexity		Source
		Pseudo-preimage	Preimage	
RIPEMD-160	31(50-79)	2^{148}	2^{155}	[8]
RIPEMD-160	34(0-33)	$2^{155.81}$	$2^{158.91}$	Section 4.1
RIPEMD-160	35(1-35)	$2^{156.75}$	$2^{159.38}$	Section 4.2
HAS-160	48(1-48, 21-68)	2^{128}	2^{145}	[9]
HAS-160	52(12-63)	2^{144}	2^{153}	[9]
HAS-160	65(0-64)	$2^{143.4}$	$2^{152.7}$	[5]
HAS-160	67(0-66)	2^{154}	2^{158}	[5]
HAS-160	68(12-79)	$2^{150.7}$	$2^{156.3}$	[5]
HAS-160	71(0-70)	$2^{158.13}$	None	Section 4.3
HAS-160	71(6-76)	$2^{155.93}$	$2^{158.97}$	Section 4.4

Section 2 summarizes the related techniques and differential meet-in-the-middle attack with bicliques. Section 3 gives a brief description of RIPEMD-160 and HAS-160. Section 4 presents preimage and pseudo-preimage attacks on step-reduced RIPEMD-160 and HAS-160. Finally, we conclude the paper in Section 5.

2. Preliminary

In this part, we briefly summarize the differential meet-in-the-middle attack and the biclique search algorithm. Here, some notations are introduced in advance(see **Table 2**).

Table 2. The major symbols used in this paper

Symbols	Definitions
LD_1	the linear space used in CF_1
LD_2	the linear space used in CF_2
d	the dimension of LD_1 and LD_2
n	the length of the chaining value and hash value

P	the output list of CF_3
Q	the input list of CF_3
$P[\delta_2]$	the element of P corresponding to δ_2
$Q[\delta_1]$	the element of Q corresponding to δ_1
T	the mask vector, where $T \in \{0,1\}^n$ is a d dimension vector
$\Delta[i]$	the i-th bit of Δ, where $0 \le i \le n-1$

2.1 Converting pseudo-preimages to a preimage

The classical algorithm that converts pseudo-preimages to a preimage [27] is described. Assume there is an algorithm that finds a pseudo-preimage in the complexity of 2^k. Prepare a table that includes $2^{n-k/2}$ entries of pseudo-preimages. The complexity of this step is $2^{n+k/2} = 2^{n-k/2} \times 2^k$. Randomly chosen $2^{n+k/2}$ 1-block message, calculate the output of the message, then one of them agrees with the entries in the table with high probability. The complexity of the matching step is $2^{n+k/2}$. The corresponding message is a preimage of the hash function. Therefore, the total complexity for finding a preimage is $2^{n+k/2+1} = 2^{n+k/2} + 2^{n+k/2}$.

2.2 The differential meet-in-the-middle attack

The differential meet-in-the-middle technique interprets the meet-in-the-middle attack in the differential view. Combined with the statistical tool, it can analyse more steps of hash functions. The differential meet-in-the-middle attack with bicliques is summarized as follows (see also **Fig. 1**).

- **Split the compression function and construct the linear spaces**. Split the compression function CF into 3 parts, $CF = CF_3 \bullet CF_2 \bullet CF_1$. Choose LD_1 and LD_2 such that $LD_1 \cap LD_2 = \{0\}$.
- **Construct the bicliques in CF_3**. For all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, using the biclique search algorithm constructs the bicliques such that $P[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, Q[\delta_1])$.
- **Search for the mask vector T**. Randomly choose M and $Q[0]$, compute $P = CF_3(M, Q[0])$. For each $\delta_1 \in LD_1$ and $\delta_2 \in LD_2$, search for the difference Δ_1 and Δ_2, such that $\Delta_1 = CF_1(M, P[0]) \oplus CF_1(M \oplus \delta_1, P[0])$ and $\Delta_2 = CF_2^{-1}(M, Q[0]) \oplus CF_2^{-1}(M \oplus \delta_2, Q[0])$ hold with high probability. Then for each $(\delta_1, \delta_2) \in LD_1 \times LD_2$, compute $\Delta = CF_1(M \oplus \delta_1, P[0]) \oplus \Delta_1 \oplus CF_2^{-1}(M \oplus \delta_2, Q[0]) \oplus \Delta_2$. For each bit i in Δ, count the number of $\Delta[i] = 1$. Set those d bits of T to 1 which have the lowest counters and the other bits of T to 0.
- **Search for the candidate preimage**. Compute the list $L_1 = CF_1(M \oplus \delta_2, P[\delta_2]) \oplus \Delta_2$ and the list $L_2 = CF_2^{-1}(M \oplus \delta_1, Q[\delta_1]) \oplus \Delta_1$. If $L_1 =_T L_2$, then $M \oplus \delta_1 \oplus \delta_2$ is a candidate pseudo-preimage. For a mask vector $T \in \{0,1\}^n$, $L_1 =_T L_2$ means $T \wedge (L_1 \oplus L_2) = 0$, where $\wedge$ denotes bitwise AND.

The error is defined as: $M \oplus \delta_1 \oplus \delta_2$ is a preimage, but we reject it as false. Assume $\overline{\alpha}$ is the the error probability, Γ is the cost of CF_1 and CF_2, and Γ_{re} is the cost of retesting a candidate preimage. The total complexity of the differential meet-in-the-middle preimage attack is $2^{n-d} \times (\Gamma + \Gamma_{re}) / (1-\overline{\alpha})$.

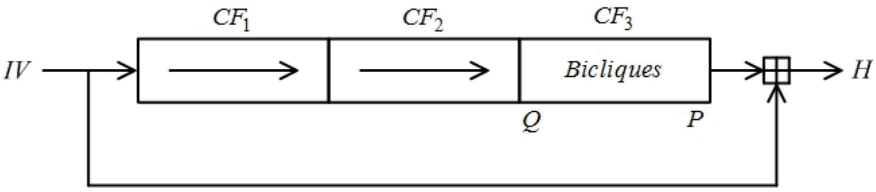

Fig. 1. Schematic view of the differential meet-in-the-middle attack with bicliques

2.3 The biclique search algorithm

A biclique for CF_3 is a tuple $\{M, LD_1, LD_2, Q, P\}$, where M is a message, such that for all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, $P[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, Q[\delta_1])$. Assume $\overline{CF_3}$ is linearized by CF_3, i.e., replacing + by $\oplus$ and setting the constants to 0. The bicliques can be searched as follows.

