Improved Semi-Free-Start Collision Attacks on RIPEMD-160 (Full Version)

Zhuolong Zhang[1,2,3], Muzhou Li[1,2,3], Haoyang Wang[4], Shiqi Hou[5,6], Wei Wang[1,3,2], and Meiqin Wang[1,2(✉)]

[1] School of Cyber Science and Technology, Shandong University, Qingdao, China
{zhuolongzhang, muzhouli}@mail.sdu.edu.cn
{weiwangsdu, mqwang}@sdu.edu.cn
[2] State Key Laboratory of Cryptography and Digital Economy Security, Shandong University, Qingdao, China
[3] Quan Cheng Laboratory, Jinan, China
[4] Shanghai Jiao Tong University, Shanghai, China
haoyang.wang@sjtu.edu.cn
[5] State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China
houshiqi@iie.ac.cn
[6] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Abstract. As an ISO/IEC standard, RIPEMD-160 has been extensively studied for (Semi-Free-Start) collision attacks. A significant breakthrough was achieved at FSE 2024 with the first 41-, 42-, and 43-step SFS collision attacks, which leveraged an automatic search model (EUROCRYPT 2023) and a message modification strategy (FSE 2020). However, these attacks are limited by reliance on heuristic objective functions and suboptimal message modification techniques. This paper enhances the existing framework from two perspectives. Firstly, we refine the automatic search model by incorporating a holistic objective function that considers all critical probability components, moving beyond simple Hamming weight. Secondly, we introduce two generic techniques to further improve (SFS) collision attacks: the first application of differential clustering and a dedicated message modification strategy. As a result, we present the first valid SFS collision attack on 44-step RIPEMD-160. Additionally, we significantly reduce the time complexities of existing attacks on 41-, 42-, and 43-step variants, making it feasible to find colliding message pairs for 41- and 42-step versions within practical time for the first time.

Keywords: SFS Collision Attack · RIPEMD-160 · Signed Difference · SAT/SMT

1 Introduction

Hash functions are fundamental components of cryptographic primitives and play a critical role in constructing secure systems. A cryptographic hash function

must satisfy some security requirements, among which collision resistance is one of the most important. Signed difference is the most powerful method to mount collision attacks on the MD-SHA hash family, proposed by Wang et al. at EUROCRYPT 2005 [22]. In combination with message modification techniques, a series of hash functions in the MD-SHA family were subsequently broken, including MD4 [22], RIPEMD [22], MD5 [24], SHA-0 [1,25], SHA-1 [7,8,20,23], and RIPEMD-128 [6]. Despite these successful attacks, RIPEMD-160 remains secure, largely owing to its complex double-branch structure.

As an ISO/IEC standardized cryptographic hash function, RIPEMD-160 has seen widespread adoption, particularly together with SHA-256 for generating Bitcoin addresses. Over the past decade, its security has been extensively studied, with a primary focus on collision attacks. Numerous works have been conducted in this area by Mendel et al., Wang et al., and Liu et al. [11,12,14,16–18,21], based on a guess-and-determine (GnD) technique [3] to search for differential characteristics. Among these efforts, the best collision attack reaches 34 steps, while the best practical one covers 31 steps. For semi-free-start (SFS) collision attacks starting from the beginning of the cipher, the most effective approach reaches 40 steps. There also exist SFS attacks starting from the middle of the cipher, with the best one reaching 48 steps [21].

At EUROCRYPT 2023, Liu et al. [15] proposed an MILP-based search model and mounted a valid 36-step collision attack on RIPEMD-160. At FSE 2024, Li et al. [9] re-implemented this MILP-based tool using SAT/SMT, achieving the first practical 40-step collision attack under a new message difference. Furthermore, they adopted the attack framework from [12] to construct SFS collision attacks on 41, 42 and 43 steps of RIPEMD-160. Li et al. [9] concluded that their work likely represents the best combination of the techniques proposed in [15] and [12]. However, they were still unable to reach 44 steps, as the probability of the right-branch differential characteristic remained significantly lower than 2^{-80}. They raised the question of whether alternative techniques could further enhance the SFS collision attack framework. We identify several limitations in the existing frameworks that may have hindered progress toward more effective attacks.

On one hand, when searching for high-probability signed differential characteristics, both the GnD technique and the two latest search models typically adopt Hamming weight as the objective function, aiming to identify differential trails that are as sparse as possible. Such a strategy is commonly adopted in the analysis of the MD-SHA family. At CRYPTO 2005, Wang et al. emphasized that finding good disturbance vectors with low Hamming weight is a crucial first step in attacking the full SHA-1 [23]. After constructing several differential paths, they selected the optimal disturbance vector by considering the number of conditions. Nevertheless, their construction of differential paths still relied primarily on low Hamming weight. Mendel et al. [18] observed that differential characteristics with lower Hamming weights generally exhibit higher probabilities than those with higher weights and developed a series of automatic search tools [16–18] based on the GnD technique, which aimed to find differences with low Hamming weights in both message words and internal state words. At

AISACRYPT 2017, Liu et al. [14] proposed a theoretical method for calculating the differential probability of RIPEMD-160 and showed that it is influenced by multiple factors beyond Hamming weight. Nonetheless, they still used the search tool from [16] to automatically find differential paths as parse as possible for RIPEMD-160. Subsequently, [11] and [12] both employed the GnD tool to search for many sparse differential paths, and then selected those with relatively high probabilities. In addition, Wang et al. may consider the modular differences when deduced the collision-generating differential characteristics by hand in attacking the MD5 [24]. However, they did not explain more in the paper and such hand-crafted analysis is highly technical and time-consuming. In general, relying solely on Hamming weight minimization, a heuristic approach, may not yield optimal characteristics.

On the other hand, in the message modification phase of RIPEMD-160, existing attacks typically satisfy a set of conditions derived from the identified signed differential characteristics. Notably, multiple distinct condition sets can lead to valid collisions, as the goal of constructing (SFS) collision attacks is to ensure that modular differences in the left branch are canceled by those in the right branch. Moreover, the complexity of condition verification varies significantly depending on the partitioning and ordering of conditions, which has not been adequately explored in prior works.

1.1 Contributions

In this paper, we improve the current (SFS) collision attack framework in two aspects. First, we enhance the automatic search model to identify differential characteristics with higher probabilities. Second, we introduce two generic techniques to further improve these probabilities and optimize the overall attack process.

More Holistic Objective Function. Liu et al. [14] identified three contributing factors to differential probability: (1) Hamming weight, (2) the probability derived by the operation $\lll s$, and (3) the number of bit conditions derived by Boolean functions. To identify optimized differential trails, we propose a holistic objective function that incorporates not only Hamming weight but also the probabilities associated with factors (2) and (3), as illustrated in Figure 1. To enable this, we present modeling approaches for these two probability components in Sections 3.1 and 3.2, respectively. We implement the enhanced automatic search model using the SAT/SMT solver STP[1].

Application of Differential Clustering Effect. To further improve the probabilities of our newly identified characteristics, we consider the differential clustering effect in the last several steps of the right and/or left branch. This effect manifests in three key ways (detailed in Section 4.1 and illustrated in Figure 3): multiple differential trails can exist within a single step function; clustering can

[1] https://stp.github.io/

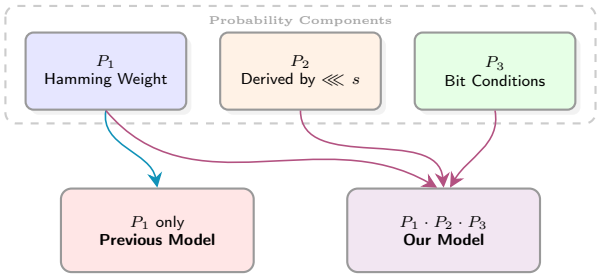


Fig. 1: The comparison of objective functions across different models.

occur across multiple consecutive steps; and signed differences in the final steps can often be adjusted to alternative values as long as the resulting modular differences between the two branches still cancel out.

Dedicated Message Modification Technique. In Section 4.2, we propose a dedicated message modification technique to improve the efficiency of (SFS) collision attacks. This is achieved by dividing the uncontrolled conditions into independent subsets and exploring all possible interleavings to determine the optimal verification order. Then, applying the step-by-step early-abort strategy based on this order can further reduce the computational cost.

Applications to RIPEMD-160. Building on the three contributions above, we present the first successful SFS collision attack on 44-step RIPEMD-160. Notably, we are the first to find colliding message pairs for the 41-step and 42-step variants within practical time. For comparison, representative SFS collision attacks are summarized in Table 1.

All source code and detailed instructions for the experiments presented in this paper are available at `https://github.com/Zhang-SDU/ripemd160_sfs_collision_attack`.

1.2 Organization

Section 2 presents the notations and some preliminary concepts used in this work. In Section 3, we introduce the improved automated search model. In Section 4, we describe two generic techniques for further improvement. Then, we show how to apply the proposed techniques to improve SFS collision attacks on RIPEMD-160 in Section 5. Finally, we conclude the paper in Section 6.

2 Preliminaries

In this paper, $x[i]$ represents the i-th bit of x, where $x[0]$ denotes the rightmost bit. The signed difference between x' and x is denoted by ∇x. Specifically, $\nabla x[i] = [\mathtt{n}]$ if $(x[i] = 0, x'[i] = 1)$; $\nabla x[i] = [\mathtt{u}]$ if $(x[i] = 1, x'[i] = 0)$; $\nabla x[i] = [\mathtt{=}]$

Table 1: Summary of SFS collision attacks on RIPEMD-160.

Starting step	Steps	Practical	Time	Memory	Reference
First Step	36		$2^{70.4}$	2^{64}	[17]
	36		$2^{55.1}$	2^{32}	[14]
	37	✓	$2^{49.0}$	negligible	[12]
	40		$2^{74.6}$	negligible	[12]
	41		$2^{59.7}$	negligible	[9]
		✓	$\mathbf{2^{49.8}}$	**negligible**	**Section 5.3**
	42		$2^{67.3}$	negligible	[9]
		✓	$\mathbf{2^{53.9}}$	**negligible**	**Section 5.3**
	43		$2^{74.8}$	negligible	[9]
			$\mathbf{2^{62.3}}$	**negligible**	**Section 5.3**
	44		$\mathbf{2^{76.9}}$	**negligible**	**Section 5.3**
Middle Step	36	✓	-	-	[16]
	42		$2^{75.5}$	2^{64}	[17]
	48		$2^{77.0\dagger}$	2^{64}	[21]
	48		$\mathbf{2^{74.6}}$	$\mathbf{2^{64}}$	**Section 5.3**

† Adjusted due to underestimation in the original complexity evaluation.

if $x'[i] = x[i]$; $\nabla x[i] = [0]$ if $x'[i] = x[i] = 0$; $\nabla x[i] = [1]$ if $x'[i] = x[i] = 1$. The Hamming weight of ∇x, denoted by $\mathbf{H}(\nabla x)$, is defined as the total number of `n` and `u` in ∇x. In addition, $\boxplus$ and $\boxminus$ represent the modular addition and subtraction on 32 bits, respectively. The modular difference is denoted by δx, i.e. $\delta x = x' \boxminus x$.

Property 1. A modular difference corresponds to multiple signed differences, while each signed difference uniquely determines its associated modular difference. In this context, a signed difference is referred to as an *expansion* of the corresponding modular difference [15].

Example 1. The following are two different signed differences, both of which are expansions of the modular difference $2^0 = 1$.

$$\nabla x_0 = [\texttt{==== ==== ==== ==== ==== ==== ==== ===n}],$$
$$\nabla x_1 = [\texttt{==== ==== ==== ==== ==== =nuu uuuu uuuu}].$$

2.1 Description of RIPEMD-160

RIPEMD-160 is a 160-bit hash function proposed by Dobbertin et al. at FSE 1996 [4], which is based on the Merkle-Damgård structure and an enhanced version of RIPEMD [2]. It adopts a compression function $H(CV, M)$ that takes a

512-bit message block M and a 160-bit chaining variable CV as inputs, generating a 160-bit output. In our Semi-Free-Start (SFS) collision attacks, the goal is to find a pair (M, M') such that $H(CV, M) = H(CV, M')$, where $M \neq M'$ and CV can be any 160-bit value.

The compression function $H(CV, M)$ employs a double-branch structure, referred to as the left and right branches, respectively. Each branch consists of 80 steps, divided into 5 rounds of 16 steps each. Hereafter, we distinguish these two branches using different symbols. Specifically, X_i denotes the variables in the left branch, while Y_i denotes those in the right branch.

Initialization. The compression function begins with an initialization phase, in which the 160-bit chaining variable CV at the input is divided into five 32-bit words $(h_0, \ldots, h_4)$. These five words are then used to initialize the states of these two parallel branches:

$$X_{-5} = h_0^{\ggg 10}, X_{-4} = h_4^{\ggg 10}, X_{-3} = h_3^{\ggg 10}, X_{-2} = h_2, X_{-1} = h_1;$$
$$Y_{-5} = h_0^{\ggg 10}, Y_{-4} = h_4^{\ggg 10}, Y_{-3} = h_3^{\ggg 10}, Y_{-2} = h_2, Y_{-1} = h_1.$$

Note that in SFS collision attacks, these h_i can be chosen freely.

Message Expansion. The 512-bit input message block is divided into 16 message words $(m_0, m_1, \ldots, m_{15})$, each of size 32 bits. Each m_i is used once per round, following different permutation orders π_l and π_r for the left and right branches, respectively.

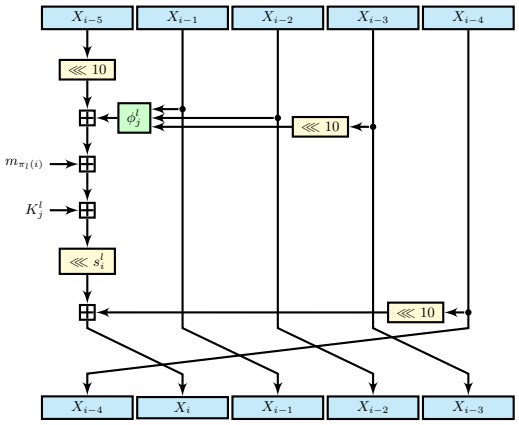


Fig. 2: The step function of left branch in RIPEMD-160.

Step Function. Figure 2 illustrates the step function of the left branch. The right branch has a similar structure and is therefore omitted. At step i of round

j, the internal state is updated as follows:

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

where $i = 0, 1, \ldots, 79$ and $j = 0, 1, 2, 3, 4$. Details of Boolean functions ϕ_j^l, ϕ_j^r and round constants K_j^l, K_j^r are given in Table 2. For more information on RIPEMD-160, please refer to [4].

