

## PDF hosted at the Radboud Repository of the Radboud University Nijmegen

The following full text is a publisher's version.

For additional information about this publication click this link.

<http://hdl.handle.net/2066/207825>

Please be advised that this information was generated on 2021-01-27 and may be subject to change.

# New Semi-Free-Start Collision Attack Framework for Reduced RIPEMD-160

Fukang Liu<sup>1,5</sup>, Christoph Dobraunig<sup>2</sup>, Florian Mendel<sup>3</sup>, Takanori Isobe<sup>4,5</sup>,  
Gaoli Wang<sup>1</sup> and Zhenfu Cao<sup>1,6</sup>

<sup>1</sup> Shanghai Key Laboratory of Trustworthy Computing,  
East China Normal University, Shanghai, China  
[liufukangs@163.com](mailto:liufukangs@163.com), [glwang, zfc@sei.ecnu.edu.cn](mailto:glwang@sei.ecnu.edu.cn)

<sup>2</sup> Radboud University, Nijmegen, The Netherlands  
[cdobraunig@cs.ru.nl](mailto:cdobraunig@cs.ru.nl)

<sup>3</sup> Infineon Technologies AG, Neubiberg, Germany  
[florian.mendel@gmail.com](mailto:florian.mendel@gmail.com)

<sup>4</sup> National Institute of Information and Communications Technology, Tokyo, Japan

<sup>5</sup> University of Hyogo, Hyogo, Japan  
[takanori.isobe@ai.u-hyogo.ac.jp](mailto:takanori.isobe@ai.u-hyogo.ac.jp)

<sup>6</sup> Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China

**Abstract.** RIPEMD-160 is a hash function published in 1996, which shares similarities with other hash functions designed in this time-period like MD4, MD5 and SHA-1. However, for RIPEMD-160, no (semi-free-start) collision attacks on the full number of steps are known. Hence, it is still used, e.g., to generate Bitcoin addresses together with SHA-256, and is an ISO/IEC standard. Due to its dual-stream structure, even semi-free-start collision attacks starting from the first step only reach 36 steps, which were firstly shown by Mendel et al. at Asiacrypt 2013 and later improved by Liu, Mendel and Wang at Asiacrypt 2017. Both of the attacks are based on a similar freedom degree utilization technique as proposed by Landelle and Peyrin at Eurocrypt 2013. However, the best known semi-free-start collision attack on 36 steps of RIPEMD-160 presented at Asiacrypt 2017 still requires  $2^{55.1}$  time and  $2^{32}$  memory. Consequently, a practical semi-free-start collision attack for the first 36 steps of RIPEMD-160 still requires a significant amount of resources. Considering the structure of these previous semi-free-start collision attacks for 36 steps of RIPEMD-160, it seems hard to extend it to more steps. Thus, we develop a different semi-free-start collision attack framework for reduced RIPEMD-160 by carefully investigating the message expansion of RIPEMD-160. Our new framework has several advantages. First of all, it allows to extend the attacks to more steps. Second, the memory complexity of the attacks is negligible. Hence, we were able to mount semi-free-start collision attacks on 36 and 37 steps of RIPEMD-160 with practical time complexity  $2^{41}$  and  $2^{49}$  respectively. Additionally, we describe semi-free-start collision attacks on 38 and 40 (out of 80) steps of RIPEMD-160 with time complexity  $2^{52}$  and  $2^{74.6}$ , respectively. To the best of our knowledge, these are the best semi-free-start collision attacks for RIPEMD-160 starting from the first step with respect to the number of steps, including the first practical colliding message pairs for 36 and 37 steps of RIPEMD-160.

**Keywords:** hash function · RIPEMD-160 · freedom degree utilization · semi-free-start collision attack

## 1 Introduction

In the 1990s, most popular hash functions, like MD4, MD5, SHA-0, RIPEMD-160 [DBP96] and SHA-1 followed a similar design strategy based on round functions involving modular additions, word-wise rotations, and XORs (ARX). For 4 out of the aforementioned hash functions, MD4 [Dob96, WLF<sup>+</sup>05], MD5 [WY05], SHA-0 [WYY05b] and SHA-1 [WYY05a, SBK<sup>+</sup>17] practical collision attacks were shown and thus have been phased out in most applications. However, if we look at RIPEMD-160, no collision attack on the full number of rounds is known. Moreover, RIPEMD-160 is still used in several applications, e.g., to generate Bitcoin addresses together with SHA-256, and is still an ISO/IEC standard. Hence, getting more insight into the security of RIPEMD-160 is of practical interest and importance.

In contrast to MD4, MD5, SHA-0 and SHA-1, the compression function of RIPEMD-160 is of a more complex nature, since the chaining value is duplicated and processed in two branches. Both branches, hereby employ a slightly different round function and also the message expansion follows a different pattern. At the end of the compression function, both branches are merged again to form the 160-bit internal state or final hash value. This increased complexity seems to complicate the analysis, and in contrast to MD4 [Dob96, WLF<sup>+</sup>05], MD5 [WY05], SHA-0 [WYY05b] and SHA-1 [WYY05a, SBK<sup>+</sup>17], collision attacks on RIPEMD-160 do not reach the full number of rounds.

Sometimes, due to the difficulty to devise a collision attack on the hash function itself, cryptanalysts may turn to analyzing its underlying compression function. Therefore, the semi-free-start collision resistance and free-start collision resistance of the underlying compression function will be investigated. A semi-free-start collision is generated with two distinct messages and the same initial value, while a free-start collision is generated with two distinct messages and two distinct initial values. In the rest of this paper, we denote semi-free-start collision and free-start collision by SFS collision and FS collision respectively. The generic time complexity of the SFS collision attack and FS collision attack are both  $2^{l/2}$  if the hash value is a  $l$ -bit value.

At Eurocrypt 2013, Landelle and Peyrin made a breakthrough [LP13] in the cryptanalysis of RIPEMD-128, whose structure is the same as that of RIPEMD-160. Specifically, they proposed a state-of-the-art technique to allow them to mount an SFS collision attack on full RIPEMD-128. Such a technique was quickly applied to analyze the compression function of RIPEMD-160 at Asiacrypt 2013. Consequently, significantly improved results for reduced RIPEMD-160 were obtained then, which were an SFS collision attack on 42 steps of RIPEMD-160 starting from an intermediate step and an SFS collision attack on 36 steps of RIPEMD-160 starting from the first step [MPS<sup>+</sup>13]. As follow-up works, an SFS collision attack on 48 steps of RIPEMD-160 starting from an intermediate step was achieved at ToSC 2017 [WSL17] and an improved SFS collision attack on 36 steps of RIPEMD-160 starting from the first step was achieved at Asiacrypt 2017 [LMW17].

As for the collision (not SFS collision) attack on reduced RIPEMD-160, the first attempt was made at Asiacrypt 2017 with rather high time complexity  $2^{70}$  [LMW17]. This attack follows the idea of using a differential characteristic that is sparse on the left branch and dense on the right branch, where message modification is used to fulfill as many conditions as possible on the dense right branch. Recently, at Crypto 2019 [LDM<sup>+</sup>19], a different strategy to find collisions was proposed, where the dense part is placed on the left branch while the sparse part is placed on the right branch. As a result, they provided the first colliding message pairs for 30 and 31 steps of RIPEMD-160 and a theoretical collision attack for up to 34 steps. A summary of the cryptanalytic results<sup>1</sup> for RIPEMD-160 is given in Table 1.

---

<sup>1</sup>The sample code to verify the SFS collisions for 36 and 37 steps of RIPEMD-160 is available at [https://github.com/Crypt-CNS/SFSCollision\\_SampleCode.git](https://github.com/Crypt-CNS/SFSCollision_SampleCode.git)

It should be noted that the two SFS collision attacks for reduced RIPEMD-160 starting from the first step share the same differential characteristic [MPS<sup>+</sup>13, LMW17]. Moreover, the underlying idea of the two SFS collision attacks is almost the same. Specifically, the dense parts with many differential conditions on both branches are firstly fixed. Then, the remaining free message words are utilized to achieve efficient merging at the initial value, which is following the idea from [LP13].

Up until now, there has not been a practical SFS collision example for the first 36 steps of RIPEMD-160. Moreover, the SFS collision attack on more steps of RIPEMD-160 starting from the first step was out of reach as well. Thus, we are motivated to further investigate the SFS collision resistance of reduced RIPEMD-160. To do so, we place the dense differential characteristic on the left branch and the sparse differential characteristic on the right branch in order to make the new SFS collision attack framework work efficiently, which follows a similar spirit as in [LDM<sup>+</sup>19]. The contribution of this paper is summarized below.

## 1.1 Our Contributions

With a new freedom degree utilization strategy, we develop an SFS collision attack framework for reduced RIPEMD-160. Different from previous SFS collision attack frameworks [MPS<sup>+</sup>13, LMW17] for RIPEMD-160 which require a costly degrees of freedom consumption to achieve efficient merging at the initial value, no merging phase is needed under the new attack framework. With such a new framework, we were able to extend the SFS collision attacks on reduced RIPEMD-160 to more steps. In addition, there are negligible memory requirements. Most importantly, combined with the use of automated techniques [MNS11, MNS13, EMS14] to solve the nonlinear differential characteristic for RIPEMD-160, improved SFS collision attacks for reduced RIPEMD-160 are obtained, as specified below.

- The SFS collision attack on 36 and 37 steps of RIPEMD-160 are achieved with time complexity  $2^{41}$  and  $2^{49}$  respectively. In addition, we also provide the corresponding colliding message pairs.
- The SFS collision attack on 38 steps is achieved with time complexity  $2^{52}$ .
- The SFS collision attack on 40 steps is achieved with time complexity  $2^{74.6}$ .

## 1.2 Organization

This paper is organized as follows. The notation, and description of RIPEMD-160 is given in Section 2. Then, we describe our SFS collision attack framework for reduced RIPEMD-160 in Section 3. Next, we discuss how to get a desirable differential characteristic in Section 4. Section 5 presents the application of our SFS collision attack framework to the discovered differential characteristics. Finally, the paper is concluded in Section 6.

## 2 Preliminaries

In this section, we will introduce the notations used in this paper and the specification of RIPEMD-160.

### 2.1 Notation

1.  $\gg$ ,  $\ll$ ,  $\ggg$ ,  $\oplus$ ,  $\vee$ ,  $\wedge$  and  $\neg$  represent the logic operations *shift right*, *rotate left*, *rotate right*, *exclusive or*, *or*, *and*, *negate*, respectively.

**Table 1:** Summary of preimage and collision attacks on reduced RIPEMD-160

Target	Attack Type	Steps	Time	Memory	Ref.
comp. function	preimage	31	$2^{148}$	$2^{17}$	[OSS12]
hash function	preimage	31	$2^{155}$	$2^{17}$	[OSS12]
comp. function	SFS collision	36/80 <sup>a</sup>	low	negligible	[MNSS12]
		42/80 <sup>a</sup>	$2^{75.5}$	$2^{64}$	[MPS+13]
		48/80 <sup>a</sup>	$2^{76.4}$	$2^{64}$	[WSL17]
		36/80	$2^{70.4}$	$2^{64}$	[MPS+13]
		36/80	$2^{55.1}$	$2^{32}$	[LMW17]
		36/80	$2^{41}$	negligible	Section 5.1
		37/80	$2^{49}$	negligible	Section 5.2
		38/80	$2^{52}$	negligible	Section 5.2
hash function	collision	40/80	$2^{74.6}$	negligible	Section 5.2
		30/80	$2^{70}$	negligible	[LMW17]
		30/80	$2^{35.9}$	$2^{32}$	[LDM+19]
		31/80	$2^{41.5}$	$2^{32}$	[LDM+19]
		33/80	$2^{67.1}$	$2^{32}$	[LDM+19]
		34/80	$2^{74.3}$	$2^{32}$	[LDM+19]

<sup>a</sup> An attack starting at an intermediate step.