- Linearize CF_3 and split $\overline{CF_3}$ into 2 parts, $\overline{CF_3} = \overline{CF_3^2} \bullet \overline{CF_3^1}$. Assume $FT(\delta_2)$ is the differences of the outputs of $\overline{CF_3^1}$ corresponding to δ_2, and $BT(\delta_1)$ is the differences of the outputs of $\overline{(CF_3^2)^{-1}}$ corresponding to δ_1.
- For each $\delta_1 \in LD_1$ and $\delta_2 \in LD_2$ calculate $BT(\delta_1)$ and $FT(\delta_2)$, such that $BT(\delta_1) = \overline{(CF_3^2)^{-1}}(\delta_1, 0)$ and $FT(\delta_2) = \overline{CF_3^1}(\delta_2, 0)$, deduce the sufficient conditions to make sure the output differences of all non-linear Boolean functions are zero.
- Assume CV is the intermediate chaining value which connects CF_3^1 and CF_3^2. Randomly choose M and CV, set a part of the sufficient conditions.
- For each $\delta_1 \in LD_1$ and $\delta_2 \in LD_2$, calculate $Q[\delta_1]$ and $P[\delta_2]$, such that $Q[\delta_1] = (CF_3^1)^{-1}(M \oplus \delta_2, CV \oplus FT(\delta_2) \oplus BT(\delta_1))$ holds for every $\delta_2 \in LD_2$ and $P[\delta_2] = CF_3^2(M \oplus \delta_1, CV \oplus FT(\delta_2) \oplus BT(\delta_1))$ holds for every $\delta_1 \in LD_1$.
- If $\{M, LD_1, LD_2, Q, P\}$ constitute a biclique, return (M, Q[0]). Otherwise, repeat the above steps until a biclique is constructed.

3. Description of RIPEMD-160 and HAS-160

In this section, we briefly introduce our targeted hash functions. Both of the hash functions adopt the Merkel-Damgård structure and process 512-bit input message blocks and produce a 160-bit hash value. They first call the message padding procedure. The message padding procedure ensures the padded message is a multiple of 512 bits. For an l-bit message, append the bit "1" to the end of the message, followed by k "0" bits, where k is the smallest non-negative solution to the equation $l + k + 1 \equiv 448 \bmod 512$. Then append the 64-bit block that is equal to the number l expressed using a binary representation. The words m_{14} and m_{15} in the last message block represent the message length. Then, using the compression function, they update the initial value and produce the hash value. The compression function of RIPEMD-160 and HAS-160 consist of a message expansion function and a state update transformation, see the details in [14] and [20].

3.1 Description of RIPEMD-160

In this part, the compression function of RIPEMD-160 is described.

3.1.1 Message Expansion

The message expansion of RIPEMD-160 splits the 512-bit message block M into 16 words m_0, …, m_{15}, and expands them into 160 expanded message words w_i^L and w_i^R $(0 \le i \le 79)$. The expansion is specified in **Table 2**.

Table 2. Message order and rotation of RIPEMD-160

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
w_i^L	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	7	4	13	1
w_i^R	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12	6	11	3	7
s_i^L	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8	7	6	8	13
s_i^R	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6	9	13	15	7
i	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
w_i^L	10	6	15	3	12	0	9	5	2	14	11	8	3	10	14	4	9	15	8	1
w_i^R	0	13	5	10	14	15	8	12	4	9	1	2	15	5	1	3	7	14	6	9
s_i^L	11	9	7	15	7	12	15	9	11	7	13	12	11	13	6	7	14	9	13	15
s_i^R	12	8	9	11	7	7	12	7	6	15	13	11	9	7	15	11	8	6	6	14
i	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
w_i^L	2	7	0	6	13	11	5	12	1	9	11	10	0	8	12	4	13	3	7	15
w_i^R	11	8	12	2	10	0	4	13	8	6	4	1	3	11	15	0	5	12	2	13
s_i^L	14	8	13	6	5	12	7	5	11	12	14	15	14	15	9	8	9	14	5	6
s_i^R	12	13	5	14	13	13	7	5	15	5	8	11	14	14	6	14	6	9	12	9
i	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
w_i^L	14	5	6	2	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13
w_i^R	9	7	10	14	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11
s_i^L	8	6	5	12	9	15	5	11	6	8	13	12	5	12	13	14	11	8	5	6
s_i^R	12	5	15	8	8	5	12	9	12	5	14	6	8	13	6	5	15	13	11	11

3.1.2 State Update Transformation

The state update transformation of RIPEMD-160 starts from an initial value of five 32-bit words A_0, B_0, C_0, D_0, E_0 and updates them in 80 steps. For the i-th $(0 \le i \le 79)$ step, the 32-bit words w_i^L and w_i^R are used and the state variables A_i^L, B_i^L, C_i^L, D_i^L, E_i^L, A_i^R, B_i^R, C_i^R, D_i^R, E_i^R are update as:

Left branch:

$$B_{i+1}^L = (A_i^L + F_i^L(B_i^L, C_i^L, D_i^L) + w_i^L + k_i^L) <<< s_i^L + E_i^L,$$

$$A_{i+1}^L = E_i^L, C_{i+1}^L = B_i^L, D_{i+1}^L = C_i^L <<< 10, E_{i+1}^L = D_i^L.$$

Right branch:

$$B_{i+1}^R = (A_i^R + F_i^R(B_i^R, C_i^R, D_i^R) + w_i^R + k_i^R) <<< s_i^R + E_i^R,$$

$$A_{i+1}^R = E_i^R, C_{i+1}^R = B_i^R, D_{i+1}^R = C_i^R <<< 10, E_{i+1}^R = D_i^R.$$

The bitwise Boolean function $F_i^L(B_i^L, C_i^L, D_i^L)$, $F_i^R(B_i^R, C_i^R, D_i^R)$, round constant k_i^L, k_i^R are given in **Table 3** and the rotational constant s_i^L, s_i^R are specified in **Table 2**.

If M is the last block, $(B_0 + C_{80}^L + D_{80}^R$, $C_0 + D_{80}^L + E_{80}^R$, $D_0 + E_{80}^L + A_{80}^R$, $E_0 + A_{80}^L + B_{80}^R$, $A_0 + B_{80}^L + C_{80}^R)$ is the hash value, otherwise as the inputs of the next message block.