Table 2: Boolean functions and round constants used in RIPEMD-160

Round j	ϕ_j^l	ϕ_j^r	K_j^l	K_j^r	Function	Expression
0	XOR	ONX	0x00000000	0x50a28be6	$XOR(x,y,z)$	$x \oplus y \oplus z$
1	IFX	IFZ	0x5a827999	0x5c4dd124	$IFX(x,y,z)$	$(x \wedge y) \oplus (\overline{x} \wedge z)$
2	ONZ	ONZ	0x6ed9eba1	0x6d703ef3	$ONX(x,y,z)$	$x \oplus (y \vee \overline{z})$
3	IFZ	IFX	0x8f1bbcdc	0x7a6d76e9	$IFZ(x,y,z)$	$(x \wedge z) \oplus (y \wedge \overline{z})$
4	ONX	XOR	0xa953fd4e	0x00000000	$ONZ(x,y,z)$	$(x \vee \overline{y}) \oplus z$

Finalization. The finalization is performed after all 80 steps have been executed in both branches, and the 160-bit output $(h'_0, h'_1, \ldots, h'_4)$ is computed as follows:

$$\begin{aligned} h'_0 &= h_1 \boxplus X_{78} \boxplus Y_{77}^{\lll 10}, \\ h'_1 &= h_2 \boxplus X_{77}^{\lll 10} \boxplus Y_{76}^{\lll 10}, \\ h'_2 &= h_3 \boxplus X_{76}^{\lll 10} \boxplus Y_{75}^{\lll 10}, \\ h'_3 &= h_4 \boxplus X_{75}^{\lll 10} \boxplus Y_{79}, \\ h'_4 &= h_0 \boxplus X_{79} \boxplus Y_{78}. \end{aligned} \tag{1}$$

2.2 Main Idea of Automatic Search Methods in [9, 15]

To construct (SFS) collision attacks, the key challenge is to identify signed differential characteristics with high probabilities for the uncontrolled parts. We briefly recall the main ideas behind the automatic models proposed in [9, 15].

Since modular addition is the primary operation in the MD-SHA hash family, the propagation of modular differences is analyzed within the step function, in addition to the Boolean function. This distinction arises because Boolean functions operate at the bit level. Apart from the message word, the inputs to the step function either directly participate in Boolean functions or undergo certain rotation operations. Therefore, signed differences are used to describe these inputs.

When considering the signed differences of the inputs, the operation $\lll 10$ only affects the ordering of variables. Therefore, the step function of RIPEMD-160 can be simplified as follows:

$$a_5 = a_1 \boxplus (F(a_4, a_3, a_2) \boxplus a_0 \boxplus m \boxplus c)^{\lll s}, \tag{2}$$

where $a_0, \ldots, a_5$ and m are 32-bit variables, c is a 32-bit constant, $s \in [0, 31]$ is an integer and F is a Boolean function defined in Table 2. Liu et al. [15] decompose Equation (2) by introducing several intermediate 32-bit variables $b_0, \ldots, b_5$ where:

$$\begin{aligned} &b_0 = m \boxplus c, b_1 = F(a_4, a_3, a_2), b_2 = b_0 \boxplus b_1, \\ &b_3 = b_2 \boxplus a_0, b_4 = b_3^{\lll s}, b_5 = a_1 \boxplus b_4, a_5 = b_5. \end{aligned} \tag{3}$$

The basic idea to construct the model is as follows:

- **Describe Signed Differences.** Use two binary variables (v, d) to describe the signed difference, where $(0, 0)$, $(0, 1)$ and $(1, 1)$ correspond to [=], [n] and [u], respectively.
- **Model the Boolean Function.** For the Boolean function F, i.e. $(\nabla a_4, \nabla a_3, \nabla a_2) \to \nabla b_1$, two models are introduced. The first model is called *fast filtering model*, which captures only valid signed difference transitions of F. Another model *full model* introduces three so-called monitoring variables (a_4, a_3, a_2) to detect the contradictions in Boolean functions across different steps.
- **Model the Modular Addition.** For $b_0 = m \boxplus c$, $b_2 = b_0 \boxplus b_1$, $b_3 = b_2 \boxplus a_0$ and $b_5 = a_1 \boxplus b_4$, it is necessary to model the transition of modular difference. However, since modular addition is linear with respect to modular difference, a single signed difference can represent the corresponding modular difference. For operations like $z = x \boxplus y$, a deterministic propagation rule is used, which means that for each given pair $(\nabla x, \nabla y)$, only one valid ∇z is considered.
- **Model the Special Operation.** The special operation $b_4 = b_3^{\lll s}$ is no longer linear with respect to modular difference. Liu et al. [14] showed that a given δb_3 can result in up to four possible values of δb_4. Since modular addition uses a deterministic propagation rule, only one ∇b_3 is retained, leading to a single ∇b_4, potentially excluding other possible options. To address this, Liu et al. [15] proposed two modeling strategies: *the first strategy* and *the second strategy*, which differ in whether low-probability propagations are permitted.
- **Model the Expansions of the Modular Difference.** For $a_5 = b_5$, there is also a possible ∇a_5 to represent δa_5. However, since a_5 is the final output of the step function, different ∇a_5 (corresponding to the same modular difference) can lead to different trails. Moreover, it is not possible to predict in advance which signed difference yields the best trail. To address this, Liu et al. [15] proposed a model called *modeling the expansion of the modular difference*, which computes all possible signed differences $\nabla a_5'$ for a given ∇a_5 such that they correspond to the same modular difference. This approach is also employed in the second strategy for ∇b_3.
- **Objective Function.** The objective function minimizes the sum of Hamming weight of all signed differences ∇a_i, which often leads to trails that are as sparse as possible.

In addition, several supplementary techniques are used to detect possible contradictions. For more details, please refer to [15].

2.3 Probability Evaluation of the Special Operation: $b_4 = b_3^{\lll s}$

In 2007, Fouque et al. [5] discussed several useful properties of cryptographic operations, including the interaction between modular difference and rotation. Notably, they introduced a simple expression to describe this interaction and provided a detailed proof for the first time. Subsequently, Liu et al. [14] examined this interaction from a different perspective and demonstrated how to evaluate the probability of the special operation in RIPEMD-160. In this paper, we will adopt their method to model probability propagation.

As mentioned above, the focus is on the propagation of modular difference within the step function. Since the special operation $b_4 = b_3^{\lll s}$ is nonlinear with respect to modular difference, Liu et al. [14] showed that additional implicit conditions should be imposed to ensure the correct propagation of these differences. These implicit conditions can be unified in the following form due to the similarity of the left and right branches:

$$(Q \boxplus \alpha)^{\lll s} = Q^{\lll s} \boxplus \beta, \tag{4}$$

where α and β are constants corresponding to the modular differences of b_3 and b_4 in Equation (3), i.e. δb_3 and δb_4. The variable Q denotes the value of b_3, and $s \in [1, 31]$.

In particular, the method for calculating the differential probability, under which Equation (4) holds, can be summarized as follows:

1. Decompose $Q \boxplus \alpha$ into high and low parts based on the value of s, i.e. $Q \boxplus \alpha = H \parallel L$, where H is an s-bit variable representing the higher s bits of $Q \boxplus \alpha$ and L is a $(32-s)$-bit variable representing the lower $(32-s)$ bits. Taking into account carry effect, H and L can be expressed as:

$$\begin{aligned} H &\equiv Q[31 \sim (32-s)] + \alpha[31 \sim (32-s)] + carry_0 \bmod (2^s), \\ L &\equiv Q[(31-s) \sim 0] + \alpha[(31-s) \sim 0] \bmod (2^{32-s}), \end{aligned} \tag{5}$$

   where $carry_0$ denotes the carry from the $(31-s)$-th bit to the $(32-s)$-th bit during the computation of $Q \boxplus \alpha$.
2. Similarly, $Q^{\lll s} \boxplus \beta$ can be expressed as a concatenation of the corresponding high $(32-s)$-bit variable H' and the low s-bit L', i.e. $Q^{\lll s} \boxplus \beta = H' \parallel L'$. Let $carry_1$ represents the carry from the $(s-1)$-th bit to the s-th bit in the computation of $Q^{\lll s} \boxplus \beta$, then H' and L' can be written as:

$$\begin{aligned} H' &\equiv Q[(31-s) \sim 0] + \beta[31 \sim s] + carry_1 \bmod (2^{32-s}), \\ L' &\equiv Q[31 \sim (32-s)] + \beta[(s-1) \sim 0] \bmod (2^s). \end{aligned} \tag{6}$$

3. Finally, based on Equations (5) and (6), the probability that Equation (4) holds is given by $\Pr(H = L' \text{ and } L = H')$. For given constants α and β, the values of $carry_0$ and $carry_1$ are determined. Thus, computing the probability essentially reduces to counting the number of Q values that produce the specified carries. This count depends on the specific bit positions where constants α and β are equal to 1. Property 2 standardizes the computation method.

Property 2. Let B be an n-bit variable and C an n-bit constant. Let k_i denote the bit position such that $C[k_i] = 1$, where $i \in [0, \ldots, m]$. For the modular addition $B + C \bmod 2^n$, let *carry* denote the carry from the n-th bit to the $(n+1)$-th bit. For simplicity, we define as κ the event that $carry = 1$ and as ω the event that $carry = 0$. Therefore, we have

- If $C = 0$, then $\Pr(\kappa) = 0$. Otherwise, $\Pr(\kappa) = \sum_{i=0}^{m} 2^{-(n-k_i)}$;
- If $C = 0$, then $\Pr(\omega) = 1$. Otherwise, $\Pr(\omega) = 1 - \sum_{i=0}^{m} 2^{-(n-k_i)}$.

Example 2. To aid understanding, we present a concrete example illustrating the computation of the probability that the equality $(Q \boxplus \texttt{0x8ffd0000})^{\lll 12} = Q^{\lll 12} \boxplus \texttt{0xd0000900}$ holds. According to Table 3, we compute:

$$\begin{aligned}
\Pr(H = L') &= \Pr(carry_0 = 1) \\
&= \Pr(Q[19] = 1) + \Pr(Q[19 \sim 18] = 01) + \Pr(Q[19 \sim 16] = 0011) \\
&= 2^{-1} + 2^{-2} + 2^{-4} \\
\Pr(L = H') &= \Pr(carry_1 = 0) \\
&= 1 - \Pr(Q[31] = 1) - \Pr(Q[31 \sim 28] = 0111) \\
&= 1 - 2^{-1} - 2^{-4}
\end{aligned}$$

Therefore, $\Pr(Q \boxplus \texttt{0x8ffd0000})^{\lll 12} = Q^{\lll 12} \boxplus \texttt{0xd0000900}) = \Pr(H = L') \times \Pr(L = H') \approx 2^{-1.492}$.

Table 3: Calculation of the probability.

Q	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Q																																
α	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	H												L																			
$Q^{\lll 12}$	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20
$Q^{\lll 12}$																																
β	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	H'																				L'											

3 Enhanced Automated Search Model

In this section, we begin by analyzing the limitations of the models proposed in [9,15]. We then detail our modeling strategy, explaining how it addresses these limitations to provide a more effective automated search model for RIPEMD-160.

Motivations. As discussed by Liu et al. [14], the differential probability for one step of RIPEMD-160 consists of the following three components:

1. The number of differential bits, i.e., the Hamming weight.

2. The probability derived by the operation $\lll s$ within the step function.
3. The number of bit conditions derived by the Boolean functions.

However, previous models [9, 15] considers only the first factor, i.e. Hamming weight, as the objective function, thereby aiming to identify the sparsest differential characteristic. Although using the Hamming weight as the sole objective function reduces the search space and simplifies the modeling process, it does not accurately reflect the actual differential probability. Consequently, potentially better characteristics with higher probabilities may be overlooked.

To handle the rotation operation $\lll s$, two different modeling strategies were proposed to cover as many cases as possible. *The first strategy* [15] restricts δb_4 to values that occur with high probability, whereas *the second strategy* [15] permits certain low-probability transitions $\delta b_3 \to \delta b_4$. However, neither strategy explicitly models the corresponding probabilities nor considers their impact on the overall differential characteristic probability.

For Boolean functions, beyond the *fast filtering model* [15] (which considers only transitions of signed differences), *monitoring variables* [15] were introduced. These variables serve to detect contradictions across different steps. This approach is motivated by the fact that each possible signed difference transition imposes specific bit constraints on the inputs, as illustrated in Example 3. However, the probability of satisfying these constraints was not incorporated into previous models, leading to an incomplete estimation of the overall differential characteristic probability.

Example 3. For $w = IFZ(x, y, z)$, when $(\nabla x[i], \nabla y[i], \nabla z[i], \nabla w[i])$=[==nn], there are two bit conditions: $x[i] = 1$ and $y[i] = 0$.

To overcome the above limitations, we improve upon the previous models [9, 15] by designing a more holistic objective function that incorporates all three components of the step differential probability. This refinement enables a more precise and effective automated search process, guiding the solver toward higher-probability differential characteristics.

To maintain consistency with prior works, we adopt the same representation of signed difference as in [9,10,15]. Specifically, for a binary variable κ, its signed difference $\nabla\kappa$ is represented as $(\kappa_v, \kappa_d) \in \{(0,0),(0,1),(1,1)\}$, corresponding to [=], [n] and [u], respectively. Our implementation is built on the SAT/SMT solver STP. The construction of our improved model is detailed in the following.

3.1 Modeling the Probability Derived by $b_4 = b_3^{\lll s}$

Although a theoretical method for computing the probability of the operation $b_4 = b_3^{\lll s}$ has been introduced in the literature, as discussed in Section 2.3, incorporating this probability evaluation into an automated search model is a nontrivial task. 1) previous models [9,15] consider only the propagation of signed difference, while the probability evaluation here depends on the modular difference. 2) the probability calculation differs depending on the value of the carry bit, thus it is necessary to determine $carry_0$ and $carry_1$ based on the constants

α and β involved in the operation. 3) the probability calculation methods vary significantly under different carry bit values, so the model must be capable of handling both cases simultaneously in a unified manner.

Consequently, modeling the differential probability of $\lll s$ requires completing the following three tasks:

- Modeling the **conversion** from signed difference to modular difference.
- Modeling the **determination** of the carry bit.
- Modeling the **probability calculation** based on the carry bit.