2.  $\boxplus$  and  $\boxminus$  represent addition and subtraction modulo  $2^{32}$ .
3.  $M = (m_0, m_1, \dots, m_{15})$  and  $M' = (m'_0, m'_1, \dots, m'_{15})$  represent two 512-bit message blocks split into 32-bit words  $m_i$  and  $m'_i$ .
4.  $K_j^l$  and  $K_j^r$  represent the constant used for left ( $l$ ) and right ( $r$ ) branch at round  $j$ .
5.  $\Phi_j^l$  and  $\Phi_j^r$  represent the 32-bit Boolean function for the left ( $l$ ) and right ( $r$ ) branch at round  $j$ .
6.  $s_i^l$  and  $s_i^r$  represent the rotation constant used at the left ( $l$ ) and right ( $r$ ) branch during step  $i$ .
7.  $\pi_1(i)$  and  $\pi_2(i)$  represent the index of the message word used at the left ( $l$ ) and right ( $r$ ) branch during step  $i$ .
8.  $X_i, Y_i$  represent the 32-bit internal state of the left ( $l$ ) and right ( $r$ ) branch updated during step  $i$ .
9.  $X_{i,k}$  and  $Y_{i,k}$  represent the  $(k+1)$ -th bit of  $X_i$  and  $Y_i$ , where the least significant bit is the 1<sup>st</sup> bit and the most significant bit is the 32<sup>nd</sup> bit. For example,  $X_{i,0}$  represents the least significant bit of  $X_i$ .
10.  $\text{MIN}(a, b)$  represents the minimal value of  $a$  and  $b$ .  $\text{MIN}(a, b) = a$  if  $a \leq b$  and  $\text{MIN}(a, b) = b$  if  $a > b$ .

We also adopt the concept of generalized conditions of De Cannière and Rechberger [DR06] presented in Table 2.

**Table 2:** Generalized conditions [DR06]

$(x, x^*)$	(0,0)	(1,0)	(0,1)	(1,1)	$(x, x^*)$	(0,0)	(1,0)	(0,1)	(1,1)
?	✓	✓	✓	✓	3	✓	✓	—	—
—	✓	—	—	✓	5	✓	—	✓	—
x	—	✓	✓	—	7	✓	✓	✓	—
0	✓	—	—	—	A	—	✓	—	✓
u	—	✓	—	—	B	✓	✓	—	✓
n	—	—	✓	—	C	—	—	✓	✓
1	—	—	—	✓	D	✓	—	✓	✓
#	—	—	—	—	E	—	✓	✓	✓

- $x$  represents one bit of the first message and  $x^*$  represents the same bit of the second message.

## 2.2 Description of RIPEMD-160

RIPEMD-160 is a 160-bit hash function based on the Merkle-Damgård construction. So it is iterating a compression function  $H$  that takes as input a 512-bit message block  $M_i$  and a 160-bit chaining variable  $CV_i$ . We refer to [DBP96] for a detailed description of the RIPEMD-160 hash function and focus on the compression function next. The RIPEMD-160 compression function consists of two different parallel branches, which we call left and right branch, indicated by the use of  $X_i$  and  $Y_i$ , respectively. The compression function is segregated into 5 rounds of 16 steps each in both branches, leading to a total of 80 steps per branch.

### 2.2.1 Initialization

The compression function starts with an initialization, where the 160-bit chaining variable  $CV_i$  at the input is divided into five 32-bit words  $h_j$  ( $j = 0, 1, 2, 3, 4$ ). Those five words  $h_j$  are used to initialize the state of the two branches:

$$\begin{aligned} X_{-4} &= h_0^{\ggg 10}, & X_{-3} &= h_4^{\ggg 10}, & X_{-2} &= h_3^{\ggg 10}, & X_{-1} &= h_2, & X_0 &= h_1. \\ Y_{-4} &= h_0^{\ggg 10}, & Y_{-3} &= h_4^{\ggg 10}, & Y_{-2} &= h_3^{\ggg 10}, & Y_{-1} &= h_2, & Y_0 &= h_1. \end{aligned}$$

The initial value ( $CV_0$ ) corresponds to:

$$\begin{aligned} X_{-4} &= Y_{-4} = \text{0xc059d148}, & X_{-3} &= Y_{-3} = \text{0x7c30f4b8}, & X_{-2} &= Y_{-2} = \text{0x1d840c95}, \\ X_{-1} &= Y_{-1} = \text{0x98badcfe}, & X_0 &= Y_0 = \text{0xefcdab89}. \end{aligned}$$

### 2.2.2 Message Expansion

Each 512-bit input message block is divided into 16 32-bit message words  $m_i$ . The words  $m_i$  will be used for a single step in a permuted order  $\pi_1$  and  $\pi_2$  for left branch and right branch, respectively.

### 2.2.3 Step Function

At step  $i$  of round  $j$ , the internal state is updated in the following way.

$$\begin{aligned} LQ_i &= X_{i-5}^{\lll 10} \boxplus \Phi_j^l(X_{i-1}, X_{i-2}, X_{i-3}^{\lll 10}) \boxplus m_{\pi_1(i)} \boxplus K_j^l, \\ X_i &= X_{i-4}^{\lll 10} \boxplus (LQ_i)^{\lll s_i^l}, \\ RQ_i &= Y_{i-5}^{\lll 10} \boxplus \Phi_j^r(Y_{i-1}, Y_{i-2}, Y_{i-3}^{\lll 10}) \boxplus m_{\pi_2(i)} \boxplus K_j^r, \\ Y_i &= Y_{i-4}^{\lll 10} \boxplus (RQ_i)^{\lll s_i^r}, \end{aligned}$$

where  $i = (1, 2, 3, \dots, 80)$  and  $j = (0, 1, 2, 3, 4)$ . The details of the Boolean functions and round constants for RIPEMD-160 are given in Table 3.

**Table 3:** Boolean Functions and Round Constants in RIPEMD-160

Round $j$	$\phi_j^l$	$\phi_j^r$	$K_j^l$	$K_j^r$	Function	Expression
0	<i>XOR</i>	<i>ONX</i>	0x00000000	0x50a28be6	<i>XOR</i> ( $x, y, z$ )	$x \oplus y \oplus z$
1	<i>IFX</i>	<i>IFZ</i>	0x5a827999	0x5c4dd124	<i>IFX</i> ( $x, y, z$ )	$(x \wedge y) \oplus (\neg x \wedge z)$
2	<i>ONZ</i>	<i>ONZ</i>	0x6ed9eba1	0x6d703ef3	<i>IFZ</i> ( $x, y, z$ )	$(x \wedge z) \oplus (y \wedge \neg z)$
3	<i>IFZ</i>	<i>IFX</i>	0x8f1bbcdc	0x7a6d76e9	<i>ONX</i> ( $x, y, z$ )	$x \oplus (y \vee \neg z)$
4	<i>ONX</i>	<i>XOR</i>	0xa953fd4e	0x00000000	<i>ONZ</i> ( $x, y, z$ )	$(x \vee \neg y) \oplus z$

### 2.2.4 Finalization

The finalization is performed after all 80 steps have been executed in both branches. The five 32-bit words  $h'_j$  ( $j = 0, 1, 2, 3, 4$ ) composing the output chaining variable are computed in the following way involving also the chaining value at the input of the compression function  $h_j$  ( $j = 0, 1, 2, 3, 4$ ):

$$\begin{aligned} h'_0 &= h_1 \boxplus X_{79} \boxplus Y_{78}^{\lll 10}, \\ h'_1 &= h_2 \boxplus X_{78}^{\lll 10} \boxplus Y_{77}^{\lll 10}, \\ h'_2 &= h_3 \boxplus X_{77}^{\lll 10} \boxplus Y_{76}^{\lll 10}, \\ h'_3 &= h_4 \boxplus X_{76}^{\lll 10} \boxplus Y_{80}, \\ h'_4 &= h_0 \boxplus X_{80} \boxplus Y_{79}. \end{aligned}$$

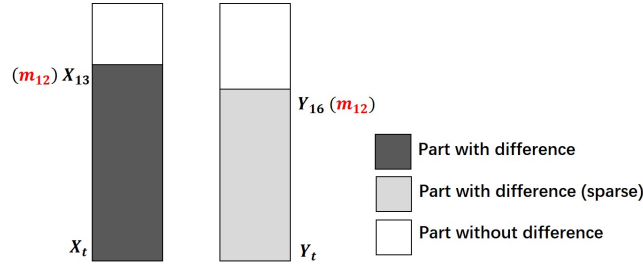
## 3 SFS Collision Attack Framework

In this section, we will present the details of the new SFS collision attack framework. For this framework, the message difference is inserted only at  $m_{12}$ , which is first used to update  $X_{13}$  and  $Y_{16}$ . For such a way to choose the message difference, the corresponding high-level presentation of the differential characteristic is depicted in Figure 1.

Since  $X_{13}$  is the first internal state with difference on the left branch and the boolean function in the first round on this branch is exclusive or (XOR), we can first confirm Observation 1 when constructing a differential characteristic.

**Observation 1.** There will be bit conditions on  $X_{12} \oplus X_{11}^{\lll 10}$ ,  $X_{13} \oplus X_{12}^{\lll 10}$  and  $X_{14} \oplus X_{12}^{\lll 10}$ . In other words, if  $X_{13}$  and  $X_{14}$  are fixed, some bits of  $X_{12}$  have to take fixed values in order to keep the conditions hold. To make the total number of the bit conditions on  $X_{12}$  small, the total number of active bits in  $X_{13}$  and  $X_{14}$  should be as small as possible.

Moreover, considering the specifics of the message expansion of RIPEMD-160, one more observation can be obtained, which will play an important role in our SFS attack framework. Observation 2 is specified below.



**Figure 1:** Attack on  $t$  steps of RIPEMD-160 by inserting difference at  $m_{12}$

$X_{13}$	$X_{14}$	$X_{15}$	$X_{16}$	$X_{17}$
$m_{12}$	$m_{13}$	$m_{14}$	$m_{15}$	$m_7$

$X_{18}$	$X_{19}$	$X_{20}$	$X_{21}$	$X_{22}$	$X_{23}$	$X_{24}$	$X_{25}$	$X_{26}$	$X_{27}$	$X_{28}$	$X_{29}$	$X_{30}$	$X_{31}$	$X_{32}$
$m_4$	$m_{13}$	$m_1$	$m_{10}$	$m_6$	$m_{15}$	$m_3$	$m_{12}$	$m_0$	$m_9$	$m_5$	$m_2$	$m_{14}$	$m_{11}$	$m_8$

$X_{33}$	$X_{34}$	$X_{35}$	$X_{36}$	$X_{37}$	$X_{38}$	$X_{39}$	$X_{40}$
$m_3$	$m_{10}$	$m_{14}$	$m_4$	$m_9$	$m_{15}$	$m_8$	$m_1$

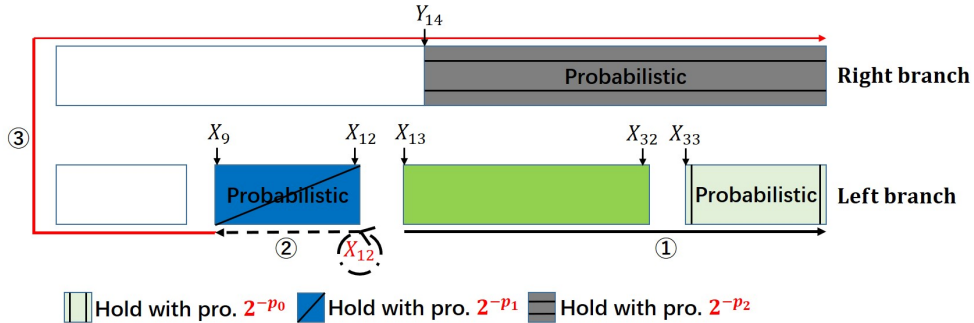
**Figure 2:** Partial information of the message expansion of RIPEMD-160

**Observation 2.** For the left branch,  $X_{17}$  is updated with  $m_7$  in the second round. Besides,  $m_7$  is used to update  $X_{42}$  in the third round.

For a better understanding of this paper, we also present partial information of the message expansion, as illustrated in Figure 2. To make our SFS work efficiently, similar to the collision attack presented at Crypto 2019 [LDM<sup>+</sup>19], we place the dense differential characteristic on the left branch and the sparse differential characteristic on the right branch.