Table 3. The Boolean function and round constant of RIPEMD-160

Step	$F_i^L(B_i^L, C_i^L, D_i^L)$	k_i^L	$F_i^R(B_i^R, C_i^R, D_i^R)$	k_i^R
$0 \le i \le 15$	$B_i^L \oplus C_i^L \oplus D_i^L$	0x00000000	$B_i^R \oplus (C_i^R \vee (\neg D_i^R))$	0x50a28be6
$16 \le i \le 31$	$(B_i^L \wedge C_i^L) \vee ((\neg B_i^L) \wedge D_i^L)$	0x5a827999	$(B_i^R \wedge D_i^R) \vee (C_i^R \wedge (\neg D_i^R))$	0x5c4dd124
$32 \le i \le 47$	$(B_i^L \vee (\neg C_i^L)) \oplus D_i^L$	0x6ed9eba1	$(B_i^R \vee (\neg C_i^R)) \oplus D_i^R$	0x6d703ef3
$48 \le i \le 63$	$(B_i^L \wedge D_i^L) \vee (C_i^L \wedge (\neg D_i^L))$	0x8f1bbcdc	$(B_i^R \wedge C_i^R) \vee ((\neg B_i^R) \wedge D_i^R)$	0x7a6d76e9
$64 \le i \le 79$	$B_i^L \oplus (C_i^L \vee (\neg D_i^L))$	0xa953fd4e	$B_i^R \oplus C_i^R \oplus D_i^R$	0x00000000

3.2 Description of HAS-160

In this part, the compression function of HAS-160 is described.

3.2.1 Message Expansion

The message expansion of HAS-160 splits the 512-bit message block M into 16 words m_0, …, m_{15}, and expands them into 80 expanded message words $w_i\,(0 \le i \le 79)$. The expansion is specified in **Table 4**.

Table 4. Message expansion of HAS-160

i	0	1	2	3	4	5	6	7	8	9
w_i	$m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$	m_{15}	m_{15}	m_{15}	m_{15}	$m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$	m_4	m_5	m_6	m_7
i	10	11	12	13	14	15	16	17	18	19
w_i	$m_0 \oplus m_1 \oplus m_2 \oplus m_3$	m_8	m_9	m_{10}	m_{11}	$m_4 \oplus m_5 \oplus m_6 \oplus m_7$	m_{12}	m_{13}	m_{14}	m_{15}
i	20	21	22	23	24	25	26	27	28	29
w_i	$m_{11} \oplus m_{14} \oplus m_1 \oplus m_4$	m_3	m_6	m_9	m_{12}	$m_7 \oplus m_{10} \oplus m_{13} \oplus m_0$	m_{15}	m_2	m_5	m_8
i	30	31	32	33	34	35	36	37	38	39
w_i	$m_3 \oplus m_6 \oplus m_9 \oplus m_{12}$	m_{11}	m_{14}	m_1	m_4	$m_{15} \oplus m_2 \oplus m_5 \oplus m_8$	m_7	m_{10}	m_{13}	m_0
i	40	41	42	43	44	45	46	47	48	49
w_i	$m_4 \oplus m_{13} \oplus m_6 \oplus m_{15}$	m_{12}	m_5	m_{14}	m_7	$m_8 \oplus m_1 \oplus m_{10} \oplus m_3$	m_0	m_9	m_2	m_{11}
i	50	51	52	53	54	55	56	57	58	59
w_i	$m_{12} \oplus m_5 \oplus m_{14} \oplus m_7$	m_4	m_{13}	m_6	m_{15}	$m_0 \oplus m_9 \oplus m_2 \oplus m_{11}$	m_8	m_1	m_{10}	m_3
i	60	61	62	63	64	65	66	67	68	69
w_i	$m_{15} \oplus m_{10} \oplus m_5 \oplus m_0$	m_7	m_2	m_{13}	m_8	$m_{11} \oplus m_6 \oplus m_1 \oplus m_{12}$	m_3	m_{14}	m_9	m_4
i	70	71	72	73	74	75	76	77	78	79
w_i	$m_7 \oplus m_2 \oplus m_{13} \oplus m_8$	m_{15}	m_{10}	m_5	m_0	$m_3 \oplus m_{14} \oplus m_9 \oplus m_4$	m_{11}	m_6	m_1	m_{12}

3.2.2 State Update Transformation

The state update transformation of HAS-160 starts from an initial value of five 32-bit words A_0, B_0, C_0, D_0, E_0 and updates them in 80 steps. For the i-th $(0 \le i \le 79)$ step, the 32-bit word w_i is used and the state variables A_i, B_i, C_i, D_i, E_i are update as:

$$A_{i+1} = (A_i \lll s_i^1) + F_i(B_i, C_i, D_i) + E_i + w_i + k_i,$$
$$B_{i+1} = A_i,\ C_{i+1} = B_i \lll s_i^2,\ D_{i+1} = C_i,\ E_{i+1} = D_i.$$

The bitwise Boolean function $F_i(B_i, C_i, D_i)$, round constant k_i and rotation const s_i^2 used in each step are given in **Table 5**. The rotational constant s_i^1 are given in **Table 6**.

Table 5. The Boolean function and round constant of HAS-160

Step	F_i (B_i, C_i, D_i)	k_i	s_i^2
$0 \le i \le 19$	$(B_i \wedge C_i) \vee ((\neg B_i) \wedge D_i)$	0x00000000	10
$20 \le i \le 39$	$B_i \oplus C_i \oplus D_i$	0x5a827999	17
$40 \le i \le 59$	$C_i \oplus (B_i \vee (\neg D_i))$	0x6ed9eba1	25
$60 \le i \le 79$	$B_i \oplus C_i \oplus D_i$	0x8f1bbcdc	30

Table 6. The shift s_i^1 of HAS-160

Step	s_i^1																			
i mod 20	5	11	7	15	6	13	8	14	7	12	9	11	8	15	6	12	9	14	5	13

If M is the last block, ($A_{80}+A_0$, $B_{80}+B_0$, $C_{80}+C_0$, $D_{80}+D_0$, $E_{80}+E_0$) is the hash value, otherwise as the inputs of the next message block.

4. Attacks on step-reduced RIPEMD-160 and HAS-160

In this section, we describe the details of the preimage and pseudo-preimage attacks on step-reduced RIPEMD-160 and HAS-160. Assume $\Delta m_i = (a, b)$ means the bits from the a-th bit to the b-th bit of Δm_i take all possible values, and the other bits are 0. Δm_i can be represented by a linear space. $\overline{CF_1}$ and $\overline{CF_2^{-1}}$ are obtained from CF_1 and CF_2^{-1} respectively by linearizing the step function, i.e., replacing + by $\oplus$ and setting the constants and input chaining values of CF_1 and CF_2^{-1} to 0.