Modeling the Conversion. In this part, we describe how to compute the modular difference δz for a given signed difference ∇z. Recall that for signed differences, "n" represents a bit flip from 0 to 1, corresponding to a difference value of 1, whereas "=" indicates no change, corresponding to a difference value of 0. We can simplify the conversion by introducing a new variable $\nabla\xi$: if $\nabla\xi[i] = [\mathtt{n}]$, then $\delta z[i] = 1$; if $\nabla\xi[i] = [=]$, then $\delta z[i] = 0$. Thus, the problem reduces to modeling $\nabla\xi$ from a given ∇z such that $\nabla\xi$ contains only n and =, and satisfies $\delta\xi = \delta z$.

To achieve this, we introduce an auxiliary variable ∇c of size 33, initialized with $\nabla c[0] = [=]$, to propagate carry information. This approach is grounded in the properties of binary arithmetic. For example, $0 = 0$, $2^i = 2^i$, $\boxminus 2^i = 2^i \boxminus 2^{i+1}$, $2^i \boxplus 2^i = 2^{i+1}$, and $\boxminus 2^i \boxminus 2^i = \boxminus 2^{i+1}$. Since each signed difference uniquely determines a modular difference, we can define deterministic propagation rules for the transformation $(\nabla z[i], \nabla c[i]) \to (\nabla\xi[i], \nabla c[i+1])$, as summarized in Table 4.

Table 4: The propagation rules for $(\nabla z[i], \nabla c[i]) \to (\nabla\xi[i], \nabla c[i+1])$.

[== → ==],				
[=n → n=],	[=u → nu],	[n= → n=],	[u= → nu],	
[nn → =n],	[uu → =u],	[nu → ==],	[un → ==].	

According to the method for describing signed difference, the 9 propagation rules in Table 4 can be converted into 9 corresponding tuples of the form

$$(z_v[i], z_d[i], c_v[i], c_d[i], \xi_v[i], \xi_d[i], c_v[i+1], c_d[i+1]).$$

Using LogicFriday, we can then generate the conjunctive normal form (CNF) formula $\mathcal{C}_{Convert}$ that encodes Table 4, which is provided in Appendix A.

Lastly, the conversion from $\nabla\xi$ to δz is unnecessary due to the signed difference representation, which reduces the number of required variables.

Modeling the Determination. According to Equations (5) and (6), to satisfy the conditions $H = L'$ and $L = H'$, we obtain the following relationships:

$$\beta[(s-1) \sim 0] = \alpha[31 \sim (32-s)] + carry_0 \bmod (2^s),$$
$$\alpha[(31-s) \sim 0] = \beta[31 \sim s] + carry_1 \bmod (2^{32-s}).$$

Note that $carry_0$ depends solely on $\alpha[32-s]$ and $\beta[0]$, and $carry_1$ depends on $\alpha[0]$ and $\beta[s]$ since they are binary variables. We introduce $flag_0$ and $flag_1$ to indicate whether $carry_0$ and $carry_1$ are equal to 0, respectively, i.e.

$$(\alpha[32-s], \beta[0]) \rightarrow flag_0,$$
$$(\alpha[0], \beta[s]) \rightarrow flag_1.$$

The rules for both cases are identical and follow the same form $(a, b) \rightarrow flag$. For simplicity, they are summarized in Table 5 and can also be expressed as the CNF $\mathcal{C}_{flag}$, which is provided in Appendix A.

Table 5: The rules for $(a, b) \rightarrow flag$.

[00 → 1],	[01 → 0],	[10 → 0],	[11 → 1].

With the help of STP, Table 5 can be described more succinctly using the CVC language, where `0bin1` denotes the binary value 1:

```
ASSERT flag = IF a = b THEN 0bin1 ELSE 0bin0 ENDIF;
```

This CVC expression reveals that when the carry is 0, we set $flag = 1$, and when the carry is 1, we set $flag = 0$, i.e. $flag = carry \oplus 1$. Although this may seem counterintuitive, it offers an efficient means of reducing the number of variables required for probability calculation.

Modeling the Probability Calculation. Here, we model the probabilities for $\Pr(H = L')$ and $\Pr(L = H')$. A unified expression for probability computation has been described, as shown in Property 2.

When modeling probability, we focus on the component that most significantly influence the probability. Specifically, we consider the position of the first "1" in C (counted from left to right) when $carry = 1$, and the position of the first "0" in C (counted from left to right) when $carry = 0$.

To unify the treatment of both cases, we invert C (defined in Property 2) when $carry = 0$, denoted as D, which allows us to consistently focus on the position of the first "1" in D. However, this inversion alone is not sufficient, an additional subtraction of 1 is required to further reduce bias. This reasoning motivates the definition of $flag = carry \oplus 1$. Two illustrative examples are provided below.

Example 4. Let $n = 4$, $C = 0b1001$ and $flag = 0$. Then $\Pr(carry = 1) = 2^{-1} + 2^{-4} \approx 2^{-0.83}$. The position of the first "1" in C is 1. Thus, in our model, we approximate this by focusing only on the dominant term 2^{-1}.

Example 5. Let $n = 4$, $C = 0b1100$ and $flag = 1$. Then $\Pr(carry = 0) = 1 - 2^{-1} - 2^{-2} = 2^{-2}$. In our model, we first compute $D = \overline{C} = 0b0011$. The position of the first "1" in D is 3. As discussed above, for the case $carry = 0$, we subtract 1 (i.e., the value of the flag), yielding a final modeled probability of 2^{-2}, which matches the actual value.

In particular, the model $\mathcal{C}_{pro}^{LRS}$ can be summarized as follows:

1. Apply different strategies to C based on the value of $carry$. Specifically, if $carry = 1$, C remains unchanged, whereas if $carry = 0$, C is bitwise inverted. We denote the result by a new variable D. This behavior is encoded in the following CVC statement:

```
ASSERT D = IF flag = 0bin1 THEN ~C ELSE C ENDIF;
```

2. Determine the position of the first "1" in D, denoted by the variable pos. This can be implemented using n CVC statements. For instance, when $n = 32$:

```
ASSERT pos = IF D[31:31] = 0bin1
             THEN 0bin000001 ELSE 0bin1 = 0bin1 ENDIF;
ASSERT pos = IF D[31:31]@D[30:30] = 0bin01
             THEN 0bin000010 ELSE 0bin1 = 0bin1 ENDIF;
                          ...
ASSERT pos = IF D[31:31]@D[30:30]@...@D[0:0] = 0bin00...01
                                                  (31)
             THEN 0bin100000 ELSE 0bin1 = 0bin1 ENDIF;
```

3. Compute the corresponding probability, denoted by W, as follows:

```
ASSERT W = BVPLUS(6, pos, BVUMINUS(0bin00000@flag));
```

3.2 Modeling the Number of Bit Conditions on Boolean Functions

To represent the number of implicit conditions in the differential transitions for IFX, ONX, IFZ, and ONZ, we use a pair of binary variables (t_1, t_2), where $(t_1, t_2) \in \{(0,0),(0,1),(1,1)\}$. Specifically, $(t_1, t_2) = (0,0)$ indicates no condition, $(0,1)$ indicates one condition, and $(1,1)$ indicates two conditions. Due to the equivalences $IFX(x,y,z) = IFZ(y,z,x)$ and $ONZ(x,y,z) = ONX(z,x,y)$, we will focus on modeling $w = IFZ(x,y,z)$ here. The detailed models for IFZ and ONX are provided in Appendix A.

Modeling IFZ. First, we extend the original *fast filtering model* [15] by introducing the binary pair (t_1, t_2). For example, consider $w = IFZ(x,y,z)$. When $(\nabla x[i], \nabla y[i], \nabla z[i], \nabla w[i]) = [\texttt{==nn}]$, there are two conditions: $x[i] = 1$ and $y[i] = 0$. Consequently, we set $(t_1, t_2) = (1,1)$. There are 45 valid transitions of $(\nabla x[i], \nabla y[i], \nabla z[i], \nabla w[i], t_1, t_2)$ and are listed in Table 6.

Similarly, we can convert the 45 transition rules into 45 corresponding tuples of the form

$$(x_v[i], x_d[i], y_v[i], y_d[i], z_v[i], z_d[i], w_v[i], w_d[i], t_1, t_2).$$

Using LogicFriday, we can derive the minimized CNF representing the constraints on this tuple, denoted as $\mathcal{C}_{pro}^{IFZ}(i)$ and provided in Appendix A.

Table 6: Valid values of $(\nabla x[i], \nabla y[i], \nabla z[i], \nabla w[i], t_1, t_2)$ for IFZ.

[====,0,0],					
[==n=,0,1],	[==nn,1,1],	[==nu,1,1],	[==u=,0,1],	[==un,1,1],	[==uu,1,1],
[=n==,0,1],	[=n=n,0,1],	[=nn=,0,1],	[=nnn,0,1],	[=nu=,0,1],	[=nun,0,1],
[=u==,0,1],	[=u=u,0,1],	[=un=,0,1],	[=unu,0,1],	[=uu=,0,1],	[=uuu,0,1],
[n===,0,1],	[n==n,0,1],	[n=n=,0,1],	[n=nn,0,1],	[n=u=,0,1],	[n=un,0,1],
[nn=n,0,0],	[nnnn,0,0],	[nnun,0,0],			
[nu=n,0,1],	[nu=u,0,1],	[nun=,0,0],	[nuu=,0,0],	[u===,0,1],	[u==u,0,1],
[u=n=,0,1],	[u=nu,0,1],	[u=u=,0,1],	[u=uu,0,1],	[un=n,0,1],	[un=u,0,1],
[unn=,0,0],	[unu=,0,0],	[uu=u,0,0],	[uunu,0,0],	[uuuu,0,0].	

Modeling XOR. The case of $w = XOR(x, y, z)$ is slightly different. Since there are only two possible values for the bit conditions (0 or 1), a single binary variable t is sufficient to represent them. As in the previous approach, we construct a table of all valid transitions for $(\nabla x, \nabla y, \nabla z, \nabla w, t)$ and derive the corresponding valid tuples:

$$(x_v[i], x_d[i], y_v[i], y_d[i], z_v[i], z_d[i], w_v[i], w_d[i], t).$$

We then use LogicFriday to obtain the minimized CNF constraints for these tuples and the full expression $\mathcal{C}_{pro}^{XOR}(i)$ is provided in Appendix A.

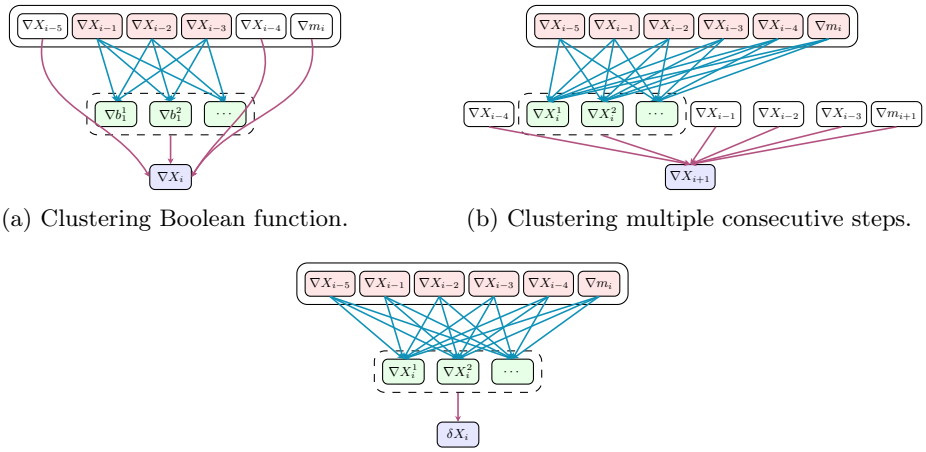
This completes the modeling of all probability-related components. Algorithm 1 in Appendix B presents the full procedure for modeling signed difference transitions and the associated probabilities in the RIPEMD-160 step function.

4 Generic Techniques for Improving Attacks

In this section, we introduce two generic techniques that enhance (SFS) collision attacks for RIPEMD-160. These techniques are independent of the search model and can be flexibly integrated with any differential characteristic.

4.1 Clustering Differentials in Last Steps

This analysis considers the differential clustering effect in the last several steps of one or both branches. A primary concern is whether the modular difference of the left branch can be offset by that of the right branch. Consequently, it is not necessary to predefine the signed differences in the last steps. Furthermore, clustering effects may also occur within individual step functions and between consecutive step functions. Once a specific signed differential trail has been established, its probability can be further enhanced by examining clustering effects from the three perspectives illustrated in Figure 3.



(a) Clustering Boolean function.

(b) Clustering multiple consecutive steps.

(c) Clustering signed differences to a modular difference.

Fig. 3: Clustering effect from three aspects.

Clustering Boolean Function. The first type of clustering arises during the probability evaluation of a single step function. In the automatic search model, only one signed difference is typically selected for b_1. In practice, however, multiple signed differences may be possible, each leading to different trails within the same step. This principle was also employed in [15], where leveraging this property doubled the probability of a 36-step right-branch trail. Specifically, the value of $Y_9[31]$ can be chosen with some flexibility, as differing outputs from the Boolean function can result in an identical output for the step function. For enhanced clarity, this scenario is further detailed in Property 3 and Example 6 below.

Property 3. For the Boolean function $w = ONX(x, y, z)$, when $(\nabla x, \nabla y, \nabla z) = [\texttt{u==}]$, the signed difference ∇w can take two possible values: $[\texttt{n}]$ and $[\texttt{u}]$. Specifically, $\nabla w = [\texttt{n}]$ when the input pair (y, z) is $(0, 0)$, $(1, 0)$, or $(1, 1)$, whereas $\nabla w = [\texttt{u}]$ if and only if $(y, z) = (0, 1)$.

Example 6. Consider the right branch of the 36-step differential characteristic presented in [15], where w denotes the output of the Boolean function at step 11. The input to this function is $(\nabla Y_{10}[31], \nabla Y_9[31], \nabla Y_8[21]) = [\texttt{u==}]$, with the condition $Y_8[21] = 1$. According to Property 3, $\nabla w[31]$ can thus take either $[\texttt{n}]$ or $[\texttt{u}]$. Moreover, since $\nabla w[31]$ is the most significant bit, both signed differences lead to an identical modular difference for w, regardless of the other bit values. This implies that the choice of signed difference for $\nabla w[31]$ does not affect the modular difference propagation through the rest of the step function. Consequently, $\nabla Y_9[31]$ can be either 0 or 1, thereby doubling the probability.