### 3.1 Specification of the SFS collision attack framework

Based on the above strategy to construct a differential characteristic as well as the the observation of the message expansion of RIPEMD-160, an efficient SFS collision attack framework can be discovered, as illustrated in Figure 3. Suppose our aim is to mount an SFS collision attack on  $t$  steps of RIPEMD-160. On the whole, the attack procedure can be divided into 3 steps as follows.



**Figure 3:** SFS collision attack framework for RIPEMD-160

Step 1: **Finding a starting point.** Find a solution (starting point) for  $X_i$  ( $13 \leq i \leq t$ ).



With single-step message modification, randomly choose values for  $X_i$  ( $13 \leq i \leq 32$ ) while keeping the conditions on them satisfied. Based on [Observation 2](#), all message words except  $m_7$  will be fixed. The remaining work is to ensure that the conditions on  $X_i$  ( $33 \leq i \leq t$ ) hold. Generally, the conditions on this part can be partially satisfied with dedicated multi-step message modification. However, it will require some manual work. As will be shown, finding a starting point is not the bottleneck of our attack framework. Therefore, we remove the dedicated hand-tuned multi-step message modification and use a non-optimized method to satisfy the conditions on  $X_i$  ( $33 \leq i \leq t$ ) for simplicity.

**Step 2: Filtering invalid  $X_{12}$ .** Suppose there are  $n$  bit conditions on  $X_{12}$ . Then, for a fixed starting point,  $n$  bits of  $X_{12}$  will be fixed, thus leaving  $2^{32-n}$  possible values for  $X_{12}$  in total. For each possible value of  $X_{12}$ , we can compute  $m_7$  as follows:

$$m_7 = (X_{17} \boxminus X_{13}^{\lll 10}) \ggg^7 \boxplus \text{XOR}(X_{16}, X_{15}, X_{14}^{\lll 10}) \boxminus X_{12}^{\lll 10} \boxplus K_0^l.$$

Consequently, for each possible value of  $X_{12}$ , all message words will become fixed. Then, we compute backward until  $X_9$  and check the bit conditions on  $X_{12} \oplus X_{11}^{\lll 10}$  as well as the conditions on

$$LQ_i = (X_i \boxminus X_{i-4}^{\lll 10}) \ggg^{s_i^t} \quad (13 \leq i \leq 16),$$

which are used to ensure the correct propagation of the modular difference of  $X_i$  ( $13 \leq i \leq 16$ ). If these conditions hold, move to Step 3. Otherwise, choose another possible value for  $X_{12}$  and repeat. If all  $2^{32-n}$  possible values are used up, start generating a new starting point and repeat Step 2.

**Step 3: Verifying the right branch.** Until this phase, all message words are fixed. Then, for the left branch, we can compute backward to obtain the initial value. At last, we compute forward to compute the internal states on the right branch. If the conditions on the right branch do not hold, return to Step 2. Otherwise, an SFS collision is found.

## 3.2 Generating a starting point

Note that when all possible values for  $X_{12}$  are used up, we have to generate another starting point, i.e. another solution for  $X_i$  ( $13 \leq i \leq t$ ). Actually, after one starting point is obtained, a new starting point can be derived from it in negligible time, thus explaining why Step 1 is not the bottleneck of our attack framework.

In the following, we will expand on how to derive a new starting point from an existing one. For a better understanding of the next parts, we strongly suggest the readers can refer to the message expansion of RIPEMD-160 in [Figure 2](#) since the next parts strongly rely on it.

There are two strategies to derive a new starting point for two different cases. Suppose the conditions on  $X_{13}$  hold with probability  $2^{-p_3}$  and the conditions on  $X_i$  ( $36 \leq i \leq t$ ) hold with probability  $2^{-p_4}$ . The two strategies are as follows.

### 3.2.1 Strategy 1

This strategy is suitable for the case when  $p_4 \leq p_3$ . The procedure to generate a new starting point can be described below.

**Strategy 1.** Randomly choose a value for  $X_i$  ( $13 \leq i \leq 15$ ) while keeping the conditions on them satisfied. Then, modify  $m_4$ ,  $m_{13}$  and  $m_1$  as follows to keep  $X_{18}$ ,  $X_{19}$

and  $X_{20}$  the same.

$$\begin{aligned} m_4 &= (X_{18} \boxplus X_{14}^{\lll 10}) \ggg^{s_{18}^l} \boxminus IFX(X_{17}, X_{16}, X_{15}^{\lll 10}) \boxminus X_{13}^{\lll 10} \boxminus K_1^l, \\ m_{13} &= (X_{19} \boxplus X_{15}^{\lll 10}) \ggg^{s_{19}^l} \boxminus IFX(X_{18}, X_{17}, X_{16}^{\lll 10}) \boxminus X_{14}^{\lll 10} \boxminus K_1^l, \\ m_1 &= (X_{20} \boxplus X_{16}^{\lll 10}) \ggg^{s_{20}^l} \boxminus IFX(X_{19}, X_{18}, X_{17}^{\lll 10}) \boxminus X_{15}^{\lll 10} \boxminus K_1^l. \end{aligned}$$

In this way,  $X_i$  ( $16 \leq i \leq 35$ ) will stay the same. However, since  $X_{36}$  is updated with  $m_4$ , we have to recompute new values for  $X_i$  ( $36 \leq i \leq t$ ) and verify whether the conditions on them can still hold. If they do not hold, start choosing another value for  $X_i$  ( $13 \leq i \leq 15$ ) while keeping the conditions on them satisfied and repeat the above procedure until the conditions on  $X_i$  ( $36 \leq i \leq t$ ) hold. Consequently, the time to generate a new starting point is about  $2^{p_4}$  computations.

### 3.2.2 Strategy 2

This strategy is suitable for the case when  $p_3 < p_4$ . The procedure to generate a new starting point can be described below.

Strategy 2. Randomly choose a value for  $X_i$  ( $14 \leq i \leq 15$ ) while keeping the conditions on them satisfied. Then, we compute  $X_{13}$  by using  $X_i$  ( $14 \leq i \leq 18$ ) and  $m_4$  as follows.

$$X_{13} = ((X_{18} \boxplus X_{14}^{\lll 10}) \ggg^{s_{18}^l} \boxminus IFX(X_{17}, X_{16}, X_{15}^{\lll 10}) \boxminus m_4 \boxminus K_1^l) \ggg^{10}.$$

Next, we verify the conditions on  $X_{13}$  and  $LQ_{17} = (X_{17} \boxplus X_{13}^{\lll 10}) \ggg^{s_{17}^l}$ . If they do not hold, start randomly choosing another valid value for  $X_i$  ( $14 \leq i \leq 15$ ) and repeat until the conditions on  $X_{13}$  and  $LQ_{17} = (X_{17} \boxplus X_{13}^{\lll 10}) \ggg^{s_{17}^l}$  hold. If they hold, modify  $m_{13}$  and  $m_1$  as follows to keep  $X_{19}$  and  $X_{20}$  the same.

$$\begin{aligned} m_{13} &= (X_{19} \boxplus X_{15}^{\lll 10}) \ggg^{s_{19}^l} \boxminus IFX(X_{18}, X_{17}, X_{16}^{\lll 10}) \boxminus X_{14}^{\lll 10} \boxminus K_1^l, \\ m_1 &= (X_{20} \boxplus X_{16}^{\lll 10}) \ggg^{s_{20}^l} \boxminus IFX(X_{19}, X_{18}, X_{17}^{\lll 10}) \boxminus X_{15}^{\lll 10} \boxminus K_1^l. \end{aligned}$$

In this way,  $X_i$  ( $16 \leq i \leq 39$ ) will stay the same. Thus, for the attack on fewer than 40 steps, the time to generate a new starting point is about  $2^{p_3}$  computations.

For the attack on 40 steps of RIPEMD-160, since  $X_{40}$  is updated with  $m_1$ , we have to recompute a new value for  $X_{40}$  and check its conditions. If they do not hold, start choosing another new valid value for  $X_i$  ( $14 \leq i \leq 15$ ) and repeat until a valid starting point is found. For the attack on 40 steps of RIPEMD-160, we only need to check whether  $LQ_{40}$  can satisfy its corresponding equation. As will be shown in the 40-step differential characteristic, such a probability is close to 1 and therefore the time to generate a starting point is also about  $2^{p_3}$  computations.

As shown in **Strategy 2**, we fix the value for  $m_4$  to keep the internal states  $X_i$  ( $36 \leq i \leq t \leq 39$ ) the same. In this case, the degrees of freedom to generate a new starting point are provided by the free bits of  $X_{14}$  and  $X_{15}$ . When the right branch holds with a relatively low probability, i.e. like the 40-step differential characteristic, a sufficient number of starting points are needed. Therefore, we can also use the degrees of freedom of  $m_4$ . Specifically, we can first store all valid values for  $m_4$  which can make the conditions on  $X_i$  ( $36 \leq i \leq t \leq 39$ ) hold in an array. This can be achieved by exhausting all valid values for  $X_{36}$  and compute  $X_i$  ( $37 \leq i \leq t \leq 39$ ) as well as check the conditions on them for a fixed

solution for  $X_i$  ( $16 \leq i \leq 35$ ). Then, instead of only randomly choosing a valid value for  $X_{14}$  and  $X_{15}$ , we can also randomly choose a valid value for  $m_4$  from this array. In a word, to generate a new starting point, the degrees of freedom can be provided by  $X_{14}$ ,  $X_{15}$  and  $m_4$ . Such a slightly modified **Strategy 2** will require some memory to store all valid  $m_4$ .

### 3.2.3 Generating the initial starting point

Indeed, the above two strategies to generate a new starting point imply that only one solution  $X_i$  ( $16 \leq i \leq 35$ ) is needed. For such a solution,  $m_7$ ,  $m_4$ ,  $m_{13}$  and  $m_1$  are not fixed. If  $p_4 \leq p_3$ , we directly use **Strategy 1** to generate a starting point. If  $p_3 < p_4$ , we first exhaust all valid values for  $X_{36}$  and compute the corresponding  $m_4$  ( $m_4$  is used to update  $X_{36}$ ) as well as  $X_i$  ( $37 \leq i \leq t \leq 39$ ). Record the values for  $m_4$  which can make the conditions on  $X_i$  ( $37 \leq i \leq t \leq 39$ ) hold. Then, **Strategy 2** can be applied to find a starting point.

Obviously, finding a solution for  $X_i$  ( $16 \leq i \leq 35$ ) cannot be the bottleneck since only three internal states  $X_{33}$ ,  $X_{34}$  and  $X_{35}$  cannot hold trivially. In our implementation, when the number of conditions on  $X_{33}$ ,  $X_{34}$  and  $X_{35}$  is small, we simply make them hold probabilistically, i.e. we repeat finding a solution for  $X_i$  ( $16 \leq i \leq 32$ ) with single-step message modification until the conditions on  $X_i$  ( $33 \leq i \leq 35$ ) hold. When the total number of conditions are not too small, we will again use a simple start-from-the-middle method to find a solution for  $X_i$  ( $16 \leq i \leq 35$ ).

## 3.3 Complexity Evaluation

Although no differential characteristic is presented now, we can give a rough estimation of the time complexity of the SFS collision attack on  $t$  ( $36 \leq t \leq 40$ ) steps of RIPEMD-160 before considering a specific differential characteristic. This is owing to the efficiency of our SFS collision attack framework.

Specifically, when a starting point is found, we can exhaust all valid values for  $X_{12}$  and initially filter them by checking the conditions on  $X_{11}$  and  $LQ_i$  ( $13 \leq i \leq 16$ ). When all possible values for  $X_{12}$  are used up for a starting point, we can efficiently generate a new starting point in time  $\text{MIN}(2^{p_3}, 2^{p_4})$ , where  $p_3$  and  $p_4$  are defined in Section 3.2.