4.1 Preimage attack on 34-step RIPEMD-160 from the first step

4.1.1 Split the compression function and construct the linear spaces

The forward direction CF_1 is from step 1 to step 33 in left branch connecting with the step of computing the hash value (marked with red dotted box in **Fig. 2**) and the linear space of this part is $LD_1 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_0 = (27, 31), \Delta m_i = 0, 1 \le i \le 15\}$. The basis of LD_1 is as follows:

Basis 1:
0x08000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 2:
0x10000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 3:
0x20000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 4:

0x40000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 5:

0x80000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Let CV_i^L and CV_i^R denote the inputs of the *i*-th step of left branch and right branch respectively. Then the difference corresponding to CV_{25}^L of $CF_1(M \oplus \delta_1, CV_1^L)$ and $CF_1(M, CV_1^L)$ is always zero. The backward direction CF_2^{-1} is from step 6 to step 33 in right branch and the linear space is $LD_2 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_2 = (4,8), \Delta m_i = 0, 0 \le i \le 15, i \ne 2\}$. The basis of LD_2 is as follows:

Basis 1:

0x00000000, 0x00000000, 0x00000010, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 2:

0x00000000, 0x00000000, 0x00000020, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 3:

0x00000000, 0x00000000, 0x00000040, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 4:

0x00000000, 0x00000000, 0x00000080, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 5:

0x00000000, 0x00000000, 0x00000100, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

The difference corresponding to CV_{31}^R of $CF_2^{-1}(M \oplus \delta_2, CV_6^R)$ and $CF_2^{-1}(M, CV_6^R)$ is always zero. The biclique CF_3 covers step 0 in left branch connecting with step 0 to step 5 in right branch. It is noted that the initial value IV is an intermediate value in the bicliques, and IV also affects the last step (marked with red dotted box in **Fig. 2**), so the truncation mask vector can only be chosen in a smaller scope, e.g., the unchanged bits of IV. Therefore, we choose the linear spaces LD_1 and LD_2 as above so that we can get a proper truncation mask vector to make sure the error probability of the attack is as small as possible. The schematic view of the preimage attack on 34-step RIPEMD-160 is shown in **Fig. 2**.

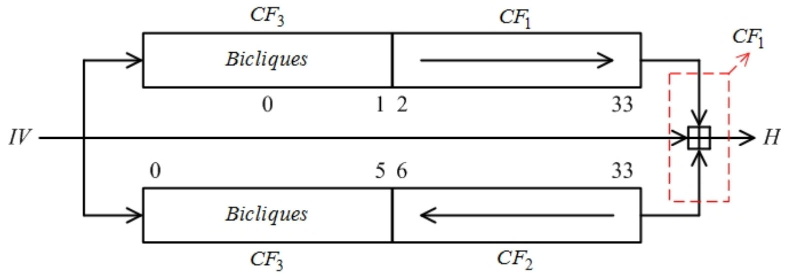

Fig. 2. Schematic view of the preimage attack on 34-step RIPEMD-160

4.1.2 Construct 3.5-step bicliques

The bicliques which are constructed for CF_3 cover step 5 to step 0 in right branch connecting with step 0 in left branch (which equals to 3.5 steps), such that for all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, $CV_1^L[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, CV_6^R[\delta_1])$. The parameters used in Section 2.3 are set as follows: CF_3^1 covers step 0 in left branch connecting with step 0 to step 2 in right branch and CF_3^2 covers step 3 to step 5 in right branch, $CV = (A_3^R, B_3^R, C_3^R, D_3^R, E_3^R)$. For $\delta_2 \in LD_2$ and $\delta_1 \in LD_1$, we can deduce a set of sufficient conditions that make sure $FT(\delta_2)$ and $BT(\delta_1)$ hold. Then the bicliques can be constructed by using the biclique search algorithm in Section 2.3. Note that in the biclique search algorithm, we only set the sufficient conditions in steps 1, 2 and 3 in right branch. By setting those conditions, a biclique can be obtained in a few seconds on a PC. So the complexity of constructing a biclique is negligible. An example of the 3.5-step bicliques is shown in **Table 7**.

Table 7. An example of 3.5-step bicliques in RIPEMD-160

word	m_0		m_2	m_5	m_7	m_9	m_{14}
value	0x07b30181		0x5effd80e	0xf7962189	0xff1152bb	0xa9bc41d8	0x00000000
CV_6^R	A		B	C	D	E	
value	0x8fcfef67		0xc592715e	0x26d8dac7	0x71f4450e	0x6c6740a0	

4.1.3 Search for the mask vector and estimate the error probability

The mask vector is $T_{34} = (0, 0xf80, 0, 0, 0)$. By extensive experiments, the total test number is 2^{26} and the error probability is about 0.331. The test algorithm is the same as Knellwolf and Khovratovich's Algorithm 3 [7].

4.1.4 The attack procedure

When the bicliques are obtained, we can get $CV_1^L[\delta_2]$ and $CV_6^R[\delta_1]$, then compute $L_1 = CF_1(M \oplus \delta_2, CV_1^L[\delta_2]) \oplus \Delta_2$ and $L_2 = CF_2^{-1}(M \oplus \delta_1, CV_6^R[\delta_1]) \oplus \Delta_1$. $\Delta_1 = \overline{CF_1}(\delta_1, 0)$ and $\Delta_2 = \overline{CF_2^{-1}}(\delta_2, 0)$, where CV_1^L, CV_6^R, k_i^L and k_i^R are 0. We need to calculate 6 steps (step 25 to step 33 in left branch add step 31 to step 33 in right branch, which equals to $6=(9+3)/2$ steps) to retest a candidate pseudo-preimage, $\Gamma_{re}=6\Gamma/34$, and the complexity of constructing a biclique is negligible. Thus, the complexity of the pseudo-preimage attack is $2^{155.81}\approx 2^{160-5}\times(1+6/34)/(1-0.331)$. Furthermore, this attack can be converted to a preimage attack (2 message blocks, with padding) with complexity $2^{158.91}$.