Clustering Multiple Consecutive Steps. The second type of clustering arises when considering multiple consecutive steps. The signed differences after

each step can take multiple values. As long as the signed differences in the final five steps conform to the identified differential characteristic, they ensure the cancellation of the differential between the left and right branches, ultimately resulting in a collision. This flexibility allows for numerous valid differential paths, each enabling a successful collision attack, thereby significantly increasing the overall probability.

Clustering Signed Differences to a Modular Difference. During the finalization phase of RIPEMD-160, the final hash output is computed by combining the initial value with the outputs of both the left and right branches, as shown in Equation (1). Previous attacks on RIPEMD-160 typically involved fixing the signed differences in both branches. However, for any hash function $H(CV, M)$, a (SFS) collision occurs if one can find $M \neq M'$ such that $H(CV, M) = H(CV, M')$, which is equivalent to ensuring $H(CV, M') \boxminus H(CV, M) = 0$. According to Property 1, this implies that it is unnecessary to fix the signed differences in the last steps of the right and/or left branch, as was done in our automatic search model. By considering multiple signed differential trails that yield the same zero modular difference in the final output, the overall probability can be significantly increased. This approach has been employed in previous works to enhance probabilities. For example, in the two-block collision attack on SHA-1 [23], the bit-difference positions and signs were treated as free variables in the last two steps of the first block. Additionally, Leurent et al. [7,8] applied clustering techniques to optimize the near-collision phase by canceling a given difference. Thus, this clustering strategy offers a notable advantage over relying on a single differential trail.

Example 7. Suppose $Z = X \boxplus Y^{\lll 10}$, where X, Y and Z are 32-bit variables and $\delta X = 2^{15}$, $\delta Y = -2^5$. Then, arbitrarily selecting one ∇X and one ∇Y from the following sets will result in $\delta Z = 0$.

$$\nabla X = \begin{cases} \texttt{==== ==== ==== ==== n=== ==== ==== ====} \\ \texttt{==== ==== ==== ===n u=== ==== ==== ====} \\ \texttt{==== ==== ==== ==nu u=== ==== ==== ====} \end{cases}$$

$$\nabla Y = \begin{cases} \texttt{==== ==== ==== ==== ==== ==== ==u= ====} \\ \texttt{==== ==== ==== ==== ==== ==== unn= ====} \end{cases}$$

To account for the above clustering effect, we experimentally evaluated the actual probability of collision occurrence. The corresponding experimental code is available on the previously mentioned anonymous website. Further experimental details are provided in Section 5.2.

4.2 Dedicated Message Modification Technique

With our enhanced automatic search model and the incorporation of differential clustering effects, we can identify higher-probability signed differential trails and

further enhance their probabilities to construct effective distinguishers. To launch an (SFS) collision attack based on these trails, a message modification technique is required. This process typically involves two stages: the first stage focuses on generating a sufficient number of valid starting points, while the second stage is responsible for verifying uncontrolled conditions.

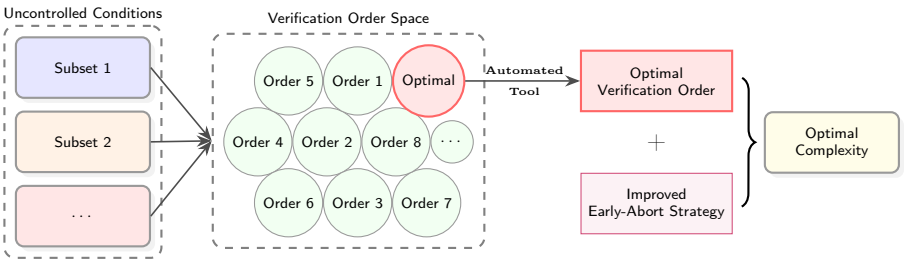


Fig. 4: Dedicated message modification technique.

Our contribution focuses on the second stage, which has the most significant impact on the overall time complexity. To reduce this cost, we propose a dedicated verification strategy. As illustrated in Figure 4, we first partition the uncontrolled conditions into several independent subsets based on different criteria, such as computation direction, branch affiliation, and others. A key principle is that the verification order within each subset is fixed, while the subsets themselves are mutually independent. This allows for flexible interleaving of conditions across subsets, where different interleaving strategies can result in significantly different time complexities. To address this, we have developed a simple automated tool that explores the interleaving space and identifies the optimal verification order.

Following this order, we implement an improved early-abort strategy: conditions are checked step by step, and the process is terminated immediately upon the first failure, thereby avoiding unnecessary computations, similar to the approach used in [23]. Moreover, this strategy is also applicable to the first stage, further improving the efficiency of the entire process. By leveraging these techniques, we effectively minimize the overall time complexity.

5 Improved SFS Collision Attacks on RIPEMD-160

With the improved techniques introduced in Section 3 and Section 4, we propose the first valid 44-step SFS collision attack on RIPEMD-160. Additionally, we present improved SFS collision attacks covering 41/42/43 steps, achieving significantly lower complexities than those reported in [9], with the 41- and 42-step attacks being practical. In the following, we detail the attack procedures.

5.1 Search for New Characteristics with Enhanced Probability

In previous attacks [9, 12], the message difference was introduced solely at the message word m_{12}, specifically $\delta m_{12} = 2^{15}$, and the conditions on the right branch largely determined the overall time complexity. Li et al. [9] provided valid SFS collision attacks for 41/42/43-step RIPEMD-160, and searched for a 44-step differential characteristic in the right branch by minimizing the objective function $\sum_{i=15}^{43} \mathbf{H}(\nabla Y_i)$. However, the resulting probability was significantly lower than 2^{-80}, rendering the 44-step SFS collision attack infeasible.

With our enhanced model, we investigated whether it is possible to enhance the current best attacks on RIPEMD-160. To assess the optimality of $\delta m_{12} = 2^{15}$, we explored this problem from two perspectives. First, we treated all 32 bits of m_{12} as free variables to search for differential paths in the right branch. Second, we further relaxed the constraints by extending the free variables to the entire 512-bit message. In both cases, the differential characteristics we obtained consistently demonstrated that $\delta m_{12} = 2^{15}$ yields better characteristics. To the best of our knowledge, increasing the number of free variables generally reduces the efficiency of the model search.

To improve the search efficiency of the model, we pre-fixed the message difference to $\delta m_{12} = 2^{15}$, which is first used to update X_{12} and Y_{15}. The corresponding high-level representation of the resulting differential characteristic under this message difference is shown in Figure 5.

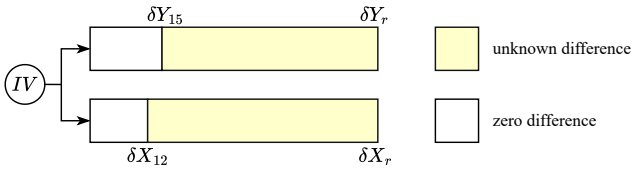


Fig. 5: Attack $r+1$ steps of RIPEMD-160 by injecting difference at m_{12}.

In the following, we provide a detailed description of how to search for the corresponding differential characteristics of $r+1$ steps of RIPEMD-160, where $40 \leq r \leq 43$:

Step 1: Use our SAT/SMT-based tool to search for an optimal solution of $(\nabla Y_{10}, \ldots, \nabla Y_r)$ such that $\delta Y_i = 0$ for $10 \leq i \leq 14$. The objective function is defined as minimizing $\sum_{i=15}^{i=r} \mathbf{H}(\nabla Y_i) + \sum_{i=15}^{i=r} \sum_{j=0}^{j=31} (t1_j^i + t2_j^i) + \sum_{i=15}^{i=r} (W_0^i + W_1^i)$, where $t_1 + t_2$ denotes the number of bit conditions on Boolean functions, as XOR is not involved in the first three rounds of the right branch. Here, W_0 and W_1 denote the probability $\Pr(H = L')$ and $\Pr(L = H')$, respectively. Note that the objective function used in [9,15] only minimizes $\sum_{i=15}^{i=r} \mathbf{H}(\nabla Y_i)$.

Step 2: Search for an optimal solution of $(\nabla X_{r-8}, \ldots, \nabla X_r)$ under a similar objective function. To enable an effective SFS collision attack on $r+1$ steps, it is required that the last five steps $(\nabla X_{r-4}, \ldots, \nabla X_r)$ cancel out the differences $(\nabla Y_{r-4}, \ldots, \nabla Y_r)$.

Step 3: Find an optimal solution of $(\nabla X_7, \ldots, \nabla X_{20})$, with $(\nabla X_7, \ldots, \nabla X_{11})$ fixed to zero differences.

Step 4: Find a valid solution of $(\nabla X_{21}, \ldots, \nabla X_{r-9})$ that bridges $(\nabla X_7, \ldots, \nabla X_{20})$ and $(\nabla X_{r-8}, \ldots, \nabla X_r)$. To improve the efficiency of the message modification phase, we explored two methods:

Method 1: Overall solution. Directly solve for all unknown differences ∇X_i for $21 \leq i \leq r-9$.

Method 2: Two-phase solution. In the first phase, find a valid solution for $(\nabla X_{21}, \ldots, \nabla X_{29})$ to ensure high probability. Then, under fixed values of $(\nabla X_i)_{21 \leq i \leq 29}$, solve for $(\nabla X_{30}, \ldots, \nabla X_{r-9})$ using an objective function that maximizes the probability.

We evaluated both methods and found that Method 2 yields differential characteristics that are more suitable for message modification.

The newly identified 44-step differential characteristic is presented in Table 7, which also includes additional conditions that impact the time complexity of the attack. Due to space limits, the corresponding 41-, 42-, and 43-step differential characteristics are provided in Table 12, Table 13, and Table 14 in Appendix C, respectively. In addition, the complete conditions on (LQ_i, RQ_i) for each case are available in Table 15 (44-step), Table 16 (41-step), Table 17 (42-step), and Table 18 (43-step) in Appendix D.

5.2 Apply Differential Clustering on New Characteristics

Upon obtaining a differential characteristic, differential clustering is performed. As depicted in Figure 2, each internal state is updated by the five preceding internal states and a message word. Accordingly, the clustering process involves the random selection of five valid consecutive internal states (i.e., X_i to X_{i+4}) along with the associated message words. The probability that the intended modular differences hold is then estimated experimentally.

On the 44-step Differential Characteristic. We evaluate the differential clustering effect from two perspectives, considering both the left and right branches:

1. First, (m_7, m_0, m_6) and 2^{24} valid tuples $(X_{37}, X_{38}, X_{39}, X_{40}, X_{41})$ that satisfy the specified conditions are randomly selected. This selection procedure is repeated 2^{10} times. Subsequently, the probability that the modular differences for (X_{42}, X_{43}) are satisfied is evaluated. The determined overall probability is $2^{-2.454}$, with the individual probabilities for δX_{42} and δX_{43} being $2^{-0.145}$ and $2^{-2.309}$, respectively.

Table 7: The 44-step differential characteristic, where $\delta m_{12} = 2^{15}$.

i	∇X_i	$\pi_l(i)$	i	∇Y_i	$\pi_r(i)$
-5	================================		-5	================================	
-4	================================		-4	================================	
-3	================================		-3	================================	
-2	================================		-2	================================	
-1	================================		-1	================================	
0	================================	0	0	================================	5
1	================================	1	1	================================	14
2	================================	2	2	================================	7
3	================================	3	3	================================	0
4	================================	4	4	================================	9
5	================================	5	5	================================	2
6	================================	6	6	================================	11
7	================================	7	7	================================	4
8	================================	8	8	================================	13
9	================================	9	9	================================	6
10	================================	10	10	================================	15
11	================================	11	11	================================	8
12	=========n======================	12	12	================================	1
13	==nu============================	13	13	====================0==========	10
14	=u========1================u=====	14	14	====================1==========	3
15	00=0==============n=====110n====	15	15	=========n======================	12
16	u0==============1=0====1=1=0====	7	16	================================	6
17	0=======1========1=====1=n=1====	4	17	==========1=====================	11
18	1======0=========0====0u=0======	13	18	==========1=====================	3
19	===u==========1=n====00=====10=	1	19	u===============================	7
20	1==0=========1===00=1=u======n==	10	20	================================	0
21	=======11======n====n=0==1===1==	6	21	1============================0	13
22	=======nu0==1==1===10=n======0==	15	22	1============================1	5
23	n====1=1011=0==1===11=0========u	3	23	====================un=========	10
24	0====0===n0=u======u==1======11n	12	24	================================	14
25	0====n===0n=0======0=11========0	0	25	====================001========	15
26	==1==0===110n======1=11=======n1	9	26	====================111========	8
27	0n1==1===0==0=========n====1==00	5	27	==========nnn===================	12
28	=un======u=01=======1=0====0===n	2	28	================================	4
29	n10======0100=n=======10=1=n=0=0	14	29	============1===================	9
30	1nu1====00n01=01=10u01011==101=n	11	30	==1=============================	1
31	0010=110n11nu0=n10un100n01=====1	8	31	==u=============================	2
32	n1=11u1uu1un1==11=00==nu0n1=100u	3	32	==1=============================	15
33	10=n10u00000u1==10=n01n1n011uu00	10	33	==1======0======================	5
34	1n=10001nn1n10=1n000u1u11un0100u	14	34	=========n=============1=======	1
35	10=11u1n100000=10=n0n=0n=0u10001	4	35	=========1=============u=======	3
36	u0=u00u01u1n=u00u=0=0==u11n1=101	9	36	=========1====1=========10======	7
37	1n=1u11=u011=0u10=u=0==0=10n10u=	15	37	========0=====0==========u=====1	14
38	u1=00u1=0=====nn=01=u0=n=00n0=1=	8	38	========u=====0==========1=====n	6
39	1====1=======0n1=u==1u=0==n1==1=	1	39	===0====1=====u==========1====1	9
40	===111=0======n==0===1====11====	2	40	===u====0=====1=10==============	11
41	===n0=00======1==============0=	7	41	===1====u=======uu===0=========0	8
42	===1==nn====================n=	0	42	========1=======11===n=========u	12
43	====================u=========n	6	43	====u==n===n====u===============	2

$X_{10}[11] = X_{11}[21], X_{11}[11] \neq X_{13}[21], X_{11}[18,19] \neq X_{12}[28,29],$
$X_{12}[18] = X_{14}[28], X_{12}[19] \neq X_{14}[29], X_{12}[20] = X_{13}[30], X_{12}[27] \neq X_{13}[5],$
$\overline{X_{39}[9]} \vee X_{40}[9] = 1, \overline{X_{39}[21]} \vee X_{40}[21] = 1,$
$Y_{16}[31] = Y_{17}[31], Y_{20}[9] = Y_{21}[9], Y_{24}[19,20] = Y_{25}[19,20], Y_{28}[29,30,31] = Y_{29}[29,30,31],$
$\overline{Y_{32}[7]} \vee Y_{33}[7] = 1, \overline{Y_{35}[0]} \vee Y_{36}[0] = 1, \overline{Y_{38}[16]} \vee Y_{39}[16] = 1,$
$\overline{Y_{39}[1]} \vee Y_{40}[1] = 1, \overline{Y_{39}[10]} \vee Y_{40}[10] = 1, \overline{Y_{40}[27]} \vee Y_{41}[27] = 1, \overline{Y_{41}[6]} \vee Y_{42}[6] = 1$

Note. The three bit conditions on X_{35} (highlighted in blue) are used to ensure that the condition on LQ_{39} is always satisfied. We further explain why setting these conditions is sufficient to guarantee this, as detailed in Appendix E.