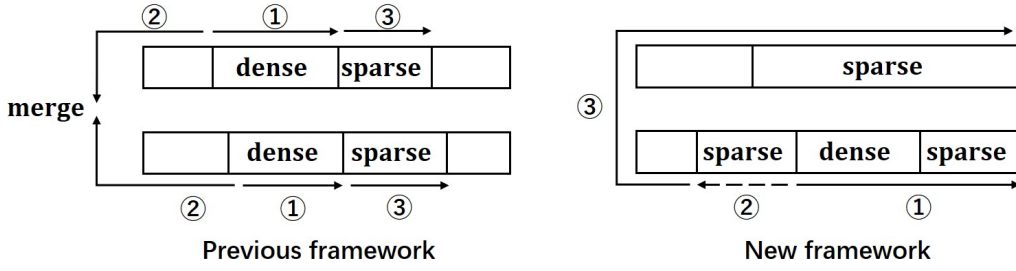
As shown in Figure 3, suppose the conditions on  $X_{12} \oplus X_{11}^{\lll 10}$  and  $LQ_i$  ( $13 \leq i \leq 16$ ) hold with probability  $2^{-p_1}$ , and the fully probabilistic right branch holds with probability  $2^{-p_2}$ . Moreover, we also suppose there are  $n$  bit conditions on  $X_{12}$ . Then, for each starting point, we will verify the right branch with different  $m_7$  for about  $2^{32-n-p_1}$  times. The time complexity of this phase (exhausting all possible values of  $X_{12}$  and checking the right branch) can be estimated as:

$$T_0 = \frac{4}{80} \cdot 2^{32-n} + \frac{13+t}{80} \cdot 2^{32-n-p_1}.$$

As will be shown,  $p_1$  will be very small in our discovered differential characteristic, i.e.  $p_1 \approx 2$ . Therefore, we roughly estimate the time complexity of this phase as

$$T_0 = \frac{4}{80} \cdot 2^{32-n} + \frac{13+t}{80} \cdot 2^{32-n-p_1} \approx 2^{32-n-p_1}.$$

However, the right branch holds with probability  $2^{-p_2}$ . Thus, it is expected to verify the right branch for about  $2^{p_2}$  times under our SFS collision attack framework in order to find an SFS collision. Since each starting point can only provide about  $2^{32-n-p_1}$  attempts, we need to have about  $2^{p_2-(32-n-p_1)} = 2^{p_1+p_2+n-32}$  starting points. Suppose only one solution for  $X_i$  ( $16 \leq i \leq 35$ ) is enough, which means  $m_4$ ,  $X_{14}$  and  $X_{15}$  can provide sufficient degrees of freedom to generate  $2^{p_1+p_2+n-32}$  starting points. Then, apart from



**Figure 4:** Comparison between our framework and previous frameworks [LMW17, MPS<sup>+</sup>13]

the initial starting point, each starting point can be generated with time  $\text{MIN}(2^{p_3}, 2^{p_4})$ . Thus, the total time complexity of our SFS collision attack on  $t$  steps of RIPEMD-160 is

$$\begin{aligned} T &= \text{MIN}(2^{p_3}, 2^{p_4}) \times 2^{p_1+p_2+n-32} + 2^{p_1+p_2+n-32} \times 2^{32-n-p_1} \\ &= \text{MIN}(2^{p_3}, 2^{p_4}) \times 2^{p_1+p_2+n-32} + 2^{p_2}. \end{aligned}$$

As will be shown in the differential characteristics,  $p_1 \approx 2$ ,  $\text{MIN}(2^{p_3}, 2^{p_4}) \leq 2^5$  and  $n \leq 3$ . Thus, we have

$$T = \text{MIN}(2^{p_3}, 2^{p_4}) \times 2^{p_1+p_2+n-32} + 2^{p_2} \leq 2^{5+2+p_2+3-32} + 2^{p_2} \approx 2^{p_2}. \quad (1)$$

In other words, under our SFS collision attack framework, the time complexity to find an SFS collision for  $t$  steps of RIPEMD-160 is fully dominated by the probabilistic right branch.

### 3.4 Advantage

As can be observed, our new SFS collision attack framework is different from previous ones presented at Asiacrypt 2013 [MPS<sup>+</sup>13] and Asiacrypt 2017 [LMW17], as depicted in Figure 4. Compared with the SFS collision attack frameworks for 36 steps of RIPEMD-160 [LMW17, MPS<sup>+</sup>13], our new SFS collision attack framework can bring the following three advantages.

- The memory complexity is negligible for our new framework, while it is  $2^{32}$  in previous work [LMW17, MPS<sup>+</sup>13] after an optimization based on [LMW17]. It should be noted that  $2^{32}$  memory is practical in a way. However, it will be still somewhat expensive for a parallel search.
- Our new framework allows us to mount SFS collision attack on more steps of RIPEMD-160 when inserting a message difference at the message word  $m_{12}$ . However, it seems difficult to attack more steps when adopting the framework [LMW17, MPS<sup>+</sup>13] by inserting difference at  $m_7$ . As will be shown, the new framework can be used to mount SFS collision attack on 36/37/38/40 steps of RIPEMD-160.
- The framework can provide significantly improved results for the SFS collision attack on reduced RIPEMD-160.

#### 3.4.1 Remark

With the start-from-the-middle structure, while it is hard to turn an SFS collision attack into a collision attack due to the match with the predefined initial value, it is easy to turn a collision attack into an SFS collision attack.

For the dense-left-and-sparse-right (DLSR) collision attack framework in [LDM<sup>+</sup>19], an intuitive idea to convert it into an SFS collision attack framework is to remove the connecting phase. Specifically, the starting point is a solution for  $X_i$  ( $11 \leq i \leq 23$ ) in the DLSR framework [LDM<sup>+</sup>19]. Then, the attacker can always first keep the conditions on  $X_i$  ( $24 \leq i \leq 32$ ) satisfied with single-step message modification since their corresponding message words are used for the first time. Finally, for each valid value for  $X_i$  ( $24 \leq i \leq 32$ ), all message words become fixed and therefore the attacker can compute the remaining internal states on both branches and verify their conditions.

For the 34-step differential characteristic in [LDM<sup>+</sup>19], the probability that the conditions on the remaining internal states hold is not too low, i.e. greater than  $2^{-40}$ , one can repeat choosing valid values for  $X_i$  ( $24 \leq i \leq 32$ ) with single-step message modification and verify these conditions. Once they are satisfied, an SFS collision is found. Obviously, the time complexity to find an SFS collision for 34 steps of RIPEMD-160 will not exceed  $2^{40}$  and is practical. However, if it is directly applied to a longer differential characteristic, one has to deal with the conditions in the third round on the left branch.

Compared with the above naive SFS collision attack framework derived from the DLSR collision attack framework in [LDM<sup>+</sup>19], our new framework adopts a better freedom degree utilization strategy, thus performing better for a longer differential characteristic. In our new framework, the starting point is a solution for  $X_i$  ( $13 \leq i \leq t$ ) while it is a solution for  $X_i$  ( $11 \leq i \leq 23$ ) in [LDM<sup>+</sup>19]. Especially, we also show that the total time complexity is fully dominated by the right branch under our new SFS collision attack framework if a suitable differential characteristic is obtained. Determining such an almost optimal freedom degree utilization strategy is obviously non-trivial.

## 4 Differential Characteristics

Our SFS collision attack procedure to find a semi-free-collision for reduced RIPEMD-160 has been explained in detail in Section 3. Thus, the next task is to find a suitable differential characteristic to make the framework work efficiently. Thanks to the use of automated techniques [MNS11, MNS13, EMS14], this task can be finished efficiently. Thus, the remaining work is to add some constraints on the differential characteristic before the search in order to find a desirable one.

As explained in Section 3, once a solution for the starting point is found, we can immediately utilize the degrees of freedom provided by  $X_{12}$ . Besides, there will be a filtering phase to filter invalid  $X_{12}$ . We expect there are sufficient valid  $X_{12}$  left after filtering. Thus, the desirable differential characteristic have the following properties.

- There should be only one active bit in  $X_{12}$  so that there is only one bit condition on  $X_{11}$ .
- The probability that  $LQ_i = (X_i \boxminus X_{i-4}^{\lll 10}) \ggg^{s_i^l}$  ( $13 \leq i \leq 16$ ) satisfy their corresponding equations should be as high as possible.
- The total number of active bits in  $X_{13}$  and  $X_{14}$  should be as small as possible so that there are a few bit conditions on  $X_{12}$ . This is to ensure  $X_{12}$  can take as many possible values as possible before filtering.

When taking the generation of new starting points into account, we can expect that the cost is as small as possible. Besides, there should be sufficient degrees of freedom provided by  $X_{14}$  and  $X_{15}$  and  $m_4$ . Thus, the desirable differential characteristic should have the following extra properties.

- The total number of active bits in  $X_{15}$ ,  $X_{16}$ ,  $X_{17}$  should be as small as possible. Besides, the probability that  $LQ_i = (X_i \boxminus X_{i-4}^{\lll 10}) \ggg^{s_i^l}$  ( $17 \leq i \leq 19$ ) satisfy their

corresponding equations should be as high as possible. In this way, it is expected that  $X_{14}$  and  $X_{15}$  can provide sufficient degrees of freedom.

- The probability that the conditions on  $X_i$  ( $36 \leq i \leq t \leq 39$ ) hold should not be too small. Then, we can also utilize the degrees of freedom provided by  $m_4$ .

In a word, the differential characteristic located at  $X_i$  ( $13 \leq i \leq 17$ ) and  $X_i$  ( $36 \leq i \leq t$ ) should be as sparse as possible. Then, we can solve the nonlinear differential characteristic located at  $X_i$  ( $18 \leq i \leq 35$ ) with the use of automated techniques [MNS11, MNS13, EMS14]. The desirable discovered differential characteristics are displayed in Table 12, Table 13, Table 14 and Table 15 in Appendix A respectively. In addition, we also provide the solution for  $X_i$  ( $16 \leq i \leq 35$ ) in Table 4.

As will be shown, the colliding message pairs for 36 and 37 steps of RIPEMD-160 have been found. For 38 steps of RIPEMD-160, the time complexity is  $2^{52}$ , which may become practical with more powerful computing resources. However, such powerful computing resources are out of our reach. For the attack on 39 steps under our framework, the right branch will hold with probability about  $2^{-59}$ , suggesting that the time complexity will be about  $2^{59}$  if a suitable differential characteristic can be found. Thus, for the attack with relatively high time complexity, we focus on more steps. In other words, we will concentrate on the theoretical SFS collision attack on 40 steps of RIPEMD-160.

## 5 Application

In this section, we present the results of the new SFS collision attack on 36/37/38/40 steps of RIPEMD-160. As our attack framework requires, we have to focus on the conditions on the following part:

1. The conditions on the right branch, which will influence the whole time complexity.
2. The conditions on  $X_{12}$ , which will influence the total number of possible values for  $X_{12}$  before filtering.
3. The conditions on  $X_{11}$  and  $LQ_i$  ( $13 \leq i \leq 16$ ), which will influence the filtering phase.
4. The conditions on  $X_{13}$  and  $LQ_{17}$ , which will influence the time to generate a new starting point.
5. The conditions on  $X_i$  ( $14 \leq i \leq 15$ ), which will influence the degrees of freedom to generate a new starting point. We stress here that we have added extra conditions on  $X_{14}$  and  $X_{15}$  to make  $LQ_i$  ( $18 \leq i \leq 19$ ) satisfy their equations. Therefore, even if  $X_{14}$  and  $X_{15}$  are changed,  $LQ_i$  ( $18 \leq i \leq 19$ ) will always satisfy their equations.
6. The conditions on  $X_i$  ( $36 \leq i \leq t$ ), which will influence the total number of valid  $m_4$ . In other words, it will also influence the degrees of freedom to generate a starting point.

Therefore, when describing the SFS collision attack in next sections, we will firstly list the above conditions.

### 5.1 Practical SFS Collision on 36 Steps of RIPEMD-160

As discussed above, we first list in Table 6 some conditions influencing the performance of the SFS collision attack, which are not presented in Table 12. As Table 6 shows, the probability that  $LQ_{36}$  satisfies its corresponding equations is close to 1 (there is no need to consider the bit conditions on  $X_{36}$  when we attack 36 steps of RIPEMD-160), while the conditions on  $X_{13}$  hold with probability  $2^{-5}$ . Therefore, we use **Strategy 1** to generate a new starting point, whose cost can be neglected.