4.2 Preimage attack on 35-step RIPEMD-160 from the second step

4.2.1 Split the compression function and construct the linear spaces

The forward direction CF_1 is from step 3 to step 35 in left branch connecting with the step of computing the hash value (marked with red dotted box in **Fig. 3**) and the linear space of this part is $LD_1=\{\Delta m_0\|...\|\Delta m_{15}\mid\Delta m_2=(27,31),\Delta m_i=0,0\le i\le 15,i\ne 2\}$. The basis of LD_1 is as follows:

Basis 1:
0x00000000, 0x00000000, 0x08000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000
Basis 2:
0x00000000, 0x00000000, 0x10000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000
Basis 3:
0x00000000, 0x00000000, 0x20000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000
Basis 4:
0x00000000, 0x00000000, 0x40000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000
Basis 5:
0x00000000, 0x00000000, 0x80000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

The difference corresponding to CV_{28}^{L} of $CF_1(M\oplus\delta_1,CV_3^{L})$ and $CF_1(M,CV_3^{L})$ is always zero. The backward direction CF_2^{-1} is from step 6 to step 35 in right branch and the linear space of this part is $LD_2=\{\Delta m_0\|...\|\Delta m_{15}\mid\Delta m_2=(13,17),\Delta m_i=0,0\le i\le 15,i\ne 2\}$. The basis of LD_2 is as follows:

Basis 1:

0x00000000, 0x00000000, 0x00002000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 2:

0x00000000, 0x00000000, 0x00004000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 3:

0x00000000, 0x00000000, 0x00008000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 4:

0x00000000, 0x00000000, 0x00010000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 5:

0x00000000, 0x00000000, 0x00020000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

The difference corresponding to CV_{31}^{R} of $CF_2^{-1}(M \oplus \delta_2, CV_6^R)$ and $CF_2^{-1}(M, CV_6^R)$ is always zero. The biclique CF_3 covers step 1 and step 2 in left branch connecting with step 1 to step 5 in right branch. Because the initial value is an intermediate value in the bicliques and affects the last step (marked with red dotted box in **Fig. 3**), the mask vector can only be chosen in a small scope, e.g., the unchanged least significant bits of the initial value. Therefore, we choose the linear spaces LD_1 and LD_2 as above so that we can get a proper mask vector to make the error probability as low as possible. The schematic view of the preimage attack on 35-step RIPEMD-160 is shown in **Fig. 3**.

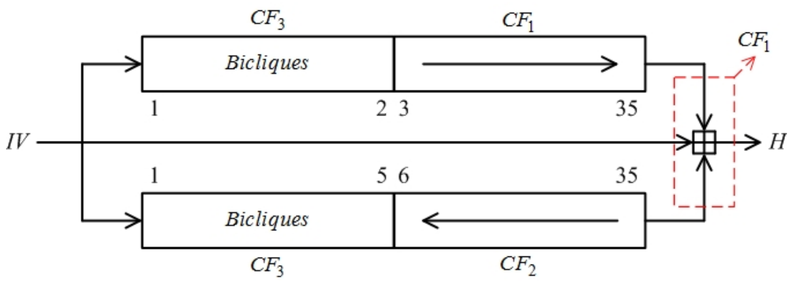

Fig. 3. Schematic view of the preimage attack on 35-step RIPEMD-160

4.2.2 Construct 3.5-step bicliques

The bicliques which are constructed for CF_3 cover step 5 to step 1 in right branch connecting with step 1 to step 2 in left branch (which equals to 3.5 steps), such that for all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, $CV_3^L[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, CV_6^R[\delta_1])$. The parameters used in Section 2.3 are set as follows: CF_3^1 covers step 5 to step 3 in right branch and CF_3^2 covers step 2 to step 1 in right branch connecting with step 1 to step 2 in left branch, $CV = (A_3^R, B_3^R, C_3^R, D_3^R, E_3^R)$. $FT(\delta_2)$ is deduced as follows:

$$A_3^R \wedge (0x3e00000) = (((m_2 \oplus \delta_2) \wedge (0x3e000)) \oplus (0x3e000)) <<< 8,$$

$$D_3^R \wedge (0x3e000) = ((m_2 \oplus \delta_2) \wedge (0x3e000)) \oplus (0x3e000),$$

$$E_3^R \wedge (0x1f) = ((m_2 \oplus \delta_2) \wedge (0x3e000)) <<< 19.$$

Condition that satisfies $FT(\delta_2)$ is $C_3^R \wedge (0x3e000) = 0x3e000$. $BT(\delta_1)$ is deduced as follows:

$$B_3^R \wedge (0x8000000f) = (((m_2 \oplus \delta_1) \wedge (0xf8000000)) \oplus (0xf8000000)) <<< 4,$$

$$C_3^R \wedge (0x7c00000) = ((m_2 \oplus \delta_1) \wedge (0xf8000000)) <<< 27,$$

$$C_3^R \wedge (0xf8000000) = ((m_2 \oplus \delta_1) \wedge (0xf8000000)) \oplus (0xf8000000).$$

Conditions that satisfy $BT(\delta_1)$ are $A_3^R \wedge (0x8000000f) = 0$, $B_3^R \wedge (0xf8) = 0xf8$, $E_3^R \wedge (0xffc00000) = 0xffc00000$.

By setting those conditions, a biclique example can be obtained less than 1 second on a PC. So the complexity of constructing a biclique is negligible. An example of the 3.5-step biclique is shown in **Table 8**. Note that to satisfy the padding process, $M_{15} = 0x3bf$ (the message length is 959 = 512 + 447) and the least significant bit of M_{13} is set to 1.

Table 8. An example of 3.5-step bicliques in RIPEMD-160

word	m_0	m_1	m_2	m_7	m_9	m_{14}
value	0x74a2aec5	0xe84269fa	0xeef5d697	0x4e5f2844	0x7e98f8a9	0x00000000
CV_6^R	A	B	C	D	E	
value	0x8fbcac5d	0x140846d5	0xdf46e46e	0x18115673	0xee7be4c7	

4.2.3 Search for the mask vector and estimate the error probability

One can check that the 4 least significant bits of D_1^R and 2 least significant bits of E_1^R are unchanged in bicliques and the error probability in these bits are acceptable. So the mask vector is $T_{35} = (0xf, 0x2, 0, 0, 0)$. By extensive experiments, the total test number is 2^{30} and the error probability is about 0.65.

4.2.4 The attack procedure

When the bicliques are obtained, we can get $CV_3^L[\delta_2]$ and $CV_6^R[\delta_1]$, then compute $L_1 = CF_1(M \oplus \delta_2, CV_3^L[\delta_2]) \oplus \Delta_2$ and $L_2 = CF_2^{-1}(M \oplus \delta_1, CV_6^R[\delta_1]) \oplus \Delta_1$. $\Delta_1 = \overline{CF_1}(\delta_1, 0)$ and $\Delta_2 = \overline{CF_2^{-1}}(\delta_2, 0)$, where CV_3^L, CV_6^R, k_i^L and k_i^R are 0. We need to calculate 6.5 steps (step 28 to step 35 in left branch add step 31 to step 35 in right branch, which equals to $6.5 = (8+5)/2$ steps) to retest a candidate pseudo-preimage, $\Gamma_{re} = 6.5\Gamma / 35$, and the complexity of constructing a biclique is negligible. Thus, the complexity of the pseudo-preimage attack is $2^{156.75} \approx 2^{160-5} \times (1+6.5/35)/(1-0.65)$. Furthermore, this attack can be converted to a preimage attack (2 message blocks, with padding) with complexity $2^{159.38}$.