2. A pair of $(m_{14}, m_6, m_9, m_{11}, m_8, m_{12}, m_2)$ is randomly selected such that the condition $\delta m_{12} = 2^{15}$ is met. Subsequently, 2^{38} valid tuples $(Y_{32}, Y_{33}, Y_{34}, Y_{35}, Y_{36})$ are generated. This entire process is repeated 200 times. Following these repetitions, the probability that the conditions for $(\delta Y_{39}, \delta Y_{40}, \delta Y_{41}, \delta Y_{42}, \delta Y_{43})$ are satisfied is evaluated. The resulting overall probability is $2^{-28.678}$, with the individual probabilities of $2^{-5.191}$, $2^{-4.289}$, $2^{-4.401}$, $2^{-7.844}$, and $2^{-6.953}$, respectively.

Additionally, we perform a further verification step to determine whether the left and right branches cancel each other out. Specifically, we check the following conditions:

$$\delta X_{40}^{\lll 10} \boxplus \delta Y_{39}^{\lll 10} = 0,\ \delta X_{41}^{\lll 10} \boxplus \delta Y_{40}^{\lll 10} = 0,\ \delta X_{42} \boxplus \delta Y_{41}^{\lll 10} = 0,$$

and the experimental results indicate that the corresponding probabilities are approximately $2^{-0.046}$, $2^{-0.093}$ and $2^{-0.020}$, respectively.

On the 41/42/43-step Differential Characteristics. The clustering procedures for the 41-, 42-, and 43-step differential characteristics are similar to the above process and detailed descriptions are provided in Appendix F.

Remark. We also apply a similar clustering method to the 41-, 42-, 43-, and 44-step differential characteristics presented in [9], achieving improved results in all cases. This highlights the broader applicability of our differential clustering technique.

To demonstrate the advantages of our improvements, including the improved search model and differential clustering technique, we compare the differential probabilities of the right branch in Table 8. This focus is justified because the conditions in the left branch are primarily satisfied through message modification, whereas, as noted by Li et al. [9], the time complexity of the SFS collision attack is mainly determined by the probability of the right branch. We also conducted experiments on an AMD EPYC 7302 (32 threads), using both the implementation from [9] and ours, to compare the search efficiency. Although the search time increases with our model, it remains within a reasonable range. More importantly, the quality of the differential characteristics is significantly improved.

Table 8: Comparison of the probabilities and search times of the right branch.

Steps	Single trail from [9]	Time [9]	Clustering on trail from [9]	Single trail (Ours)	Time (Ours)	Clustering on our trail
41	$2^{-57.20}$	3 min	$2^{-53.12}$	$2^{-53.76}$	0.8 h	$2^{-50.94}$
42	$2^{-67.30}$	4 min	$2^{-64.78}$	$2^{-59.59}$	1.5 h	$2^{-55.00}$
43	$2^{-71.90}$	6 min	$2^{-66.72}$	$2^{-68.06}$	4.0 h	$2^{-61.93}$
44	$2^{-83.50}$	8 min	$2^{-79.56}$	$2^{-78.06}$	9.0 h	$2^{-71.51}$

5.3 Perform Message Modification Process

In this section, we first present the message modification framework for the 41- to 44-step SFS collision attacks, incorporating our dedicated message modification technique. We then use the 44-step attack as a case study to illustrate the detailed modification process, followed by the results of the 41-, 42-, and 43-step attacks.

Message Modification Framework. The overall approach follows the two-stage framework proposed in [12], similar to the strategy in [9]. However, as the number of attack steps increases, the overall complexity becomes more complex, requiring more careful analysis. To address this, we adopt the dedicated message modification technique introduced in Section 4.2 to further reduce the overall complexity. Figure 6 shows the message modification framework, which differs from [9, Figure 3]. Specifically, we generate a sufficient number of starting points in Part 1. The remaining uncontrolled conditions are then divided into two parts based on their computation direction: Part 2 (forward computation) and Part 3 (backward computation). An automated tool is employed to determine the optimal verification order across these two parts. Moreover, a step-by-step early-abort strategy is adopted to ensure minimal overall complexity.

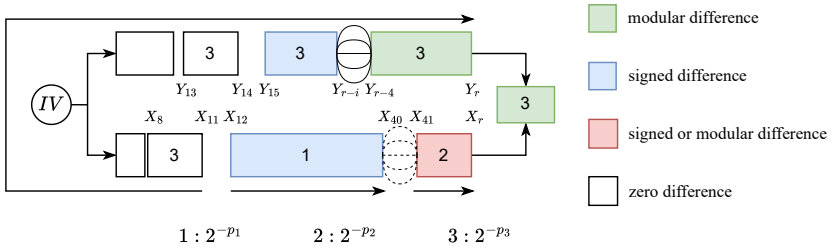


Fig. 6: Message modification framework for $(r+1)$-step SFS collision attaks, where $40 \leq r \leq 43$.

In the following, we provide a detailed description of the message modification framework.

The first stage: generate starting points. The goal of this stage is to find a sufficient number of valid solutions for $(X_{12}, \ldots, X_{40})$, referred to as starting points for the SFS collision attack. In response, Li et al. [9] proposed an effective two-phase strategy:

Phase 1: Find a solution for $(X_{24}, \ldots, X_{38})$ by using the model that describes value transitions [13, 19] covering this part, since conditions on these states are typically dense. Next, satisfy the conditions on $(X_{15}, \ldots, X_{23})$ by exploiting the available degrees of freedom in $(m_0, m_2, m_5, m_6, m_{12})$. Since these conditions are sparse, this step is

expected to generate many solutions of $(X_{15}, \ldots, X_{38})$. Finally, use (m_1, m_{13}) to determine $(X_{12}, X_{13}, X_{14}, X_{39}, X_{40})$, resulting in a complete solution for $(X_{12}, \ldots, X_{40})$.

Phase 2: Once a valid solution for $(X_{12}, \ldots, X_{40})$ is obtained, referred to as an initial starting point, additional starting points can be efficiently generated by utilizing the degrees of freedom in (m_1, m_{13}). First, one can randomly choose values of (X_{13}, X_{14}) and then recompute X_{12} by using $(X_i)_{13 \le i \le 17}$ and m_4, where $(X_{15}, X_{16}, X_{17}, m_4)$ remain fixed from the initial starting point:

$$X_{12} = ((X_{17} \boxminus X_{13}^{\lll 10}) \ggg s_{17}^l \boxminus IFX(X_{16}, X_{15}, X_{14}^{\lll 10}) \boxminus m_4 \boxminus K_1^l) \ggg 10.$$

The values of $(X_{12}, LQ_{16}, LQ_{17}, LQ_{18})$ are then checked to meet the required conditions. If not, another pair (X_{13}, X_{14}) is selected, and the procedure is repeated. If the conditions hold, the values of m_{13} and m_1 are modified to ensure X_{18} and X_{19} remain unchanged so that $(X_i)_{20 \le i \le 38}$ can stay the same as the initial starting point.

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

Under the updated m_1, recompute (X_{39}, X_{40}) and enforce the corresponding conditions on them. Specifically, for $r > 40$, the required conditions include $(X_{39}, X_{40}, LQ_{39}, LQ_{40})$, while for $r = 40$, checking X_{40} is no longer necessary. If the conditions are not met, the process is repeated with a new (X_{13}, X_{14}) pairs. For convenience, let 2^{-p_1} denote the probability of the following conditions:

$$(X_{12}, LQ_{16}, LQ_{17}, LQ_{18}, X_{39}, X_{40}, LQ_{39}, LQ_{40})$$

when $r > 40$, excluding X_{40} when $r = 40$.

By counting the conditions on (X_{21}, X_{22}, X_{23}), we can generate a sufficiently large number of solutions for $(X_{12}, \ldots, X_{40})$ during Phase 1, typically far exceeding 2^{32}. These solutions serve as initial starting points for Phase 2, in which additional starting points are produced.

As for time complexity, generating initial starting points in Phase 1 is efficient and negligible in practice [9]. Indeed, we can generate one initial starting point within 30 seconds. The corresponding solutions for the 43- and 44-step SFS collision attacks are summarized in Table 21 in Appendix G.

In Phase 2, each new starting point requires five steps of the step function, with specific conditions verified at each step. Rather than executing all five steps unconditionally and verifying the conditions only at the end [9], efficiency can be improved by adopting a step-by-step early-abort strategy. This approach avoids redundant computation and improves overall efficiency.

The second stage: validate the uncontrolled conditions. In the first stage, all message words are determined except for m_7. The remaining degree of freedom offered by m_7 is then used to satisfy the remaining differential conditions. First, we exhaust all possible values of X_{11} and compute the corresponding m_7 as follows:

$$m_7 = (X_{16} \boxminus X_{12}^{\lll 10})^{\ggg s_{16}^l} \boxminus IFX(X_{15}, X_{14}, X_{13}^{\lll 10}) \boxminus X_{11}^{\lll 10} \boxminus K_0^l.$$

If there are n_1 bit conditions on X_{11}, each starting point yields 2^{32-n_1} possible values of X_{11}. Once this step is completed, all message words are fixed. The remaining uncontrolled conditions include

$$(X_{41}, \ldots, X_{r-1}, LQ_{41}, \ldots, LQ_r, X_{10}, LQ_{15}, LQ_{14}, LQ_{13}, LQ_{12})$$

as well as all conditions in the right branch. All of these conditions hold with probability $2^{-(p_2+p_3)}$, and verifying them requires executing $2r-22$ steps of the step function. In previous work [9], these conditions were divided into two sets according to different branches and verified in two stages: the conditions on the right branch were checked only if all conditions on the left branch were satisfied.

However, we found that this approach is not optimal, as it does not fully exploit the impact of the verification order on time complexity. Moreover, the two-stage early-abort strategy introduces redundant computations. To overcome these limitations, we adopt the dedicated message modification technique described in Section 4.2. We first divide the uncontrolled conditions into two independent subsets based on their computation direction: the forward computation(Part 2) involves $(X_{41}, \ldots, X_{r-1}, LQ_{41}, \ldots, LQ_r)$, while the remaining conditions belong to the backward computation(Part 3). To find the most efficient order, we use an automated tool to explore all possible interleavings and select the one that yields the lowest complexity. Combined with the step-by-step early-abort technique, this approach achieves the optimal time complexity. Next, we present the first 44-step attack as a case study to demonstrate the complete message modification process in detail.

First Attack on 44-step RIPEMD-160. According to Table 7, Table 15 and the above analysis, we have

$$n_1 = 3,\ p_1 = 27.168,\ p_2 = 8.805,\ p_3 = 73.365.$$

Therefore, in the first stage, a total of $2^{(p_2+p_3)-(32-n_1)} = 2^{53.170}$ starting points are required. Each starting point must satisfy a set of conditions with probability $2^{-p_1} = 2^{-27.168}$. The detailed procedure for generating a valid starting point, along with the corresponding complexity, is presented in Table 9.

Once a sufficient number of starting points have been generated, the second stage verifies the remaining uncontrolled conditions, which hold with an overall probability of $2^{-p_2-p_3} = 2^{-82.170}$. In this attack, the forward computation (Part 2) includes $(X_{41}, LQ_{41}, \delta X_{42}, \delta X_{43})$, while the remaining uncontrolled conditions along with the conditions ensuring cancellation between the two branches

Table 9: Detailed computation of complexity for the first stage.

The probability to be verified is $2^{-27.168}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	X_{12}	$X_{12}[21] = 0$ $X_{12}[18] = X_{14}[28]$ $X_{12}[19] \neq X_{14}[29]$ $X_{12}[20] = X_{13}[30]$ $X_{12}[27] \neq X_{13}[5]$	2^{-5}	$2^{-22.168}$	$\frac{1}{2\times 44} \times 2^{27.618}$
2	m_{13}, m_1 X_{39}, X_{40}	$X_{39}[5, 8, 17, 18] = 0$ $X_{40}[14, 17, 24] = 0$ $X_{39}[1, 4, 10, 11, 14, 16, 26, 31] = 1$ $X_{40}[4, 5, 10, 26, 27, 28] = 1$ $\overline{X_{39}[9]} \vee X_{40}[9] = 1$ $\overline{X_{39}[21]} \vee X_{40}[21] = 1$ $LQ_{40}(pro \approx 2^{-0.338})$	$2^{-22.168}$	-	$\frac{4}{2\times 44} \times 2^{22.168}$
Total	-	-	-	-	$2^{20.878}$
The total time complexity of the first stage is $2^{53.170} \times 2^{20.878} \approx 2^{74.05}$					

† Time complexity is evaluated by one 44-step compression of RIPEMD-160.

are grouped into the backward computation (Part 3). To optimize this process, we employ an automated tool to determine the optimal verification order. The detailed order and complexity of this stage are provided in Table 10.

Consequently, the whole time complexity of the 44-step SFS collision attack on RIPEMD-160 is approximately $2^{74.05} + 2^{76.69} \approx 2^{76.9}$.

Attacks on 41/42/43-step RIPEMD-160. The detailed attack procedures for 41-step, 42-step, and 43-step RIPEMD-160 are provided in Table 22, Table 23, and Table 24 in Appendix H, respectively. The corresponding time complexities are $2^{49.8}$, $2^{53.9}$, and $2^{62.3}$.

Notably, this work presents the first known instance of successfully generating colliding message pairs for 41-step and 42-step RIPEMD-160 within practical time, as shown in Table 11. More colliding message pairs are provided in Appendix I.