Table 4: Solution for  $X_i$  ( $16 \leq i \leq 35$ )

36 steps		37 steps	
$m_0 = 0x6c2c8526, m_2 = 0x16188d15,$	$m_3 = 0xc6c5da57, m_5 = 0xf7a7a97a,$	$m_6 = 0xa7cbbf38, m_8 = 0xb6477677,$	$m_9 = 0x47f24a3e, m_{10} = 0xb1bdf3b5,$
$m_{11} = 0x78aaa252, m_{12} = 0x69a579f0,$	$m_{14} = 0xbb877480, m_{15} = 0x5caa647e.$	$m_0 = 0x2a3e3e5d, m_2 = 0xc5ab4a9c,$	$m_3 = 0xdc1f16ce, m_5 = 0x848cc0fe,$
$m_6 = 0xa7cbbf38, m_8 = 0xb6477677,$	$m_9 = 0x47f24a3e, m_{10} = 0xb1bdf3b5,$	$m_{11} = 0x59b4296f, m_{12} = 0x14a47a10,$	$m_{14} = 0x3b3e4837, m_{15} = 0x7fd45b3f.$
$X_{16}$ 101001111011110011u0011110n1100	$X_{16}$ 101010000010001101n1000101uu0110	$X_{17}$ n0010101101100111100110111101111	$X_{17}$ u1000100111000111001111001000100
$X_{17}$ 01000101110111001111101nu001001	$X_{18}$ 0000000010011111u0001001u111000	$X_{18}$ 1100110111011111u0100n00100001	$X_{19}$ n111110100011000110111u10000001
$X_{19}$ 11010000110001101n10001011010000	$X_{20}$ 0010n100n0110111u1uu110010100110	$X_{20}$ 11010000110001101n10001011010000	$X_{21}$ 001110u000101101000100u100011011
$X_{21}$ nnnn1nn1010111101011nu110100u10	$X_{21}$ 1101100u10u000un0100001uuuuu1000	$X_{22}$ 11010000110001101n10001011010000	$X_{22}$ 1uu11nuu11100u0u11u1uuun1001010u
$X_{22}$ 1001001nu1101u0n1001n1nuunnnu11	$X_{23}$ 110n11nn1001uu110u0n10u1u0nu001u	$X_{23}$ 11010000110001101n10001011010000	$X_{24}$ 00n0010n0nn0un0000nnn01u1u10100
$X_{23}$ nn110u11n10nu100n001nnn10u11nnu1	$X_{24}$ 01000011nn0110u1100n010n0n0nn1	$X_{24}$ 11010000110001101n10001011010000	$X_{25}$ 0nnnnn001101011110nnnnn011u0u10
$X_{24}$ 1101nu0010uun01nu1n0000nn1101u10	$X_{25}$ 1nu1111uu1un0u01n0n1nnnn1100n00	$X_{25}$ 11010000110001101n10001011010000	$X_{26}$ 01000011nn0110u1100n01uu00n01u1
$X_{25}$ 11uu1nn10n011001n0u01uuu1n101uuu	$X_{26}$ 1nu1111uu1un0u01n0n1nnnn1100n00	$X_{26}$ 11010000110001101n10001011010000	$X_{27}$ u001uuuu00u0101un010011u0001n0uu
$X_{26}$ 1101011u1un0u100u01uuuuuu0010010	$X_{27}$ 11010000110001101n10001011010000	$X_{27}$ 011u0010011n111uuun001111000111	$X_{28}$ 1nu1111uu1un0u01n0n1nnnn1100n00
$X_{27}$ 0u11u0010011n111uuun001111000111	$X_{28}$ u001uuuu00u0101un010011u0001n0uu	$X_{28}$ 11000100n11nn0n11100n010n0n0nn1	$X_{29}$ 1110000101110u10u100uuu0010uuu01
$X_{28}$ 11000100n11nn0n11100n010n0n0nn1	$X_{29}$ 1101110010101nnn0110nnn10110111	$X_{29}$ 01n00nu000u01nnu101uuu0ununu11n	$X_{30}$ 0000110110111u01n00101000010u
$X_{29}$ 01n00nu000u01nnu101uuu0ununu11n	$X_{30}$ 10n0101000000011111001001000010u	$X_{30}$ n1u01n10u010011n000110100000un1u	$X_{31}$ 101010101001011110101011nnm0110u
$X_{30}$ n1u01n10u010011n000110100000un1u	$X_{31}$ 01n0001101101000100100001000110u	$X_{31}$ 01n1000nnn0110101010111nnm0110u	$X_{32}$ 0000110110111u01n00101110111101
$X_{31}$ 01n1000nnn0110101010111nnm0110u	$X_{32}$ 01n0001101101000100100001000110u	$X_{32}$ 10n0101000000011111001001000010u	$X_{33}$ 101101100000111011111011011001
$X_{32}$ 10n0101000000011111001001000010u	$X_{33}$ 01010101010010111110101000111111	$X_{33}$ 01n0001101101000100100001000110u	$X_{34}$ 0101101010101010001010100001110
$X_{33}$ 01n0001101101000100100001000110u	$X_{34}$ 1011000010111011110111011011001	$X_{34}$ 01010101010010111110101000111111	$X_{35}$ 0000111100000101011110011000100
$X_{34}$ 01010101010010111110101000111111	$X_{35}$ 10110000101110111110111101101101	$X_{35}$ 1011000010111011111011101101101	
$X_{35}$ 10110000101110111110111101101101			
38 steps		40 steps	
$m_0 = 0xc32cb8b2, m_2 = 0xdcebf941,$	$m_3 = 0x473889d3, m_5 = 0x38875789,$	$m_6 = 0xc32cb8b2, m_8 = 0xb8dce09a,$	$m_9 = 0xc32cb8b2, m_{10} = 0x67c766e5,$
$m_6 = 0xc32cb8b2, m_8 = 0xb8dce09a,$	$m_9 = 0xc32cb8b2, m_{10} = 0x67c766e5,$	$m_{11} = 0x9c0866a4, m_{12} = 0x6dcd4ef1,$	$m_{14} = 0x74e28f11, m_{15} = 0x898b12aa.$
$m_{11} = 0x9c0866a4, m_{12} = 0x6dcd4ef1,$	$m_{14} = 0x74e28f11, m_{15} = 0x898b12aa.$	$m_0 = 0x31973617, m_2 = 0x3f5a3668,$	$m_3 = 0xfc3ffea2, m_5 = 0x97ccd10f,$
$m_6 = 0xc32cb8b2, m_8 = 0xb8dce09a,$	$m_9 = 0xc32cb8b2, m_{10} = 0x67c766e5,$	$m_6 = 0x41688e61, m_8 = 0x69a1d2a2,$	$m_9 = 0x5b2331f3, m_{10} = 0x1c7c9435,$
$m_{11} = 0x9c0866a4, m_{12} = 0x6dcd4ef1,$	$m_{14} = 0x74e28f11, m_{15} = 0x898b12aa.$	$m_{11} = 0x9d41f6eb, m_{12} = 0x1f5f0b1b,$	$m_{14} = 0x19ec64c9, m_{15} = 0x86c9080.$
$X_{16}$ 000100010101111111u1000101uu1100	$X_{16}$ 001110010011001110u111101n0u1001	$X_{17}$ u1111101010000011100001110001110	$X_{17}$ u1110110001101101100001110100001
$X_{17}$ u1111101010000011100001110001110	$X_{18}$ 0000001010101010101111000u011010	$X_{18}$ 010101111001000111100001u101000	$X_{18}$ 0000001010101010101111000u011010
$X_{18}$ 010101111001000111100001u101000	$X_{19}$ 1000011100010110u001111u10000001	$X_{19}$ 010101100110100nu111101n10101010	$X_{19}$ 1000011100010110u001111u10000001
$X_{19}$ 010101100110100nu111101n10101010	$X_{20}$ 11110011u111011n0u10010001001011	$X_{20}$ 00000001u101101uu11110011010011	$X_{20}$ 11110011u111011n0u10010001001011
$X_{20}$ 00000001u101101uu11110011010011	$X_{21}$ 010111100000100001001u0010000000	$X_{21}$ 1101110101100000110001u100011000	$X_{21}$ 010111100000100001001u0010000000
$X_{21}$ 1101110101100000110001u100011000	$X_{22}$ 101001010101100u1100010110011110	$X_{22}$ 1100n0unu10un1nn01unnnnnnnnnn10	$X_{22}$ 101001010101100u1100010110011110
$X_{22}$ 1100n0unu10un1nn01unnnnnnnnnn10	$X_{23}$ 100010u011001u100011100001010111	$X_{23}$ n0uuu1100010nuuu0uuuu1nun101nu11	$X_{23}$ 100010u011001u100011100001010111
$X_{23}$ n0uuu1100010nuuu0uuuu1nun101nu11	$X_{24}$ 10011n0u01011011110unnnn1001um11	$X_{24}$ 101111101un01u10011000001101000	$X_{24}$ 10011n0u01011011110unnnn1001um11
$X_{24}$ 101111101un01u10011000001101000	$X_{25}$ 111100010n0u0nnn010011un10011100	$X_{25}$ uu1110n001000100010nuu01u11101n0	$X_{25}$ 111100010n0u0nnn010011un10011100
$X_{25}$ uu1110n001000100010nuu01u11101n0	$X_{26}$ unu001uu001u01001u101101111u11u1	$X_{26}$ 01nu101u11u00010100u000011u10111	$X_{26}$ unu001uu001u01001u101101111u11u1
$X_{26}$ 01nu101u11u00010100u000011u10111	$X_{27}$ 011un011nnuuuuuuu10000u00u100u0	$X_{27}$ 01000uu0010u1n0100nnn11011n11101	$X_{27}$ 011un011nnuuuuuuu10000u00u100u0
$X_{27}$ 01000uu0010u1n0100nnn11011n11101	$X_{28}$ 00111nun0111u1nn0nnnnnn10n0nnn1	$X_{28}$ 0u10u000110000011100001000000101	$X_{28}$ 00111nun0111u1nn0nnnnnn10n0nnn1
$X_{28}$ 0u10u000110000011100001000000101	$X_{29}$ nn111010n0101010100010u1un10101n	$X_{29}$ 1010101011n001000001n0n1001n100	$X_{29}$ nn111010n0101010100010u1un10101n
$X_{29}$ 1010101011n001000001n0n1001n100	$X_{30}$ 0111u11010000n01101100uuu11101u0	$X_{30}$ n100010u01101100u010100110n10101	$X_{30}$ 0111u11010000n01101100uuu11101u0
$X_{30}$ n100010u01101100u010100110n10101	$X_{31}$ n001u110n10u1110nn11u0100111110n	$X_{31}$ 00uuuuuu0110nu111100010101111110	$X_{31}$ n001u110n10u1110nn11u0100111110n
$X_{31}$ 00uuuuuu0110nu111100010101111110	$X_{32}$ 1000uuu010n1n011u011110100n0nu01	$X_{32}$ u11111n100110110n01u0101nuu10000	$X_{32}$ 1000uuu010n1n011u011110100n0nu01
$X_{32}$ u11111n100110110n01u0101nuu10000	$X_{33}$ 0u1110101100u01u1111n01u000111n0	$X_{33}$ 00011111110un0100010101100001100	$X_{33}$ 0u1110101100u01u1111n01u000111n0
$X_{33}$ 00011111110un0100010101100001100	$X_{34}$ 011n1n0010u10n01000000100001011	$X_{34}$ u11n1001101110101101000100000111	$X_{34}$ 011n1n0010u10n01000000100001011
$X_{34}$ u11n1001101110101101000100000111	$X_{35}$ 0u000000100110110111111110001n1	$X_{35}$ 1001001001001111011010111010110	$X_{35}$ 0u000000100110110111111110001n1
$X_{35}$ 1001001001001111011010111010110			

Moreover, based on Table 6 and Table 12, there will be  $2^{32-3} = 2^{29}$  possible values for  $X_{12}$  for a given starting point. After filtering, about  $2^{29-1.7} = 2^{27.3}$  valid values for  $X_{12}$  are left. Since the right branch holds with probability  $2^{-41}$ , we need to generate about  $2^{41-27.3} = 2^{13.7}$  starting points. It should be noticed in Table 6 that there is a sufficient number of free bits in  $X_i$  ( $13 \leq i \leq 15$ ). Therefore, we can only use one solution for  $X_i$  ( $16 \leq i \leq 35$ ). Thus, the time complexity to mount an SFS collision attack on 36 steps of RIPEMD-160 can be evaluated with the Eqn. 1 in Section 3.3, where

$$(p_1, p_2, p_3, p_4, n) = (1.7, 41, 5, 0, 3).$$

Therefore, the time complexity to find an SFS collision for 36 steps is  $2^{41}$ .