4.3 Pseudo-preimage attack on 71-step HAS-160 from the first step

4.3.1 Split the compression function and construct the linear spaces

The forward direction CF_1 is from step 59 to step 70 connecting with step 0 to step 20 and the linear space of this part is $LD_1 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_{10} = \Delta m_{11} = \Delta m_{12} = \Delta m_{15} = (26, 28),$ $\Delta m_i = 0, 0 \le i \le 15, i \ne 10,11,12,15\}$. The basis of LD_1 is as follows:

Basis 1:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x04000000, 0x04000000,
0x04000000, 0x00000000, 0x00000000, 0x04000000

Basis 2:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x08000000, 0x08000000,
0x08000000, 0x00000000, 0x00000000, 0x08000000

Basis 3:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x10000000, 0x10000000,
0x10000000, 0x00000000, 0x00000000, 0x10000000

Let CV_i denote the input of the *i*-th step of HAS-160. In this case, the input difference of the 13-th step of $CF_1(M \oplus \delta_1, CV_{59})$ and $CF_1(M, CV_{59})$ is zero. The backward direction CF_2^{-1} is from step 52 to step 21 and the linear space of this part is $LD_2 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_3 = \Delta m_6 = \Delta m_8 = \Delta m_{15} = (29, 31), \Delta m_i = 0, 0 \le i \le 15, i \ne 3,6,8,15\}$. The basis of LD_2 is as follows:

Basis 1:

0x00000000, 0x00000000, 0x00000000, 0x20000000,
0x00000000, 0x00000000, 0x20000000, 0x00000000,
0x20000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x20000000

Basis 2:

0x00000000, 0x00000000, 0x00000000, 0x40000000,
0x00000000, 0x00000000, 0x40000000, 0x00000000,
0x40000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x40000000

Basis 3:

0x00000000, 0x00000000, 0x00000000, 0x80000000,
0x00000000, 0x00000000, 0x80000000, 0x00000000,
0x80000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x80000000

Obviously, the input difference of the 30-th step of $CF_2^{-1}(M \oplus \delta_2, CV_{53})$ and $CF_2^{-1}(M, CV_{53})$ is zero. The bicliques CF_3 is from step 53 to step 58, and the schematic view of the pseudo-preimage attack on 71-step HAS-160 is shown in **Fig. 4**.

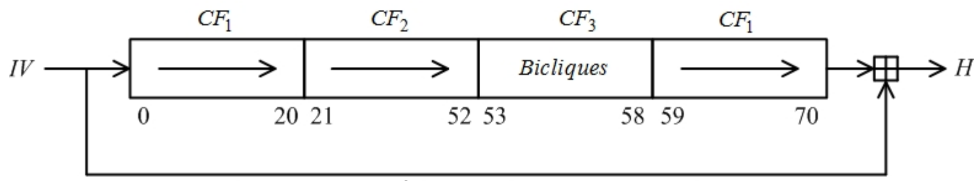

Fig. 4. Schematic view of the pseudo-preimage attack on HAS-160

4.3.2 Construct 6-step bicliques

The bicliques which are constructed for CF_3 cover step 53 to step 58, such that for all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, $CV_{59}[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, CV_{53}[\delta_1])$. The parameters used in Section 2.3 are set as follows: CF_3^1 covers step 53 to step 55 and CF_3^2 covers step 56 to step 58, $CV = (A_{56}, B_{56}, C_{56}, D_{56}, E_{56})$. For $\delta_2 \in LD_2$ and $\delta_1 \in LD_1$, we can deduce a set of sufficient conditions that make sure $FT(\delta_2)$ and $BT(\delta_1)$ hold. Then the bicliques can be constructed by using the biclique search algorithm in Section 2.3. Note that in the biclique search algorithm, we only set the sufficient conditions in steps 55, 56 and 57 in right branch. By setting those conditions, a biclique can be obtained in a few seconds on a PC. So the complexity of constructing a biclique is negligible. An example of the 6-step biclique is shown in **Table 9**.

Table 9. An example of the 6-step bicliques in HAS-160

word	m_1	m_6	m_8	m_{10}	m_{15}	$m_0 \oplus m_9 \oplus m_2 \oplus m_{11}$
value	0x33de090e	0x092798cd	0x1f5d84c6	0xe053d3e1	0x00ba96c2	0xdcad415a
CV_{53}	A	B	C	D	E	
value	0x8fc640fe	0xd387c000	0x1dcc8b3b	0xcc96b2a0	0x7cb3634a	

4.3.3 Search for the mask vector and estimate the error probability

The mask vector is $T_{42} = (0,0,0x86,0,0)$. By extensive experiments, the total test number is 2^{26} and the error probability is about 0.429.

4.3.4 The attack procedure

When the bicliques are obtained, we can get $CV_{59}[\delta_2]$ and $CV_{53}[\delta_1]$, and compute $L_1 = CF_1(M \oplus \delta_2, CV_{59}[\delta_2]) \oplus \Delta_2$ and $L_2 = CF_2^{-1}(M \oplus \delta_1, CV_{53}[\delta_1]) \oplus \Delta_1$. $\Delta_1 = \overline{CF_1}(\delta_1, 0)$

and $\Delta_2 = \overline{CF_2^{-1}}(\delta_2, 0)$, where CV_{53}, CV_{59} and k_i are 0. We need to calculate 18 steps (step 12 to step 29) to retest a candidate pseudo-preimage, $\Gamma_{re} = 18\Gamma / 71$, and the complexity of constructing a biclique is negligible. Thus, the complexity of the pseudo-preimage attack is $2^{158.13} \approx 2^{160-3} \times (1 + 18/71)/(1 - 0.429)$.