Remark. We also apply each of the three techniques individually to the baseline results from [9], achieving significant improvements in the time complexities of SFS collision attacks on RIPEMD-160. For the 41-step attack, the three improvements contribute roughly equally to the overall complexity reduction. For the 42-step attack, their respective contributions are approximately 41%, 35%, and 24%. For the 43-step attack, the contributions are 37%, 38%, and 25%, and for the 44-step attack, they are 37%, 35%, and 28%, respectively. These results indicate that all three techniques play significant roles and contribute in a relatively balanced manner. Notably, as shown in Table 11, we successfully generated colliding message pairs using the 41-step characteristic from [9] for the first time. A detailed comparison is presented in Figure 7.

While the improvements on the original trails [9] are already significant, they remain inferior to the results obtained using our newly identified differential characteristics. In addition, even after applying differential clustering and

Table 10: Detailed computation of complexity for the second stage.

The probability to be verified is $2^{-82.170}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	m_7	LQ_{15}	$2^{-0.093}$	$2^{-82.077}$	$\frac{1}{2\times44}\times 2^{82.170}$
2	X_{41}	$X_{41}[17]=1, LQ_{41}$ $X_{41}[1,24,25,27,28]=0$	$2^{-6.351}$	$2^{-75.726}$	$\frac{1}{2\times44}\times 2^{82.077}$
3	X_{10}	LQ_{14} $X_{10}[11]=X_{11}[21]$	$2^{-1.508}$	$2^{-74.218}$	$\frac{1}{2\times44}\times 2^{75.726}$
4	X_{42}	δX_{42}	$2^{-0.145}$	$2^{-74.073}$	$\frac{1}{2\times44}\times 2^{74.218}$
5	X_{43}	δX_{43}	$2^{-2.309}$	$2^{-71.764}$	$\frac{1}{2\times44}\times 2^{74.073}$
6	X_9	LQ_{13}	$2^{-0.093}$	$2^{-71.671}$	$\frac{1}{2\times44}\times 2^{71.764}$
7	X_8	LQ_{12}	$2^{-0.001}$	$2^{-71.670}$	$\frac{1}{2\times44}\times 2^{71.671}$
8	Y_{13}	$Y_{13}[11]=0$	2^{-1}	$2^{-70.670}$	$\frac{27}{2\times44}\times 2^{71.670}$
9	Y_{14}	$Y_{14}[11]=1$	2^{-1}	$2^{-69.670}$	$\frac{1}{2\times44}\times 2^{70.670}$
10	Y_{15}	RQ_{15} $Y_{15}[21]=0$	$2^{-1.001}$	$2^{-68.669}$	$\frac{1}{2\times44}\times 2^{69.670}$
11	Y_{16}, Y_{17}	$Y_{17}[21]=1$ $Y_{16}[31]=Y_{17}[31]$	2^{-2}	$2^{-66.669}$	$\frac{2}{2\times44}\times 2^{68.669}$
12	Y_{18}	$Y_{18}[21]=1$	2^{-1}	$2^{-65.669}$	$\frac{1}{2\times44}\times 2^{66.669}$
13	Y_{19}	$Y_{19}[31]=1$	2^{-1}	$2^{-64.669}$	$\frac{1}{2\times44}\times 2^{65.669}$
14	Y_{20}, Y_{21}	$Y_{21}[0]=0$ $Y_{21}[31]=1$ $Y_{20}[9]=Y_{21}[9]$	2^{-3}	$2^{-61.669}$	$\frac{2}{2\times44}\times 2^{64.669}$
15	Y_{22}	$Y_{22}[0,31]=1$	2^{-2}	$2^{-59.669}$	$\frac{1}{2\times44}\times 2^{61.669}$
16	Y_{23}	$Y_{23}[9]=0$ $Y_{23}[10]=1$	2^{-2}	$2^{-57.669}$	$\frac{1}{2\times44}\times 2^{59.669}$
17	Y_{24}, Y_{25}	$Y_{25}[9]=1$ $Y_{25}[10,11]=0$ $Y_{24}[19,20]=Y_{25}[19,20]$	2^{-5}	$2^{-52.669}$	$\frac{2}{2\times44}\times 2^{57.669}$
18	Y_{26}	$Y_{26}[9,10,11]=1$	2^{-3}	$2^{-49.669}$	$\frac{1}{2\times44}\times 2^{52.669}$
19	Y_{27}	RQ_{27} $Y_{27}[19,20,21]=0$	$2^{-3.001}$	$2^{-46.668}$	$\frac{1}{2\times44}\times 2^{49.669}$
20	Y_{28}, Y_{29}	$Y_{29}[19]=1$ $Y_{28}[31\sim29]=Y_{29}[31\sim29]$	2^{-4}	$2^{-42.668}$	$\frac{2}{2\times44}\times 2^{46.668}$
21	Y_{30}	$Y_{30}[29]=1$	2^{-1}	$2^{-41.668}$	$\frac{1}{2\times44}\times 2^{42.668}$
22	Y_{31}	$Y_{31}[29]=1$	2^{-1}	$2^{-40.668}$	$\frac{1}{2\times44}\times 2^{41.668}$
23	Y_{32}, Y_{33}	$Y_{32}[29]=1$ $Y_{33}[22]=0$ $Y_{33}[29]=1$ $\overline{Y_{32}[7]}\vee Y_{33}[7]=1$	$2^{-3.415}$	$2^{-37.253}$	$\frac{2}{1\times44}\times 2^{40.668}$
24	Y_{34}	RQ_{34} $Y_{34}[7]=1$ $Y_{34}[22]=0$	$2^{-2.001}$	$2^{-35.252}$	$\frac{1}{2\times44}\times 2^{37.253}$
25	Y_{35}, Y_{36}	$Y_{36}[6]=0$ $Y_{35}[7,22]=1$ $Y_{36}[7,17,22]=1$ $\overline{Y_{35}[0]}\vee Y_{36}[0]=1$	$2^{-6.415}$	$2^{-28.837}$	$\frac{2}{1\times44}\times 2^{35.252}$
26	Y_{39}	δY_{39} $\delta X_{40}^{\lll 10}\boxplus\delta Y_{39}^{\lll 10}=0$	$2^{-5.237}$	$2^{-23.6}$	$\frac{3}{1\times44}\times 2^{28.837}$
27	Y_{40}	δY_{40} $\delta X_{41}^{\lll 10}\boxplus\delta Y_{40}^{\lll 10}=0$	$2^{-4.382}$	$2^{-19.218}$	$\frac{1}{2\times44}\times 2^{23.6}$
28	Y_{41}	δY_{41} $\delta X_{42}\boxplus\delta Y_{41}^{\lll 10}=0$	$2^{-4.421}$	$2^{-14.797}$	$\frac{1}{2\times44}\times 2^{19.218}$
29	Y_{42}	δY_{42}	$2^{-7.844}$	$2^{-6.953}$	$\frac{1}{2\times44}\times 2^{14.797}$
30	Y_{43}	δY_{43}	$2^{-6.953}$	-	$\frac{1}{2\times44}\times 2^{6.953}$
Total	-	-	-	-	$2^{76.69}$
The total time complexity of the second stage is $2^{76.69}$					

† Time complexity is evaluated by one 44-step compression of RIPEMD-160.

Table 11: Colliding message pairs (M_0, M_1) and CV_0.

41-step RIPEMD-160 using our characteristic in Table 12	
CV_0	e6e10080 2e684ca7 d12e322a 96b78c44 d21402b3
M_0	22fd841e 09f6d5ae f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 f4fca809 c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d108c 37a95c6f 970b8ed5 4ca0cc22
M_1	22fd841e 09f6d5ae f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 f4fca809 c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d908c 37a95c6f 970b8ed5 4ca0cc22
hash	d2a97aae 76560c6b b40b2ad6 e2cbbf41 3408ce8a
42-step RIPEMD-160 using our characteristic in Table 13	
CV_0	1c591735 50c18163 1e1a223a 8e8f3abb 7c7caf6b
M_0	79f928a9 5f84cccd 8353ba65 e2e8af98 06a68efd 4fe043f5 efccef7e eb18c4c7 c1703e5e 64ee7ec9 ae7ad7b5 078d7695 6ceb3df9 5ff84205 5001508f eaff152f
M_1	79f928a9 5f84cccd 8353ba65 e2e8af98 06a68efd 4fe043f5 efccef7e eb18c4c7 c1703e5e 64ee7ec9 ae7ad7b5 078d7695 6cebbdf9 5ff84205 5001508f eaff152f
hash	e6b6d1db 130cf3c6 e8b52167 16eafae0 137e29ce
41-step RIPEMD-160 using characteristic from [9]	
CV_0	16ab6628 8330e8d0 19a38e51 97ea5ecc 65dc063c
M_0	c408cf53 27f76690 d523de93 8ba64003 e5b58a85 b99d6243 54e6af34 d138090b 27c09f39 e8300da6 a4fff494 3efb77d4 04adaef9 ab38ffe5 8ae1c0fa bc002a65
M_1	c408cf53 27f76690 d523de93 8ba64003 e5b58a85 b99d6243 54e6af34 d138090b 27c09f39 e8300da6 a4fff494 3efb77d4 04ae2ef9 ab38ffe5 8ae1c0fa bc002a65
hash	cf4518fa 4a478743 4a7cf2ab bd19a609 6f8c5f4f

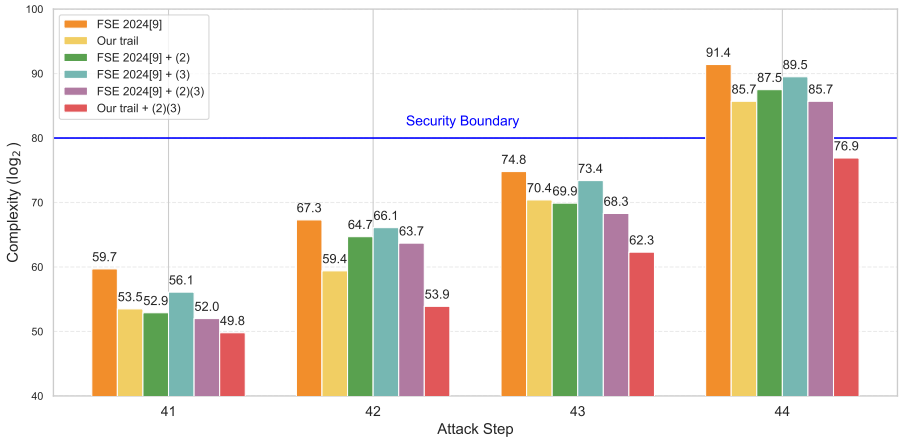


Fig. 7: Comparison of attack complexities using various techniques on steps 41–44 of RIPEMD-160. Here, (2) and (3) denote the differential clustering and the dedicated message modification technique, respectively.

message modification techniques to the 44-step characteristic from [9], the time complexity is still higher than 2^{80}, making a successful attack infeasible.

Discussion: Attacks from the Middle. We also apply our techniques to attacks from the middle. The best known attack of this type is the 48-step SFS collision attack by Wang et al. [21], which uses the differential characteristic proposed in [17] with a message difference of $\delta m_7 = 2^{31}$. The key idea is that certain bits of the chaining variable at step i can be derived by adding a few carefully selected conditions in advance, thereby increasing the uncontrolled probability.

To improve this result, we treat all 32 bits of m_7 as free variables and use our improved model to search for a better trail, where $\delta m_7 = 2^{16}$. Based on this new trail, we can improve the overall attack complexity to $2^{74.6}$ by applying our differential clustering and dedicated message modification techniques. Notably, this result is achieved entirely using the techniques developed in this paper, it is still possible to improve the complexity with additional manual optimization.

Mendel et al. [17] also provided a rough estimation of the maximum number of steps that could potentially be attacked under different message difference injection strategies, with some cases exceeding 48 steps, including m_7. However, the main bottleneck in achieving longer-step results lies in the lack of more advanced methods to merge the two branches, which requires exploiting structural properties such as message permutations. Nevertheless, this challenge is beyond the primary scope of our work.

The full details of our 48-step attack, along with an explanation of why it cannot be extended to more steps, are provided in Appendix J.

6 Conclusion

This paper presents substantial improvements to SFS collision attacks on RIPEMD-160. We developed an enhanced automatic search model, incorporating a holistic objective function that considers probabilities derived from Boolean functions and rotation operations, moving beyond simple Hamming weight. This allows for the identification of optimal differential characteristics with maximized probabilities for uncontrolled parts. Furthermore, we introduced two generic techniques: differential clustering, which exploits clustering effects from three perspectives, and a dedicated message modification strategy to minimize time complexity. As a result, we present the first successful SFS collision attack on RIPEMD-160 reaching 44 steps. Moreover, we significantly reduced the time complexities of existing 41-, 42-, and 43-step SFS attacks, enabling us to find practical colliding message pairs for the 41- and 42-step variants for the first time. This achievement marks a break of more than half of the full-round RIPEMD-160, two decades after its initial publication. We also demonstrate the applicability of our methods by extending them to attacks from the middle, achieving improved outcomes.

Acknowledgments. We sincerely thank the anonymous reviewers for providing valuable comments to help us improve the overall quality of the paper. This

work is supported by the National Key R&D Program of China (Grant No. 2024YFA1013000, 2023YFA1009500), the National Natural Science Foundation of China (Grant No. 62032014, U2336207, 62372274), the National Cryptologic Science Fund of China (2025NCSF01013) and the Natural Science Foundation of Shandong Province, China (Grant No.ZR2024QF218). In addition, Haoyang Wang is supported by the National Natural Science Foundation of China (Grant No. 62302293). Generative AI tools (OpenAI's ChatGPT) were used to assist with spelling and grammatical corrections. The authors remain fully responsible for the accuracy and integrity of the final manuscript.