We have implemented the attack. Specifically, we ran 25 different SFS collision search instances on 25 CPUs with different seeds simultaneously. Moreover, the solution for  $X_i$  ( $16 \leq i \leq 35$ ) for the 25 search instances is also different from each other. We obtained 4 colliding message pairs in about one day. The colliding message pair in Table 5 was obtained in less than 20 minutes.

**Table 5:** SFS collision for 36 steps of RIPEMD-160

$h_0 \sim h_4$	809825f7 d2a55861 6bd86be7 fc58a6cb 11f6a005
$M$	6c2c8526 dc3084cc 16188d15 c6c5da57 73f15b99 f7a7a97a a7cbbf38 53a4b30 b6477677 47f24a3e b1bdf3b5 78aaa252 <b>69a579f0</b> 72b32f35 bb877480 5caa647e
$M'$	6c2c8526 dc3084cc 16188d15 c6c5da57 73f15b99 f7a7a97a a7cbbf38 53a4b30 b6477677 47f24a3e b1bdf3b5 78aaa252 <b>69a5f9f0</b> 72b32f35 bb877480 5caa647e
hash value	88f79fa4 c9973719 dcf0ff7f 15cef816 a9d702a5

**Table 6:** Other conditions influencing the attack for the 36-step differential characteristic

	Conditions	Probability
$Y_{18}$	$Y_{18,31} = Y_{17,31}$	$2^{-1}$
$Y_{22}$	$Y_{22,9} = Y_{21,9}$	$2^{-1}$
$Y_{26}$	$Y_{26,20} = Y_{25,20}, Y_{26,19} = Y_{25,19}$	$2^{-2}$
$Y_{30}$	$Y_{30,0} = Y_{29,0}, Y_{30,29} = Y_{29,29}, Y_{30,30} = Y_{29,30}$	$2^{-3}$
$Y_{34}$	$Y_{34,7} \vee \neg Y_{33,7} = 1, Y_{34,10} \vee \neg Y_{33,10} = 1$	$2^{-1}$
$RQ_{16}$	$(RQ_{16} \boxplus 0x8000) \lll 6 = RQ_{16} \lll 6 \boxplus 0x200000$	Negligible
$RQ_{28}$	$(RQ_{28} \boxplus 0x8000) \lll 7 = RQ_{28} \lll 7 \boxplus 0x400000$	Negligible
$RQ_{35}$	$(RQ_{35} \boxplus 0xffffc80) \lll 15 = RQ_{35} \lll 15 \boxplus 0xfe400000$	Negligible
	Right Branch	$2^{-33-8} = 2^{-41}$
$X_{12}$	$X_{12,19} \neq X_{13,29}, X_{12,18} \neq X_{13,28}, X_{12,11} = X_{14,21}$	$2^{-3}$
$X_{11}$	$X_{11,11} = X_{12,21}$	$2^{-1}$
$LQ_{13}$	$(LQ_{13} \boxplus 0x8000) \lll 6 = LQ_{13} \lll 6 \boxplus 0x200000$	Negligible
$LQ_{14}$	$(LQ_{14} \boxplus 0x200000) \lll 7 = LQ_{14} \lll 7 \boxplus 0x10000000$	$2^{-0.1}$
$LQ_{15}$	$(LQ_{15} \boxplus 0xf0200000) \lll 9 = LQ_{15} \lll 9 \boxplus 0x3ffffe0$	$2^{-0.6}$
$LQ_{16}$	$(LQ_{16} \boxplus 0xffffe0) \lll 8 = LQ_{16} \lll 8 \boxplus 0xffffe010$	Negligible
	Filtering	$2^{-1.7}$
$X_{13}$	$X_{13,27} = X_{14,5}, X_{13,20} \neq X_{14,30}, X_{13,19} \neq X_{15,29}, X_{13,18} = X_{15,28}$	$2^{-4}$
$X_{15}$	$X_{15,13} = X_{14,3}, X_{15,4} = X_{14,26}, X_{15,21} \neq X_{16,31}$	$2^{-3}$
	The number of free bits in $X_{14}$ and $X_{15}$ : $64 - 3 - 4 = 57$	
$LQ_{36}$	$(LQ_{36} \boxplus 0x38007) \lll 7 = LQ_{36} \lll 7 \boxplus 0x1c00380$	Negligible
	The expected number of valid $m_4$ : $2^{32-0} = 2^{32}$	
	The expected number of starting points for a fixed $X_i$ ( $16 \leq i \leq 35$ ): $2^{57+32-5} = 2^{84}$	

## 5.2 SFS Collision for 37/38/40 Steps of RIPEMD-160

Similarly, for the SFS collision attack on 37/38/40 steps of RIPEMD-160 under our attack framework, we first list some conditions influencing the performance of the SFS collision attack in Table 7, Table 9, Table 10, which are not presented in Table 13, Table 14, Table 15.



### 5.2.1 Attack on 37 steps of RIPEMD-160

Based on Table 7 and Table 13, we conclude that there will be  $2^{32-2} = 2^{30}$  possible values for  $X_{12}$  for a given starting point. After filtering, about  $2^{30-2} = 2^{28}$  are left. Since the conditions on  $X_i$  ( $36 \leq i \leq 37$ ) hold with probability  $2^{-2.3}$  and the conditions on  $X_{13}$  hold with probability  $2^{-4}$ , we use **Strategy 1** to generate a new starting point, whose cost is about  $2^{2.3}$  computations. Since the right branch holds with probability  $2^{-49}$ , we expect that it will be required to generate  $2^{49-28} = 2^{21}$  starting points. As Table 7 shows,  $X_i$  ( $13 \leq i \leq 15$ ) can provide sufficient degrees of freedom to generate so many starting points for a fixed solution for  $X_i$  ( $16 \leq i \leq 35$ ). Thus, the time complexity to mount an SFS collision attack on 37 steps of RIPEMD-160 can be evaluated with the Eqn. 1, where

$$(p_1, p_2, p_3, p_4, n) = (2, 49, 4, 2.3, 2).$$

Therefore, the time complexity to find an SFS collision for 37 steps of RIPEMD-160 is  $2^{49}$ . We tried our best and have found a colliding message pair for 37 steps of RIPEMD-160, as shown in Table 8. Specifically, we started a search on 31 CPUs simultaneously and this colliding message pair was obtained on one of the 31 CPUs with the right branch checked for about  $2^{44}$  times.

**Table 7:** Other conditions influencing the attack for the 37-step differential characteristic

	Conditions	Probability
$Y_{18}$	$Y_{18,31} = Y_{17,31}$	$2^{-1}$
$Y_{22}$	$Y_{22,9} = Y_{21,9}$	$2^{-1}$
$Y_{26}$	$Y_{26,20} = Y_{25,20}, Y_{26,19} = Y_{25,19}$	$2^{-2}$
$Y_{30}$	$Y_{30,0} = Y_{29,0}, Y_{30,29} = Y_{29,29}, Y_{30,30} = Y_{29,30}$	$2^{-3}$
$Y_{34}$	$Y_{34,7} \vee \neg Y_{33,7} = 1, Y_{34,10} \vee \neg Y_{33,10} = 1$	$2^{-1}$
$RQ_{16}$	$(RQ_{16} \boxplus 0x8000) \lll 6 = RQ_{16} \lll 6 \boxplus 0x200000$	Negligible
$RQ_{28}$	$(RQ_{28} \boxplus 0x8000) \lll 7 = RQ_{28} \lll 7 \boxplus 0x400000$	Negligible
$RQ_{35}$	$(RQ_{35} \boxplus 0xffffc80) \lll 15 = RQ_{35} \lll 15 \boxplus 0xfe400000$	Negligible
Right Branch		$2^{-41-8} = 2^{-49}$
$X_{12}$	$X_{12,18} \neq X_{13,28}, X_{12,11} \neq X_{14,21}$	$2^{-2}$
$X_{11}$	$X_{11,11} \neq X_{12,21}$	$2^{-1}$
$LQ_{13}$	$(LQ_{13} \boxplus 0x8000) \lll 6 = LQ_{13} \lll 6 \boxplus 0x200000$	Negligible
$LQ_{14}$	$(LQ_{14} \boxplus 0xffe00000) \lll 7 = LQ_{14} \lll 7 \boxplus 0xf0000000$	$2^{-0.1}$
$LQ_{15}$	$(LQ_{15} \boxplus 0xfe00000) \lll 9 = LQ_{15} \lll 9 \boxplus 0xc0000020$	$2^{-0.6}$
$LQ_{16}$	$(LQ_{16} \boxplus 0xd0000020) \lll 8 = LQ_{16} \lll 8 \boxplus 0x1fd0$	$2^{-0.3}$
Filtering		$2^{-2}$
$X_{13}$	$X_{13,20} \neq X_{14,30}, X_{13,18} \neq X_{15,28}$	$2^{-2}$
$X_{15}$	$X_{15,13} = X_{14,3}, X_{15,4} = X_{14,26}$	$2^{-2}$
The number of free bits in $X_{14}$ and $X_{15}$ : $64 - 2 - 7 = 55$		
$LQ_{36}$	$(LQ_{36} \boxplus 0xe1c0000) \lll 7 = LQ_{36} \lll 7 \boxplus 0xe000007$	$2^{-0.2}$
$LQ_{37}$	$(LQ_{37} \boxplus 0xf2000000) \lll 14 = LQ_{37} \lll 14 \boxplus 0xffffc80$	$2^{-0.1}$
The expected number of valid $m_4$ : $2^{32-0.2-0.1-2} = 2^{29.7}$		
The expected number of starting points for a fixed $X_i$ ( $16 \leq i \leq 35$ ): $2^{55+29.7-4} = 2^{80.7}$		

**Table 8:** SFS collision for 37 steps of RIPEMD-160

$h_0 \sim h_4$	51c683bc e9cd8258 75924d6d b31d5b2b 9f1418b8
$M$	2a3e3e5d 2f3acda8 c5ab4a9c dc1f16ce 695a6d71 848cc0fe f11aa5a3 65da8473 9e6914b7 fe96a9cf da48b5c6 59b4296f <b>14a47a10</b> c0870c31 3b3e4837 7f4d5b3f
$M'$	2a3e3e5d 2f3acda8 c5ab4a9c dc1f16ce 695a6d71 848cc0fe f11aa5a3 65da8473 9e6914b7 fe96a9cf da48b5c6 59b4296f <b>14a4fa10</b> c0870c31 3b3e4837 7f4d5b3f
hash value	4ba88e59 fe3d1b6d 92324a6e 124af3ea e0206481

### 5.2.2 Attack on 38 steps of RIPEMD-160

Based on Table 9 and Table 14, we conclude that there will be  $2^{32-2} = 2^{30}$  possible values for  $X_{12}$  for a given starting point. After filtering, about  $2^{30-2} = 2^{28}$  are left. Since the

conditions on  $X_i$  ( $36 \leq i \leq 38$ ) hold with probability  $2^{-13.3}$  and the conditions on  $X_{13}$  hold with probability  $2^{-4}$ , we use **Strategy 2** to generate a new starting point, whose cost is about  $2^4$  computations. Since the right branch holds with probability  $2^{-52}$ , we expect that it will be required to generate  $2^{52-28} = 2^{24}$  starting points. As Table 7 shows,  $X_i$  ( $14 \leq i \leq 15$ ) can provide sufficient degrees of freedom to generate so many starting points for a fixed solution for  $X_i$  ( $16 \leq i \leq 35$ ). Specifically, for a valid  $m_4$ , there are 57 free bits in  $X_{14}$  and  $X_{15}$ , while the conditions on  $X_{13}$  hold with probability  $2^{-4}$ . Therefore, for a fixed solution for  $X_i$  ( $16 \leq i \leq 35$ ) and a valid  $m_4$ , we can expect to generate  $2^{57-4} = 2^{53}$  starting points in total. Thus, the time complexity to mount an SFS collision attack on 38 steps of RIPEMD-160 can be evaluated with the Eqn. 1, where

$$(p_1, p_2, p_3, p_4, n) = (2, 52, 4, 13.3, 2).$$

Therefore, the time complexity to find an SFS collision for 38 steps of RIPEMD-160 is  $2^{52}$ .