4.4 Preimage attack on 71-step HAS-160 from the seventh step

4.4.1 Split the compression function and construct the linear spaces

The forward direction CF_1 is from step 14 to step 42 and the linear space of this part is $LD_1 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_0 = \Delta m_{10} = (27, 31), \Delta m_i = 0, 0 \le i \le 15, i \ne 0,10\}$. The basis of LD_1 is as follows:

Basis 1:

0x08000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x08000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 2:

0x10000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x10000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 3:

0x20000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x20000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 4:

0x40000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x40000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

Basis 5:

0x80000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x80000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000

In this case, the input difference of the 37-th step of $CF_1(M \oplus \delta_1, CV_{14})$ and $CF_1(M, CV_{14})$ is zero. The backward direction CF_2^{-1} is from step 7 to step 6 connecting with step 76 to step 43 and the linear space of this part is $LD_2 = \{\Delta m_0 \| ... \| \Delta m_{15} \mid \Delta m_6 = \Delta m_{12} = (27, 31), \Delta m_i = 0, 0 \le i \le 15, i \ne 6,12\}$. The basis of LD_2 is as follows:

Basis 1:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x08000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x08000000, 0x00000000, 0x00000000, 0x00000000

Basis 2:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x10000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x10000000, 0x00000000, 0x00000000, 0x00000000

Basis 3:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x20000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x20000000, 0x00000000, 0x00000000, 0x00000000

Basis 4:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x40000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x40000000, 0x00000000, 0x00000000, 0x00000000

Basis 5:

0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000, 0x80000000, 0x00000000,
0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x80000000, 0x00000000, 0x00000000, 0x00000000

Obviously, the input difference of the 54-th step of $CF_2^{-1}(M \oplus \delta_2, CV_8)$ and $CF_2^{-1}(M, CV_8)$ is zero. The bicliques CF_3 is from step 8 to step 13, and the schematic view of the preimage attack on 71-step HAS-160 is shown in **Fig. 5**.

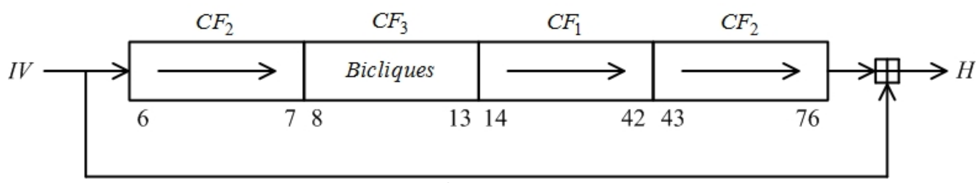

Fig. 5. Schematic view of the preimage attack on HAS-160

4.4.2 Construct 6-step bicliques

The bicliques which are constructed for CF_3 cover step 8 to step 13, such that for all $(\delta_1, \delta_2) \in LD_1 \times LD_2$, $CV_8[\delta_2] = CF_3(M \oplus \delta_1 \oplus \delta_2, CV_{14}[\delta_1])$. The parameters used in Section 2.3 are set as follows: CF_3^1 covers step 8 to step 10 and CF_3^2 covers step 11 to step 13, $CV = (A_{11}, B_{11}, C_{11}, D_{11}, E_{11})$. $FT(\delta_2)$ is deduced as follows:

$$A_{11} \wedge (0x1f0000) = ((m_6 \oplus \delta_2) \wedge (0xf8000000)) <<< 21,$$
$$B_{11} \wedge (0xf80) = ((m_6 \oplus \delta_2) \wedge (0xf8000000)) <<< 12,$$
$$C_{11} \wedge (0x3e0) = ((m_6 \oplus \delta_2) \wedge (0xf8000000)) <<< 10.$$

Condition that satisfies $FT(\delta_2)$ is $D_{11} \wedge (0xf8000000) = E_{11} \wedge (0xf8000000)$. $BT(\delta_1)$ is deduced as follows:

$$C_{11} \wedge (0xf8000000) = ((m_0 \oplus \delta_1) \wedge (0xf8000000)) \oplus (0xf8000000).$$

Conditions that satisfy $BT(\delta_1)$ are $A_{11} \wedge (0xf8000000) = 0xf8000000$, $B_{11} \wedge (0xf8000000)$ = 0. Extra condition is $m_1 \wedge (0xf8000000) = (m_2 \oplus m_3) \wedge (0xf8000000)$. By setting those conditions, a biclique example can be obtained with probability 1. So the complexity of constructing a biclique is negligible. An example of the 6-step biclique is shown in **Table 10**.

Table 10. An example of the 6-step bicliques in HAS-160

word	m_6	m_7	$m_0 \oplus m_1 \oplus m_2 \oplus m_3$	m_8	m_9	m_{10}
value	0x49c1e50b	0x6c67d9a3	0x63a5dd43	0xd8b52f39	0xbef4c8f3	0x39b89099
CV_8	A	B	C	D	E	
value	0x8f7b70f3	0x873ae2a1	0x19f7ee05	0xf467e82d	0xd3fe5704	

4.4.3 Search for the mask vector and estimate the error probability

The mask vector is $T_{42} = (0,0x180,0x3,0x2,0)$. By extensive experiments, the total test number is 2^{30} and the error probability is about 0.35.

4.4.4 The attack procedure

When the bicliques are obtained, we can get $CV_8[\delta_2]$ and $CV_{14}[\delta_1]$, and compute $L_1 = CF_1(M \oplus \delta_2, CV_{14}[\delta_2]) \oplus \Delta_2$ and $L_2 = CF_2^{-1}(M \oplus \delta_1, CV_8[\delta_1]) \oplus \Delta_1$. $\Delta_1 = \overline{CF_1}(\delta_1,0)$ and $\Delta_2 = \overline{CF_2^{-1}}(\delta_2,0)$, where CV_8, CV_{14} and k_i 0. We need to calculate 17 steps (step 37 to step 53) to retest a candidate pseudo-preimage, $\Gamma_{re} = 17\Gamma / 71$, and the complexity of constructing a biclique is negligible. Thus, the complexity of the pseudo-preimage attack is $2^{155.93} \approx 2^{160-5} \times (1+17/71)/(1-0.35)$. Furthermore, this attack can be converted to a preimage attack (2 message blocks, with padding) with complexity $2^{158.97}$.

5. Conclusion

In this paper, a preimage attack on 34-step RIPEMD-160 with message padding and a pseudo-preimage attack on 71-step HAS-160 without message padding are proposed. The former holds with a complexity of $2^{158.91}$, the latter holds with a complexity of $2^{158.13}$. Furthermore, we locate the linear spaces in another message words and exchange the bicliques construction process and the mask vector search process. A preimage attack on 35-step RIPEMD-160 with complexity $2^{159.38}$ and a preimage attack on 71-step HAS-160 with complexity $2^{158.97}$ are obtained. Both of the attacks satisfy the message padding. Future analysis should be able to explore how to construct bicliques including more steps so that preimage attacks can be implemented on more steps of hash functions.