References

1. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 36–57. Springer (2005). https://doi.org/10.1007/11426639_3, https://doi.org/10.1007/11426639_3
2. Bosselaers, A., Preneel, B. (eds.): Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evaluation RIPE-RACE 1040, Lecture Notes in Computer Science, vol. 1007. Springer (1995). https://doi.org/10.1007/3-540-60640-8, https://doi.org/10.1007/3-540-60640-8
3. Cannière, C.D., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4284, pp. 1–20. Springer (2006). https://doi.org/10.1007/11935230_1, https://doi.org/10.1007/11935230_1
4. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of RIPEMD. In: Gollmann, D. (ed.) Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1039, pp. 71–82. Springer (1996). https://doi.org/10.1007/3-540-60865-6_44, https://doi.org/10.1007/3-540-60865-6_44
5. Fouque, P., Leurent, G., Nguyen, P.Q.: Automatic search of differential path in MD4. IACR Cryptol. ePrint Arch. p. 206 (2007), http://eprint.iacr.org/2007/206
6. Landelle, F., Peyrin, T.: Cryptanalysis of full RIPEMD-128. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 228–244. Springer (2013). https://doi.org/10.1007/978-3-642-38348-9_14, https://doi.org/10.1007/978-3-642-38348-9_14
7. Leurent, G., Peyrin, T.: From collisions to chosen-prefix collisions application to full SHA-1. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11478, pp. 527–555. Springer (2019). https://doi.org/10.1007/978-3-030-17659-4_18, https://doi.org/10.1007/978-3-030-17659-4_18
8. Leurent, G., Peyrin, T.: SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. In: Capkun, S., Roesner, F. (eds.) 29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020. pp. 1839–1856. USENIX Association (2020), https://www.usenix.org/conference/usenixsecurity20/presentation/leurent
9. Li, Y., Liu, F., Wang, G.: Automating collision attacks on RIPEMD-160. IACR Trans. Symmetric Cryptol. **2023**(4), 112–142 (2023). https://doi.org/10.46586/TOSC.V2023.I4.112-142, https://doi.org/10.46586/tosc.v2023.i4.112-142
10. Li, Y., Liu, F., Wang, G.: New records in collision attacks on SHA-2. In: Joye, M., Leander, G. (eds.) Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual

International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14651, pp. 158–186. Springer (2024). https://doi.org/10.1007/978-3-031-58716-0_6, https://doi.org/10.1007/978-3-031-58716-0_6

11. Liu, F., Dobraunig, C., Mendel, F., Isobe, T., Wang, G., Cao, Z.: Efficient collision attack frameworks for RIPEMD-160. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. Lecture Notes in Computer Science, vol. 11693, pp. 117–149. Springer (2019). https://doi.org/10.1007/978-3-030-26951-7_5, https://doi.org/10.1007/978-3-030-26951-7_5
12. Liu, F., Dobraunig, C., Mendel, F., Isobe, T., Wang, G., Cao, Z.: New semi-free-start collision attack framework for reduced RIPEMD-160. IACR Trans. Symmetric Cryptol. **2019**(3), 169–192 (2019). https://doi.org/10.13154/TOSC.V2019.I3.169-192, https://doi.org/10.13154/tosc.v2019.i3.169-192
13. Liu, F., Isobe, T., Meier, W.: Automatic verification of differential characteristics: Application to reduced gimli. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12172, pp. 219–248. Springer (2020). https://doi.org/10.1007/978-3-030-56877-1_8, https://doi.org/10.1007/978-3-030-56877-1_8
14. Liu, F., Mendel, F., Wang, G.: Collisions and semi-free-start collisions for round-reduced RIPEMD-160. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 158–186. Springer (2017). https://doi.org/10.1007/978-3-319-70694-8_6, https://doi.org/10.1007/978-3-319-70694-8_6
15. Liu, F., Wang, G., Sarkar, S., Anand, R., Meier, W., Li, Y., Isobe, T.: Analysis of RIPEMD-160: new collision attacks and finding characteristics with MILP. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV. Lecture Notes in Computer Science, vol. 14007, pp. 189–219. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_7, https://doi.org/10.1007/978-3-031-30634-1_7
16. Mendel, F., Nad, T., Scherz, S., Schläffer, M.: Differential attacks on reduced RIPEMD-160. In: Gollmann, D., Freiling, F.C. (eds.) Information Security - 15th International Conference, ISC 2012, Passau, Germany, September 19-21, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7483, pp. 23–38. Springer (2012). https://doi.org/10.1007/978-3-642-33383-5_2, https://doi.org/10.1007/978-3-642-33383-5_2
17. Mendel, F., Peyrin, T., Schläffer, M., Wang, L., Wu, S.: Improved cryptanalysis of reduced RIPEMD-160. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8270, pp. 484–503. Springer (2013). https://doi.org/10.1007/978-3-642-42045-0_25, https://doi.org/10.1007/978-3-642-42045-0_25

18. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the collision resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4176, pp. 101–116. Springer (2006). https://doi.org/10.1007/11836810_8, https://doi.org/10.1007/11836810_8
19. Mironov, I., Zhang, L.: Applications of SAT solvers to cryptanalysis of hash functions. In: Biere, A., Gomes, C.P. (eds.) Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4121, pp. 102–115. Springer (2006). https://doi.org/10.1007/11814948_13, https://doi.org/10.1007/11814948_13
20. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 570–596. Springer (2017). https://doi.org/10.1007/978-3-319-63688-7_19, https://doi.org/10.1007/978-3-319-63688-7_19
21. Wang, G., Shen, Y., Liu, F.: Cryptanalysis of 48-step RIPEMD-160. IACR Trans. Symmetric Cryptol. **2017**(2), 177–202 (2017). https://doi.org/10.13154/TOSC.V2017.I2.177-202, https://doi.org/10.13154/tosc.v2017.i2.177-202
22. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 1–18. Springer (2005). https://doi.org/10.1007/11426639_1, https://doi.org/10.1007/11426639_1
23. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer (2005). https://doi.org/10.1007/11535218_2, https://doi.org/10.1007/11535218_2
24. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 19–35. Springer (2005). https://doi.org/10.1007/11426639_2, https://doi.org/10.1007/11426639_2
25. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 1–16. Springer (2005). https://doi.org/10.1007/11535218_1, https://doi.org/10.1007/11535218_1

A New CNFs

In this section, we present new CNF formulas utilized for modeling probabilities, as described in Section 3.1 and 3.2.

$$\mathcal{C}_{Convert}(i): \begin{cases} \overline{\xi_v[i]} = 1 \\ \overline{c_v[i]} \vee c_d[i] = 1 \\ z_d[i] \vee c_d[i] \vee \overline{\xi_d[i]} = 1 \\ z_d[i] \vee \overline{c_d[i]} \vee \xi_d[i] = 1 \\ \overline{z_v[i]} \vee \overline{c_d[i]} \vee \overline{\xi_d[i]} = 1 \\ \overline{z_d[i]} \vee c_d[i] \vee \xi_d[i] = 1 \\ c_v[i] \vee \xi_d[i] \vee \overline{c_v[i+1]} = 1 \\ z_v[i] \vee c_v[i] \vee \overline{c_v[i+1]} = 1 \\ \overline{z_v[i]} \vee c_d[i] \vee c_d[i+1] = 1 \\ \overline{c_v[i]} \vee \overline{\xi_d[i]} \vee c_d[i+1] = 1 \\ \overline{z_v[i]} \vee \overline{c_v[i]} \vee c_d[i+1] = 1 \\ z_v[i] \vee \xi_d[i] \vee \overline{c_v[i+1]} = 1 \\ c_d[i] \vee c_v[i+1] \vee \overline{c_d[i+1]} = 1 \\ \overline{\xi_d[i]} \vee c_v[i+1] \vee \overline{c_d[i+1]} = 1 \\ \overline{z_v[i]} \vee c_v[i+1] \vee \overline{c_d[i+1]} = 1 \\ z_v[i] \vee \overline{z_d[i]} \vee \overline{c_v[i]} \vee \overline{c_d[i+1]} = 1 \\ z_v[i] \vee \overline{z_d[i]} \vee c_v[i] \vee \overline{c_d[i]} \vee c_d[i+1] = 1 \end{cases}$$

$$\mathcal{C}_{flag}(i): \begin{cases} a[i] \vee \overline{b[i]} \vee \overline{flag[i]} = 1 \\ \overline{a[i]} \vee b[i] \vee \overline{flag[i]} = 1 \\ \overline{a[i]} \vee \overline{b[i]} \vee flag[i] = 1 \\ a[i] \vee b[i] \vee flag[i] = 1 \end{cases}$$

$$\mathcal{C}_{pro}^{IFZ}(i): \begin{cases} \overline{\overline{y_d[i]} \vee \overline{t_1[i]}} = 1 \\ \overline{\overline{x_d[i]} \vee \overline{t_1[i]}} = 1 \\ \overline{\overline{z_v[i]} \vee z_d[i]} = 1 \\ \overline{\overline{y_v[i]} \vee y_d[i]} = 1 \\ w_d[i] \vee \overline{t_1[i]} = 1 \\ \overline{\overline{x_v[i]} \vee x_d[i]} = 1 \\ \overline{\overline{w_v[i]} \vee w_d[i]} = 1 \\ \overline{\overline{x_d[i]} \vee y_d[i] \vee t_2[i]} = 1 \\ \overline{x_d[i] \vee \overline{y_d[i]} \vee t_2[i]} = 1 \\ \overline{\overline{x_v[i]} \vee \overline{y_v[i]} \vee \overline{t_2[i]}} = 1 \\ \overline{y_d[i] \vee \overline{w_d[i]} \vee t_2[i]} = 1 \\ \overline{\overline{x_v[i]} \vee \overline{y_v[i]} \vee w_v[i]} = 1 \\ \overline{x_d[i] \vee y_d[i] \vee z_d[i] \vee \overline{t_2[i]}} = 1 \\ \overline{\overline{x_d[i]} \vee \overline{y_d[i]} \vee \overline{z_d[i]} \vee \overline{t_2[i]}} = 1 \\ \overline{x_d[i] \vee y_d[i] \vee \overline{w_d[i]} \vee t_1[i]} = 1 \\ \overline{x_v[i] \vee y_v[i] \vee \overline{w_v[i]} \vee t_1[i]} = 1 \\ \overline{\overline{x_v[i]} \vee y_v[i] \vee \overline{w_d[i]} \vee t_2[i]} = 1 \\ \overline{x_v[i] \vee \overline{y_v[i]} \vee \overline{w_d[i]} \vee t_2[i]} = 1 \\ \overline{\overline{x_d[i]} \vee \overline{y_d[i]} \vee z_d[i] \vee w_d[i]} = 1 \\ \overline{\overline{x_v[i]} \vee y_d[i] \vee w_v[i] \vee \overline{w_d[i]}} = 1 \\ \overline{x_d[i] \vee \overline{y_v[i]} \vee w_v[i] \vee \overline{w_d[i]}} = 1 \\ \overline{x_v[i] \vee \overline{x_d[i]} \vee y_v[i] \vee \overline{y_d[i]} \vee \overline{t_2[i]}} = 1 \\ \overline{x_v[i] \vee y_v[i] \vee \overline{z_d[i]} \vee w_d[i] \vee t_2[i]} = 1 \end{cases}$$

$$
\mathcal{C}_{pro}^{ONX}(i):\begin{cases}
\overline{t_1[i]} \vee t_2[i] = 1 \\
\overline{z_v[i]} \vee z_d[i] = 1 \\
\overline{y_v[i]} \vee y_d[i] = 1 \\
w_d[i] \vee \overline{t_1[i]} = 1 \\
\overline{x_v[i]} \vee x_d[i] = 1 \\
\overline{w_v[i]} \vee w_d[i] = 1 \\
\overline{y_d[i]} \vee \overline{z_d[i]} \vee \overline{t_1[i]} = 1 \\
y_d[i] \vee \overline{z_d[i]} \vee t_2[i] = 1 \\
\overline{y_d[i]} \vee z_d[i] \vee t_2[i] = 1 \\
\overline{x_d[i]} \vee \overline{z_d[i]} \vee \overline{t_1[i]} = 1 \\
\overline{x_d[i]} \vee \overline{y_d[i]} \vee \overline{t_1[i]} = 1 \\
\overline{x_v[i]} \vee w_v[i] \vee \overline{t_1[i]} = 1 \\
\overline{x_v[i]} \vee \overline{w_v[i]} \vee t_1[i] = 1 \\
y_d[i] \vee z_d[i] \vee t_1[i] \vee \overline{t_2[i]} = 1 \\
x_d[i] \vee y_d[i] \vee z_d[i] \vee \overline{t_2[i]} = 1 \\
\overline{x_d[i]} \vee \overline{y_d[i]} \vee \overline{z_d[i]} \vee \overline{t_2[i]} = 1 \\
x_d[i] \vee y_v[i] \vee \overline{z_v[i]} \vee t_2[i] = 1 \\
x_d[i] \vee \overline{y_v[i]} \vee z_v[i] \vee t_2[i] = 1 \\
y_v[i] \vee \overline{z_v[i]} \vee \overline{w_d[i]} \vee t_2[i] = 1 \\
\overline{y_v[i]} \vee z_v[i] \vee \overline{w_d[i]} \vee t_2[i] = 1 \\
\overline{y_d[i]} \vee \overline{z_d[i]} \vee w_d[i] \vee \overline{t_2[i]} = 1 \\
x_d[i] \vee y_d[i] \vee \overline{w_d[i]} \vee t_1[i] = 1 \\
x_d[i] \vee z_d[i] \vee \overline{w_d[i]} \vee t_1[i] = 1 \\
x_v[i] \vee \overline{x_d[i]} \vee \overline{w_v[i]} \vee \overline{t_1[i]} = 1 \\
\overline{x_d[i]} \vee \overline{y_v[i]} \vee \overline{z_v[i]} \vee w_d[i] = 1 \\
x_d[i] \vee \overline{y_v[i]} \vee \overline{z_v[i]} \vee \overline{w_d[i]} = 1 \\
x_d[i] \vee y_v[i] \vee z_v[i] \vee \overline{w_d[i]} \vee t_1[i] = 1 \\
\overline{x_d[i]} \vee y_v[i] \vee z_v[i] \vee w_d[i] \vee t_2[i] = 1 \\
x_v[i] \vee \overline{x_d[i]} \vee w_v[i] \vee \overline{w_d[i]} \vee t_1[i] = 1
\end{cases}
$$

$$
\mathcal{C}_{pro}^{XOR}(i): \begin{cases}
w_d[i] \vee \overline{t[i]} = 1 \\
\overline{x_v[i]} \vee x_d[i] = 1 \\
\overline{z_v[i]} \vee z_d[i] = 1 \\
\overline{y_v[i]} \vee y_d[i] = 1 \\
\overline{w_v[i]} \vee w_d[i] = 1 \\
\overline{x_d[i]} \vee \overline{z_d[i]} \vee \overline{t[i]} = 1 \\
\overline{x_d[i]} \vee \overline{y_d[i]} \vee \overline{t[i]} = 1 \\
z_d[i] \vee \overline{w_d[i]} \vee t[i] = 1 \\
y_d[i] \vee \overline{w_d[i]} \vee t[i] = 1 \\
x_d[i] \vee y_d[i] \vee z_d[i] \vee \overline{t[i]} = 1 \\
x_v[i] \vee \overline{y_v[i]} \vee \overline{z_v[i]} \vee \overline{w_v[i]} = 1 \\
\overline{x_v[i]} \vee y_v[i] \vee \overline{z_v[i]} \vee \overline{w_v[i]} = 1 \\
\overline{x_v[i]} \vee \overline{y_v[i]} \vee z_v[i] \vee \overline{w_v[i]} = 1 \\
\overline{x_v[i]} \vee \overline{y_v[i]} \vee \overline{z_v[i]} \vee w_v[i] = 1 \\
\overline{x_d[i]} \vee \overline{y_d[i]} \vee \overline{z_d[i]} \vee w_d[i] = 1 \\
x_d[i] \vee \overline{y_d[i]} \vee \overline{z_d[i]} \vee \overline{w_d[i]} = 1 \\
x_d[i] \vee \overline{y_d[i]} \vee z_d[i] \vee w_d[i] = 1 \\
x_d[i] \vee y_d[i] \vee \overline{z_d[i]} \vee w_d[i] = 1 \\
\overline{x_d[i]} \vee y_d[i] \vee z_d[i] \vee w_d[i] = 1 \\
x_v[i] \vee y_v[i] \vee z_v[i] \vee \overline{w_v[i]} \vee t[i] = 1 \\
x_v[i] \vee \overline{y_v[i]} \vee z_v[i] \vee \overline{z_d[i]} \vee w_v[i] \vee \overline{w_d[i]} = 1 \\
x_v[i] \vee y_v[i] \vee \overline{y_d[i]} \vee \overline{z_v[i]} \vee w_v[i] \vee \overline{w_d[i]} = 1 \\
\overline{x_v[i]} \vee y_v[i] \vee z_v[i] \vee \overline{z_d[i]} \vee w_v[i] \vee \overline{w_d[i]} = 1
\end{cases}
$$

B The Full Model for RIPEMD-160

With all probability components fully modeled, we incorporate them into the framework of [15, Algorithm6], as presented in Algorithm 1.