**Table 9:** Other conditions influencing the attack for the 38-step differential characteristic

	Conditions	Probability
$Y_{18}$	$Y_{18,31} = Y_{17,31}$	$2^{-1}$
$Y_{22}$	$Y_{22,9} = Y_{21,9}$	$2^{-1}$
$Y_{26}$	$Y_{26,20} = Y_{25,20}, Y_{26,19} = Y_{25,19}$	$2^{-2}$
$Y_{30}$	$Y_{30,0} = Y_{29,0}, Y_{30,29} = Y_{29,29}, Y_{30,30} = Y_{29,30}$	$2^{-3}$
$Y_{34}$	$Y_{34,7} \vee \neg Y_{33,7} = 1, Y_{34,10} \vee \neg Y_{33,10} = 1$	$2^{-1}$
$Y_{37}$	$Y_{37,3} \vee \neg Y_{36,3} = 1, Y_{37,0} \vee \neg Y_{36,0} = 1$	$2^{-1}$
$RQ_{16}$	$(RQ_{16} \boxplus 0x8000) \lll 6 = RQ_{16} \lll 6 \boxplus 0x200000$	Negligible
$RQ_{28}$	$(RQ_{28} \boxplus 0x8000) \lll 7 = RQ_{28} \lll 7 \boxplus 0x400000$	Negligible
$RQ_{35}$	$(RQ_{35} \boxplus 0xffffc80) \lll 15 = RQ_{35} \lll 15 \boxplus 0xfe400000$	Negligible
$RQ_{38}$	$(RQ_{38} \boxplus 0x7) \lll 6 = RQ_{38} \lll 6 \boxplus 0x1c0$	Negligible
	Right Branch	$2^{-43-9} = 2^{-52}$
$X_{12}$	$X_{12,18} \neq X_{13,28}, X_{12,11} = X_{14,21}$	$2^{-2}$
$X_{11}$	$X_{11,11} \neq X_{12,21}$	$2^{-1}$
$LQ_{13}$	$(LQ_{13} \boxplus 0x8000) \lll 6 = LQ_{13} \lll 6 \boxplus 0x200000$	Negligible
$LQ_{14}$	$(LQ_{14} \boxplus 0xffe00000) \lll 7 = LQ_{14} \lll 7 \boxplus 0xf0000000$	$2^{-0.1}$
$LQ_{15}$	$(LQ_{15} \boxplus 0xfe00000) \lll 9 = LQ_{15} \lll 9 \boxplus 0xc0000020$	$2^{-0.6}$
$LQ_{16}$	$(LQ_{16} \boxplus 0xd0000020) \lll 8 = LQ_{16} \lll 8 \boxplus 0x1fd0$	$2^{-0.3}$
	Filtering	$2^{-2}$
$X_{13}$	$X_{13,20} = X_{14,30}, X_{13,18} \neq X_{15,28}, X_{13,27} \neq X_{14,5}$	$2^{-3}$
$X_{15}$	$X_{15,13} = X_{14,3}, X_{15,4} = X_{14,26}, X_{15,21} \neq X_{16,31}$	$2^{-3}$
	The number of free bits in $X_{14}$ and $X_{15}$ : $64 - 3 - 4 = 57$	
$LQ_{36}$	$(LQ_{36} \boxplus 0xeffff04) \lll 7 = LQ_{36} \lll 7 \boxplus 0xffff81f8$	$2^{-0.1}$
$LQ_{37}$	$(LQ_{37} \boxplus 0x7fc8) \lll 14 = LQ_{37} \lll 14 \boxplus 0x1ff20000$	$2^{-0.2}$
$LQ_{38}$	$(LQ_{38} \boxplus 0xe0000000) \lll 9 = LQ_{38} \lll 9 \boxplus 0x1c0$	$2^{-3}$
	The expected number of valid $m_4$ : $2^{32-0.1-0.2-3-10} = 2^{18.7}$	
	The expected number of starting points for a fixed $X_i$ ( $16 \leq i \leq 35$ ): $2^{57+18.7-4} = 2^{71.7}$	

### 5.2.3 Attack on 40 steps of RIPEMD-160

Based on Table 10 and Table 15, we conclude that there will be  $2^{32-2} = 2^{30}$  possible values for  $X_{12}$  for a given starting point. After filtering, about  $2^{30-2} = 2^{28}$  are left. Since the conditions on  $X_i$  ( $36 \leq i \leq 39$ ) hold with probability  $2^{-21.8}$ , the conditions on  $X_{13}$  hold with probability  $2^{-4}$ , and  $LQ_{40}$  satisfies its equation with a probability close to 1, we use **Strategy 2** to generate a new starting point, whose cost is about  $2^4$  computations. Since the right branch holds with probability  $2^{-74.6}$ , we expect that it will be required to generate  $2^{74.6-28} = 2^{46.6}$  starting points. As Table 7 shows,  $X_i$  ( $14 \leq i \leq 15$ ) can provide sufficient degrees of freedom to generate so many starting points for a fixed solution for  $X_i$  ( $16 \leq i \leq 35$ ). Specifically, for a valid  $m_4$ , there are 57 free bits in  $X_{14}$  and  $X_{15}$ , while the conditions on  $X_{13}$  hold with probability  $2^{-4}$ . Therefore, for a fixed solution for  $X_i$  ( $16 \leq i \leq 35$ ) and a valid  $m_4$ , we can expect to generate  $2^{57-4} = 2^{53}$  starting points in total. Indeed, we can also store some solutions for  $m_4$  in an array with negligible memory. Then, as stated previously, we not only can choose valid values for  $X_{14}$  and  $X_{15}$ , but

also can randomly choose valid values for  $m_4$  from this array. In this way, the degrees of freedom of  $m_4$  can be utilized as well. Thus, the time complexity to mount an SFS collision attack on 40 steps of RIPEMD-160 can be evaluated with the Eqn. 1, where

$$(p_1, p_2, p_3, p_4, n) = (2, 74.6, 4, 21.8, 2).$$

Therefore, the time complexity to find an SFS collision for 40 steps of RIPEMD-160 is  $2^{74.6}$ .

**Table 10:** Other conditions influencing the attack for the 40-step differential characteristic

	Conditions	Probability
$Y_{18}$	$Y_{18,31} = Y_{17,31}$	$2^{-1}$
$Y_{22}$	$Y_{22,9} = Y_{21,9}$	$2^{-1}$
$Y_{26}$	$Y_{26,20} = Y_{25,20}, Y_{26,19} = Y_{25,19}$	$2^{-2}$
$Y_{30}$	$Y_{30,0} = Y_{29,0}, Y_{30,29} = Y_{29,29}, Y_{30,30} = Y_{29,30}$	$2^{-3}$
$Y_{37}$	$Y_{37,3} \vee \neg Y_{36,3} = 1$	$2^{-0.5}$
$Y_{38}$	$Y_{38,17} \vee \neg Y_{37,17} = 1, Y_{38,20} \vee \neg Y_{37,20} = 1$	$2^{-1}$
$RQ_{16}$	$(RQ_{16} \boxplus 0x8000) \lll 6 = RQ_{16} \lll 6 \boxplus 0x200000$	Negligible
$RQ_{28}$	$(RQ_{28} \boxplus 0x8000) \lll 7 = RQ_{28} \lll 7 \boxplus 0x400000$	Negligible
$RQ_{35}$	$(RQ_{35} \boxplus 0x380) \lll 15 = RQ_{35} \lll 15 \boxplus 0x1c00000$	Negligible
$RQ_{38}$	$(RQ_{38} \boxplus 0xffffffff) \lll 6 = RQ_{38} \lll 6 \boxplus 0xffffdc0$	Negligible
$RQ_{39}$	$(RQ_{39} \boxplus 0xffff2000) \lll 6 = RQ_{39} \lll 6 \boxplus 0xfc800000$	$2^{-0.1}$
$RQ_{40}$	$(RQ_{40} \boxplus 0xffffffffc8) \lll 14 = RQ_{40} \lll 14 \boxplus 0xffff20000$	Negligible
Right Branch		$2^{-66-8.6} = 2^{-74.6}$
$X_{12}$	$X_{12,18} = X_{13,28}, X_{12,11} \neq X_{14,21}$	$2^{-2}$
$X_{11}$	$X_{11,11} \neq X_{12,21}$	$2^{-1}$
$LQ_{13}$	$(LQ_{13} \boxplus 0x8000) \lll 6 = LQ_{13} \lll 6 \boxplus 0x200000$	Negligible
$LQ_{14}$	$(LQ_{14} \boxplus 0xffe00000) \lll 7 = LQ_{14} \lll 7 \boxplus 0xf0000000$	$2^{-0.1}$
$LQ_{15}$	$(LQ_{15} \boxplus 0xfe00000) \lll 9 = LQ_{15} \lll 9 \boxplus 0xbffffe0$	$2^{-0.6}$
$LQ_{16}$	$(LQ_{16} \boxplus 0x2fffffe0) \lll 8 = LQ_{16} \lll 8 \boxplus 0xffffe030$	$2^{-0.3}$
Filtering		$2^{-2}$
$X_{13}$	$X_{13,20} = X_{14,30}, X_{13,18} = X_{15,28}, X_{13,27} = X_{14,5}$	$2^{-3}$
$X_{15}$	$X_{15,13} = X_{14,3}, X_{15,4} = X_{14,26}, X_{15,21} \neq X_{16,31}$	$2^{-3}$
The number of free bits in $X_{14}$ and $X_{15}$ : $64 - 3 - 4 = 57$		
$X_{36}$	$X_{36,6} \vee \neg X_{35,6} = 1$	$2^{-0.5}$
$LQ_{36}$	$(LQ_{36} \boxplus 0x50c3fee0) \lll 7 = LQ_{36} \lll 7 \boxplus 0x61ff7028$	$2^{-1.3}$
$LQ_{37}$	$(LQ_{37} \boxplus 0xd6008f90) \lll 14 = LQ_{37} \lll 14 \boxplus 0x23e3f580$	$2^{-0.5}$
$LQ_{38}$	$(LQ_{38} \boxplus 0xdc1c0000) \lll 9 = LQ_{38} \lll 9 \boxplus 0x37ffffb8$	$2^{-0.6}$
$LQ_{39}$	$(LQ_{39} \boxplus 0xc8000048) \lll 13 = LQ_{39} \lll 13 \boxplus 0x8f900$	$2^{-0.4}$
$LQ_{38}$	$(LQ_{40} \boxplus 0xffff20700) \lll 15 = LQ_{40} \lll 15 \boxplus 0x37ffff9$	Negligible
The expected number of valid $m_4$ : $2^{32-1.3-0.5-0.6-0.4-1.9} = 2^{10.2}$		
The expected number of starting points for a fixed $X_i$ ( $16 \leq i \leq 35$ ): $2^{57+10.2-4} = 2^{63.2}$		

## 5.2.4 Experiments

To make the above theoretical analysis more convincing, we carried out the following experiments. For the  $t$ -step ( $t \geq 37$ ) differential characteristic and its corresponding solution for  $X_i$  ( $16 \leq i \leq 35$ ), we exhaust all possible values for  $X_{36}$  to verify the conditions on  $X_i$  ( $37 \leq i \leq t \leq 39$ ) and record how many valid  $m_4$  exists. Moreover, for a fixed valid  $m_4$ , we also randomly choose  $2^{32}$  valid values for  $(X_{14}, X_{15})$  and compute  $X_{13}$ . Then, we count the success times when the conditions on  $X_{13}$  hold (we will also check the conditions on  $X_{40}$  if analyzing 40 steps of RIPEMD-160). We list the experimental results in Table 11. Obviously, our theoretical analysis is reasonable.