References

[1] Xiaoyun Wang and Hongbo Yu, "How to break MD5 and other hash functions," in *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 19-35, May 22-26, 2005. Article (CrossRef Link)

[2] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet and William Jalby, "Collisions of SHA-0 and reduced SHA-1," in *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 36-57, May 22-26, 2005. Article (CrossRef Link)

[3] Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu, "Finding collisions in the full SHA-1," in *Proc. of the 25th Annual International Cryptology Conference*, pp. 17-36, August 14-18, 2005. Article (CrossRef Link)

[4] Kazumaro Aoki and Yu Sasaki, "Preimage attacks on one-block MD4, 63-step MD5 and more," in *Proc. of the 15th International Workshop on Selected Areas in Cryptography*, pp. 103-119, August 14-15, 2008. Article (CrossRef Link)

[5] Deukjo Hong, Bonwook Koo and Yu Sasaki, "Improved preimage attack for 68-step HAS-160," in *Proc. of the 12th International Conference on Information Security and Cryptology*, pp. 332-348, December 2-4, 2009. Article (CrossRef Link)

[6] Dmitry Khovratovich, Christian Rechberger and Alexandra Savelieva, "Bicliques for preimages: Attacks on Skein-512 and the SHA-2 family," in *Proc. of the 19th International Workshop on Fast Software Encryption*, pp. 244-263, March 19-21, 2012. Article (CrossRef Link)

[7] Simon Knellwolf and Dmitry Khovratovich, "New preimage attacks against reduced SHA-1," in *Proc. of the 32nd Annual Cryptology Conference*, pp. 367-383, August 19-23, 2012. Article (CrossRef Link)

[8] Chiaki Ohtahara, Yu Sasaki and Takeshi Shimoyama, "Preimage attacks on step-reduced RIPEMD-128 and RIPEMD-160," in *Proc. of the 6th International Conference on Information Security and Cryptology*, pp. 169-186, October 20-24, 2010. Article (CrossRef Link)

[9] Yu Sasaki and Kazumaro Aoki, "A preimage attack for 52-step HAS-160," in *Proc. of the 11th International Conference on Information Security and Cryptology*, pp. 302-317, December 3-5, 2008. Article (CrossRef Link)

[10] Yu Sasaki and Kazumaro Aoki, "Finding preimages in full MD5 faster than exhaustive search," in *Proc. of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 134-152, April 26-30, 2009. Article (CrossRef Link)

[11] Jian Guo, San Ling, Christian Rechberger and Huaxiong Wang, "Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2," in *Proc. of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 56-75, December 5-9, 2010. Article (CrossRef Link)

[12] Whitfield Diffie and Martin E. Hellman, "Exhaustive cryptanalysis of the NBS data encryption standard," *Computer*, vol. 10, no. 6, pp. 74-84, 1977. Article (CrossRef Link)

[13] Thomas Espitau, Pierre-Alain Fouque and Pierre Karpman, "Higher-order differential meet-in-the-middle preimage attacks on SHA-1 and BLAKE," in *Proc. of the 35th Annual Cryptology Conference*, pp. 683-701, August 16-20, 2015. Article (CrossRef Link)

[14] Hans Dobbertin, Antoon Bosselaers and Bart Preneel, "RIPEMD-160: A strengthened version of RIPEMD," in *Proc. of the 3rd International Workshop on Fast Software Encryption*, pp. 71-82, February 21–23 1996. Article (CrossRef Link)

[15] International Organization for Standardization, "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash functions (2004)," ISO/IEC 10118-3:2004, Article (CrossRef Link).

[16] Florian Mendel, Norbert Pramstaller, Christian Rechberger and Vincent Rijmen, "On the collision resistance of RIPEMD-160," in *Proc. of the 9th International Conference on Information Security*, pp. 101-116, August 30 - September 2, 2006. Article (CrossRef Link)

[17] Florian Mendel, Tomislav Nad, Stefan Scherz and Martin Schläffer, "Differential attacks on reduced RIPEMD-160," in *Proc. of the 15th International Conference on Information Security*, pp.

23-38, September 19-21, 2012. Article (CrossRef Link)
[18] Florian Mendel, Thomas Peyrin, Martin Schläffer, Lei Wang and Shuang Wu, "Improved cryptanalysis of reduced RIPEMD-160," in *Proc. of the 19th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 484-503, December 1-5, 2013. Article (CrossRef Link)
[19] Yu Sasaki and Lei Wang, "Distinguishers beyond three rounds of the RIPEMD-128/-160 compression functions," in *Proc. of the 10th International Conference on Applied Cryptography and Network Security*, pp. 275-292, June 26-29, 2012. Article (CrossRef Link)
[20] Telecommunications Technology Association, "Hash Function Standard Part 2: Hash Function Algorithm Standard, HAS-160 (2000)," TTAS.KO-12.0011/R2. Article (CrossRef Link).
[21] Aaram Yun, Soo Hak Sung, Sangwoo Park, Donghoon Chang, Seokhie Hong and Hong-Su Cho, "Finding collision on 45-step HAS-160," in *Proc. of the 8th International Conference on Information Security and Cryptology*, pp. 146-155, December 1-2, 2005. Article (CrossRef Link)
[22] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen and Xiuyuan Yu, "Cryptanalysis of the hash functions MD4 and RIPEMD," in *Proc. of the 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1-18, May 22-26, 2005. Article (CrossRef Link)
[23] Hong-Su Cho, Sangwoo Park, Soo Hak Sung and Aaram Yun, "Collision search attack for 53-step HAS-160," in *Proc. of the 9th International Conference on Information Security and Cryptology*, pp. 286-295, November 30 - December 1, 2006. Article (CrossRef Link)
[24] Florian Mendel and Vincent Rijmen, "Colliding message pair for 53-step HAS-160," in *Proc. of the 10th International Conference on Information Security and Cryptology*, pp. 324-334, November 29-30, 2007. Article (CrossRef Link)
[25] Florian Mendel, Tomislav Nad and Martin Schläffer, "Cryptanalysis of round-reduced HAS-160," in *Proc. of the 14th International Conference on Information Security and Cryptology*, pp. 33-47, November 30 - December 2, 2011. Article (CrossRef Link)
[26] Aleksandar Kircanski, Riham AlTawy and Amr M. Youssef, "Heuristic for finding compatible differential paths with application to HAS-160," in *Proc. of the 19th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 464-483, December 1-5, 2013. Article (CrossRef Link)
[27] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

Yanzhao Shen received the B.S. degree in School of Computer and Information Engineering form Henan University, Kaifeng, China, M.S. in School of Computer Science and Technology from Donghua University, Shanghai, China. Now he is a Ph.D student in School of Mathematics, Shandong University. His current research interests include cryptography, computer and network security.

Gaoli Wang received the B.S. degree in Fundamental Mathematics from Shandong University, Jinan, China, M.S. and Ph.D degree in Information Security from Shandong University, Jinan, China. She is currently an assistant professor at School of Computer Science and Software Engineering, East China Normal University. Her research interests include cryptography, computer and network security.