## 6 Conclusion

Relying on the specifics of RIPEMD-160's message expansion, an SFS collision attack framework for reduced RIPEMD-160 is developed. Compared with previous SFS collision attack framework, this new framework allows us to attack as many steps of RIPEMD-160

**Table 11:** Experimental results

Steps	The number of valid $m_4$	Success times	Success probability
37	0x36d40000	0x10001110	$2^{-4}$
38	0xe0000	0xffff6f3	$2^{-4}$
40	0x2d80	0xf1bd6ed	$2^{-4}$

as possible. One more advantage of this new framework is negligible requirement of memory. As a direct result, we present the first colliding message pairs for 36 and 37 steps of RIPEMD-160 with time complexity  $2^{41}$  and  $2^{49}$  respectively. Moreover, benefiting from this framework, we can mount SFS collision attack on 38/40 steps of RIPEMD-160 with time complexity  $2^{52}/2^{74.6}$  respectively, thus extending the previously best known SFS collision attack on RIPEMD-160 by four steps.

## Acknowledgments

We thank the anonymous reviewers of ToSC Issue 3 for their many helpful and insightful comments. Fukang Liu and Zhenfu Cao are supported by National Natural Science Foundation of China (Grant No.61632012, 61672239). In addition, Fukang Liu is also supported by Invitation Programs for Foreigner-based Researchers of the National Institute of Information and Communications Technology (NICT). Takanori Isobe is supported by Grant-in-Aid for Scientific Research (B) (KAKENHI 19H02141) for Japan Society for the Promotion of Science. Gaoli Wang is supported by the National Natural Science Foundation of China (No. 61572125) and National Cryptography Development Fund (No. MMJJ20180201).

## References

- [DBP96] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In Dieter Gollmann, editor, *Fast Software Encryption - FSE 1996*, volume 1039 of *LNCS*, pages 71–82. Springer, 1996.
- [Dob96] Hans Dobbertin. Cryptanalysis of MD4. In Dieter Gollmann, editor, *Fast Software Encryption - FSE 1996*, volume 1039 of *LNCS*, pages 53–69. Springer, 1996.
- [DR06] Christophe De Cannière and Christian Rechberger. Finding SHA-1 characteristics: General results and applications. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.
- [EMS14] Maria Eichlseder, Florian Mendel, and Martin Schl affer. Branching heuristics in differential collision search with applications to SHA-512. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - FSE 2014*, volume 8540 of *LNCS*, pages 473–488. Springer, 2014.
- [LDM<sup>+</sup>19] Fukang Liu, Christoph Dobraunig, Florian Mendel, Takanori Isobe, Gaoli Wang, and Zhenfu Cao. Efficient collision attack frameworks for RIPEMD-160. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 117–149, 2019.

- [LMW17] Fukang Liu, Florian Mendel, and Gaoli Wang. Collisions and semi-free-start collisions for round-reduced RIPEMD-160. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017*, volume 10624 of *LNCS*, pages 158–186. Springer, 2017.
- [LP13] Franck Landelle and Thomas Peyrin. Cryptanalysis of full RIPEMD-128. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 228–244. Springer, 2013.
- [MNS11] Florian Mendel, Tomislav Nad, and Martin Schl affer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 288–307. Springer, 2011.
- [MNS13] Florian Mendel, Tomislav Nad, and Martin Schl affer. Improving local collisions: New attacks on reduced SHA-256. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 262–278. Springer, 2013.
- [MNSS12] Florian Mendel, Tomislav Nad, Stefan Scherz, and Martin Schl affer. Differential attacks on reduced RIPEMD-160. In Dieter Gollmann and Felix C. Freiling, editors, *Information Security - ISC 2012*, volume 7483 of *LNCS*, pages 23–38. Springer, 2012.
- [MPS<sup>+</sup>13] Florian Mendel, Thomas Peyrin, Martin Schl affer, Lei Wang, and Shuang Wu. Improved cryptanalysis of reduced RIPEMD-160. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, volume 8270 of *LNCS*, pages 484–503. Springer, 2013.
- [OSS12] Chiaki Ohtahara, Yu Sasaki, and Takeshi Shimoyama. Preimage attacks on the step-reduced RIPEMD-128 and RIPEMD-160. *IEICE Transactions*, 95-A(10):1729–1739, 2012.
- [SBK<sup>+</sup>17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017*, volume 10401 of *LNCS*, pages 570–596. Springer, 2017.
- [WLF<sup>+</sup>05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
- [WSL17] Gaoli Wang, Yanzhao Shen, and Fukang Liu. Cryptanalysis of 48-step RIPEMD-160. *IACR Transactions of Symmetric Cryptology*, 2017(2):177–202, 2017.
- [WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
- [WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
- [WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 1–16, 2005.

## A Differential Characteristics

The 36-step, 37-step, 38-step and 40-step differential characteristic are displayed in Table 12, Table 13, Table 14 and Table 15 respectively.

**Table 12:** 36-step differential characteristic

$\Delta m_{12} = 2^{15}$			
i	X	$\pi_1(i)$	Y
1		0	5
2		1	14
3		2	7
4		3	0
5		4	9
6		5	2
7		6	11
8		7	4
9		8	13
10		9	6
11		10	15
12		11	8
13	n	12	1
14	-nu	13	10
15	n	14	3
16	-0-0	15	12
17	n0	7	6
18	1	4	11
19	-011011-0111111u0	13	3
20	11010000	1	7
21	nnnn1nn101011111-1011nu110100u10	10	0
22	1001-01nu1-01u0n1--1n1nuunnnu1-	6	13
23	nn110u11n10nu100n001nnn10u11nnu1	15	5
24	1101nu0-10uun01nu1n0000nn1101u10	3	10
25	1uu1nn1-n011001n0u01uuu1n101uuu	12	14
26	1101011u1un0u10-u01uuuuu001-010	0	15
27	0u11u0010011n1-1uuun001111000111	9	8
28	11000100n11nnn0n11100n-10n0n0nn1	5	12
29	01n00nu000u01nnu-01uuu-ununun11n	2	4
30	n1u01n10u010011n000110-00000un1u	14	9
31	-1n100-nnn01-0101011-11-nnn0-10u	11	1
32	10n010-000--0-111110010-100--10u	8	2
33	-n	3	15
34	-0	10	5
35		14	1
36	-n-u	4	3

**Table 13:** 37-step differential characteristic

$\Delta m_{12} = 2^{15}$				
i	X	$\pi_1(i)$	Y	$\pi_2(i)$
1		0		5
2		1		14
3		2		7
4		3		0
5		4		9
6		5		2
7		6		11
8		7		4
9		8		13
10		9		6
11		10		15
12		11		8
13	0	12		1
14	u0	13	0	10
15	u	14	1	3
16	10	15	n	12
17	u1	7		6
18	0	4	1	11
19	n1	13	1	3
20	0010n100n0110-11u1uu1-0010--0--	1	u	7
21	001110u00-10110100010-u100011011	10		0
22	110-100u10u--0un010-001uuuu-00-	6	1	13
23	1un11nuu1-100u0u11u1uuun1001010u	15	1	5
24	110n11nn1001uu110u0n10u1u0nu001u	3		10
25	00n0010n0nn0un0000nnn01u11u101-0	12		14
26	0nnnnn-01101011-10nnnnnn011u0u10	0		15
27	010-0011nn0110u1100n01uu00n101u1	9		8
28	1nu11111uu1un0u01n0n1nnnn1100n00	5	n-un	12
29	u00-uuuu00u0101un0-0011u0001n0uu	2		4
30	111--001-1110u10u--0uuu00--uuu--	14	1-1	9
31	1-1-11--01-1nnn011-nnn--1--11	11	1	1
32	--0011011011-u01n-01-111-1111101	8	u	2
33	--011--0-1--0--	3	1	15
34	-----0-1-----	10	1-0-0	5
35	-----0-0-0	14	u-n	1
36	-----n--u	4	1-1	3
37	-----u-n-----	9		7

**Table 14:** 38-step differential characteristic

$\Delta m_{12} = 2^{15}$				
i	X	$\pi_1(i)$	Y	$\pi_2(i)$
1	-----	0	-----	5
2	-----	1	-----	14
3	-----	2	-----	7
4	-----	3	-----	0
5	-----	4	-----	9
6	-----	5	-----	2
7	-----	6	-----	11
8	-----	7	-----	4
9	-----	8	-----	13
10	-----	9	-----	6
11	-----	10	-----	15
12	-----	11	-----	8
13	-----n-----	12	-----	1
14	--u0-----	13	-----0-----	10
15	-n-----u-----n-----	14	-----1-----	3
16	-0-1-----u-----1uu-----	15	-----n-----	12
17	u-----1-0-----1-000-----	7	-----	6
18	0-----1-----0111-----u-----	4	-----1-----	11
19	01-101100--0---nu-----n-01---10	13	-----1-----	3
20	-----01u1---01uu1--10011-----	1	u-----	7
21	110111010110000011---u-00--1000	10	-----	0
22	1-00n-unu--un-nn0-unnnnnnnnnn1-	6	1-----	0
23	n0uuu-100-10nuuu0uuuu1nun101nu11	15	1-----	5
24	1011111101un01u1001100000-1-1000	3	-----un-----	10
25	uu1110n0010-01000-0nuu01u11101n0	12	-----	14
26	1nu101u--u0-01--00u00--11u-011-	0	-----0-01-----	15
27	-1000uu0-10u1n0--0nnn110--n-1--1	9	-----1-11-----	8
28	0u10u00--100-001110000-0000-01--	5	-----n-un-----	12
29	1--01--0111n-01-00---n0n--01n---	2	-----	4
30	n--0--0u-11011--u---001--n10---	14	-----1-1-----	9
31	00uuuuuu--10nu--1-00-10--1111--	11	-1-----	1
32	u-1111n1--101--n--u--0-nuu1-0--	8	-u-----	2
33	0-01--1-1-un---0--0--1-00001---	3	-1-----	15
34	u11n--0--0-11---1--1-0--0---	10	-1---0-0-----	1
35	1001--1-1-----0-----	14	-----u-n-----1-1-----	1
36	-----n--u-0-0-----1-----1	4	-----1-1-----n-u-----	3
37	-----1-1-u-n-----	9	-----1-1-----	7
38	-----	15	-----n-u-----	14



**Table 15:** 40-step differential characteristic

$\Delta m_{12} = 2^{15}$				
i	X	$\pi_1(i)$	Y	$\pi_2(i)$
1		0		5
2		1		14
3		2		7
4		3		0
5		4		9
6		5		2
7		6		11
8		7		4
9		8		13
10		9		6
11		10		15
12		11		8
13	n	12		1
14	u	13	0	10
15	u	14	1	3
16	0-1	15	n	12
17	u1	7		6
18	0	4	1	11
19	1	13	1	3
20	u	1	u	7
21	1-0	10	u	0
22	0-01-10	6	1	13
23	100010u0	15	1	5
24	-1-n0u0-1-011110unnnn-01un-	3		10
25	-1-100010n0u0nnn010011un1001-10-	12		14
26	unu001uu001u01001u10-01111u11u1	0	0-01	15
27	011un011nnuuuuuuu10000u00u100u0	9	1-11	8
28	00111nun0111u1nn0nnnnn10n00nnn1	5	n-un	12
29	nn111010n0101010100-0u1un10101n	2		4
30	011-u1101000n-1101100uu11101u0	14	1-1	9
31	n001u11-n-0u1110nn-u0100111110n	11	-1	1
32	1000uun01-n1n011u01111010-n0nu01	8	u	2
33	0u1110101-00u01u1-11n0-u-0111n0	3	1-1	15
34	-11n-1n00-0u1-n--000001--00101-	10	-1--0-0--0--0--0--0--1	5
35	0u000000--01--1--11111-000-n-	14	n-u	1
36	1-----0-0-----1-	4	1-1-----n-u-----1	3
37	-----1u-1n-----	9	-1-----1-----0-----10-10-----0	7
38	-----10-10-----1-1-----	15	-0--0--0-----u-u-0--1	14
39	-----0n-0n-----	8	u-n-0-----1-0--n--u	6
40	-----n-u-----u-n	1	-----1-0-----	9