C Differential Characteristics for 41/42/43-step RIPEMD-160

We provide concrete differential trails for 41-, 42-, and 43-step RIPEMD-160 in Table 12, 13 and 14, respectively.

D Full Implicit Conditions for Differential Characteristics

For the sake of research completeness, the conditions on (LQ_i, RQ_i) used in the 44-, 41-, 42-, and 43-step attacks on RIPEMD-160 are given in Table 15, 16, 17 and 18, respectively.

Algorithm 1 Model the signed difference transitions and probabilities for $a_5 = a_1 \boxplus (F(a_4, a_3, a_2) \boxplus a_0 \boxplus m \boxplus c)^{\lll s}$.

procedure R(fNa, isC, isF, isV, isK, s,∇m,∇a_0,∇a_1,∇a_2,∇a_3,∇a_4,∇a_5,a_4,a_3,a_2,a_5,a_1)
    Claim vectors ∇b_0, ∇b_1, ∇b_2, ∇b_3, ∇b_4, δb_3, δb_4 of size 32
    Claim vectors ∇c_2, ∇c_3, ∇c_{inner}, ∇c_{outer} of size 33
    Claim signed difference vectors ∇q of size $s+1$
    Claim vectors $flag_0$, $flag_1$, pro
    $\nabla b_0 = \nabla m$
    if fNa = "XOR" **then**
        Claim vector t of size 32
        For $i = 0$ to 31 **do**
            add constraints $\mathcal{C}_{pro}^{XOR}(\nabla a_4[i], \nabla a_3[i], \nabla a_2[i], \nabla b_1[i], t[i])$
    else
        Claim two vectors t_1, t_2 of size 32
        For $i = 0$ to 31 **do**
            add constraints $\mathcal{C}_{pro}^{fNa}(\nabla a_4[i], \nabla a_3[i], \nabla a_2[i], \nabla b_1[i], t_1[i], t_2[i])$
    if `isC = 1` **then**
        `BOOLCOND_MODEL`(fNa, ∇a_4, ∇a_3, ∇a_2, ∇b_1, a_4, a_3, a_2)
    $\nabla c_2[0] =$`[=]`, $\nabla c_3[0] =$`[=]`
    `MODADD_MODEL`(∇b_0, ∇b_1, ∇c_2, b_2), `MODADD_MODEL`(∇b_2, ∇a_0, ∇c_3, b_3)
    if `isF = 1` **then**
        `ROTATE_DIFF_FIRST`(s, ∇b_3, ∇a_1, ∇a_5, ∇q, isV, isK, ∇b_4)
    else
        `ROTATE_DIFF_SECOND`(s, ∇b_3, ∇a_1, ∇a_5, ∇q, isV, ∇b_4)
    add constraints $\mathcal{C}_{Convert}(\nabla b_3, \delta b_3, \nabla c_{inner})$
    add constraints $\mathcal{C}_{Convert}(\nabla b_4, \delta b_4, \nabla c_{outer})$
    `ASSERT` $flag_0$ `= IF` $\delta b_3[32-s] = \delta b_4[0]$ `THEN 0bin1 ELSE 0bin0 ENDIF;`
    `ASSERT` $flag_1$ `= IF` $\delta b_3[0] = \delta b_4[s]$ `THEN 0bin1 ELSE 0bin0 ENDIF;`
    add constraints $\mathcal{C}_{pro}^{LRS}(\delta b_3[0:32-s], flag_0, W_0)$
    add constraints $\mathcal{C}_{pro}^{LRS}(\delta b_4[0:s], flag_1, W_1)$
    if `isV = 1` **then**
        $\nabla q[0:s] = \nabla b_4[0:s]$
        `ROTATE_DIFF_FILTER`(s, ∇a_5, ∇a_1, ∇b_3, ∇q, a_5, a_1)
    $pro = W_0 + W_1$
    For $i = 0$ to 31 **do**
        if fNa = "XOR" **then**
            $pro = pro + a_{5d}[i] + t[i]$
        else
            $pro = pro + a_{5d}[i] + t_1[i] + t_2[i]$
procedure `ROTATE_DIFF_FIRST`(s, ∇b_3, ∇a_1, ∇a_5, ∇q, isV, isK, ∇b_4)
    Claim two signed difference vectors $\nabla tempb_4$, ∇b_5 of size 32
    For $i = 0$ to $30 - s$ **do**
        $\nabla tempb_4[i+s \bmod 32] = \nabla b_3[i]$
    For $i = 32 - s$ to 30 **do**
        $\nabla tempb_4[i+s \bmod 32] = \nabla b_3[i]$
    Claim two signed difference vectors ∇c_0, ∇c_{temp} of size 33
    Claim a signed difference vector ∇c_h
    `ROTATE_MODEL`($\nabla b_3[31-s]$, $\nabla b_3[30-s]$, $\nabla tempb_4[31]$,$\nabla tempb_4[30]$, $\nabla c_0[0]$)
    `ROTATE_MODEL`($\nabla b_3[31]$, $\nabla b_3[30]$, $\nabla tempb_4[31+s]$, $\nabla tempb_4[30+s]$, ∇c_h)
    `MODADD_MODEL`(∇a_1, $\nabla tempb_4$, ∇c_0, ∇b_5), `EXPAND_MODEL`(∇b_5, ∇a_5, isK)
    $\nabla c_{temp} =$`[=]`
    `SIGNED_Q_MODEL`($\nabla tempb_4$, $\nabla c_0[0]$, ∇c_{temp}, ∇b_4, 31)
procedure `ROTATE_DIFF_SENCOND`(s, ∇b_3, ∇a_1, ∇a_5, ∇q, isV, isK, ∇b_4)
    Claim a signed difference vectors $\nabla tempb_4$ of size 32
    `EXPAND_MODEL`(∇b_3, $\nabla tempb_4$, 0), `ADDEXP_MODEL`(∇a_1, $\nabla tempb_4 \lll s$, ∇a_5)
    $\nabla b_4 = \nabla tempb_4 \lll s$

Table 12: The 41-step differential characteristic, where $\delta m_{12} = 2^{15}$.

i	∇X_i	$\pi_l(i)$	i	∇Y_i	$\pi_r(i)$
-5	================================		-5	================================	
-4	================================		-4	================================	
-3	================================		-3	================================	
-2	================================		-2	================================	
-1	================================		-1	================================	
0	================================	0	0	================================	5
1	================================	1	1	================================	14
2	================================	2	2	================================	7
3	================================	3	3	================================	0
4	================================	4	4	================================	9
5	================================	5	5	================================	2
6	================================	6	6	================================	11
7	================================	7	7	================================	4
8	================================	8	8	================================	13
9	================================	9	9	================================	6
10	================================	10	10	================================	15
11	================================	11	11	================================	8
12	==========n=====================	12	12	================================	1
13	==nu============================	13	13	====================0==========	10
14	=u========0===============u=====	14	14	====================1==========	3
15	10=0==============n=====110n====	15	15	==========n====================	12
16	u0==============1=0====1=1=0====	7	16	================================	6
17	0=======1========1=====1=n=1====	4	17	==========1====================	11
18	1================0====0u=0======	13	18	==========1====================	3
19	===u==========1=n====00=1======	1	19	u===============================	7
20	===0=========1=0=1====u1=====n==	10	20	================================	0
21	=======1n====0=n=1====0==1===0==	6	21	1==============================0	13
22	=======01===1u=0===1==0======0==	15	22	1==============================1	5
23	=====1=n1===10=1===1==========1=	3	23	====================un=========	10
24	===1=0=0==nnn1=====u==========0=	12	24	================================	14
25	===0=n====000======0=========1n=	0	25	====================001=========	15
26	111u=0===1==1==01==1=====01=0=0=	9	26	====================111=========	8
27	==0001100010===10=011======0=n0=	5	27	==========nnn===================	12
28	nnn10unnnn==0==nunu=1====111000=	2	28	================================	4
29	nn0=0110nn===0=11010n===10ununn=	14	29	============1===================	9
30	10nu=000011=0=1u01nuu=111uuu0001	11	30	==1=============================	1
31	==100=n=1100==u0nnnnn101nu0u=u=0	8	31	==u=============================	2
32	=0n=1=1100=n=0uu10000u==101110==	3	32	==1=============================	15
33	==n1n0n0==10==110===00=10=u=0===	10	33	==1======0====================	5
34	==1n0=1===u===n======u=u1=01====	14	34	=========n==============1=======	1
35	===1==n===0==10======1=10==u====	4	35	=========1==============u=======	3
36	===1==1======n==============1====	9	36	=========1==============10======	7
37	=============0===========1======	15	37	========0================u=====1	14
38	====================0===n====0=	8	38	========n=====0==========1=====n	6
39	====================u===1====u=	1	39	========1=====u================1	9
40	==============n================	2	40	===u============================	11

$X_{10}[11] = X_{11}[21], X_{11}[11] \neq X_{13}[21], X_{11}[18,19] \neq X_{12}[28,29],$
$X_{12}[18] = X_{14}[28], X_{12}[19] \neq X_{14}[29], X_{12}[20] = X_{13}[30], X_{12}[27] \neq X_{13}[5],$
$Y_{16}[31] = Y_{17}[31], Y_{20}[9] = Y_{21}[9], Y_{24}[19,20] = Y_{25}[19,20], Y_{28}[29,30,31] = Y_{29}[29,30,31],$
$\overline{Y_{32}[7]} \vee Y_{33}[7] = 1, \overline{Y_{35}[0]} \vee Y_{36}[0] = 1, \overline{Y_{36}[17]} \vee Y_{37}[17] = 1, \overline{Y_{38}[16]} \vee Y_{39}[16] = 1$

Table 13: The 42-step differential characteristic, where $\delta m_{12} = 2^{15}$.

i	∇X_i	$\pi_l(i)$	i	∇Y_i	$\pi_r(i)$
-5	================================		-5	================================	
-4	================================		-4	================================	
-3	================================		-3	================================	
-2	================================		-2	================================	
-1	================================		-1	================================	
0	================================	0	0	================================	5
1	================================	1	1	================================	14
2	================================	2	2	================================	7
3	================================	3	3	================================	0
4	================================	4	4	================================	9
5	================================	5	5	================================	2
6	================================	6	6	================================	11
7	================================	7	7	================================	4
8	================================	8	8	================================	13
9	================================	9	9	================================	6
10	================================	10	10	================================	15
11	================================	11	11	================================	8
12	==========n=====================	12	12	================================	1
13	==nu============================	13	13	====================0===========	10
14	=u========1===============u=====	14	14	====================1===========	3
15	00=0==============n=====110n====	15	15	==========n====================	12
16	u0==============1=0====1=1=0====	7	16	================================	6
17	0=======1========1=====1=n=1====	4	17	==========1====================	11
18	1================0====0u=0======	13	18	==========1====================	3
19	===u==========1=n====00=1======	1	19	u===============================	7
20	===0======1==1=0=0====u=00===n==	10	20	================================	0
21	0==1===0==1=0=1n=1====1==10==0==	6	21	1============================0	13
22	u======0====u=nn1==1=11==0===1==	15	22	1============================1	5
23	u====1=n===00=00un=0=====u======	3	23	====================un=========	10
24	0=1=11=0===un1=100=n==1==0======	12	24	================================	14
25	===0=n00===00==1===0==1=0====1==	0	25	====================001=========	15
26	=11uun===1==1=0====0==n=u====11=	9	26	====================111=========	8
27	==0100=u=1=========10=101==1=u==	5	27	==========nnn===================	12
28	==u1u==0nu==1u1=10=nu00=u111u0==	2	28	================================	4
29	==01000110=11n==1==1111=10=n10==	14	29	============1===================	9
30	==10=1nu=01un01=n=1=0==01u11==00	11	30	==1=============================	1
31	0=u0=001=0100==111==00=uun=n1==1	8	31	==u=============================	2
32	u1nn=1u0=uu1u0=1nu==0101101u0=10	3	32	==1=============================	15
33	unn1=n010unnu1=unn=0nun=un001100	10	33	==1======0====================	5
34	u11u111un11n1==100=10n0=00nu0nn=	14	34	=========n==============1=======	1
35	1100=1001u1111=n1n==101==n10u11=	4	35	=========1==============u=======	3
36	=u=u=1=1=0=1===1011=0n0==0n11011	9	36	=========1==============10======	7
37	=0=0=0=0=0=====1==n==1=1=n1n===0	15	37	========0================u=====1	14
38	========1======0=10====0=n=1===1	8	38	========n=====0==========1=====n	6
39	====0===u========0=======0=====u	1	39	===0====1=====u================1	9
40	====n===1====================1	2	40	===u==========1=================	11
41	===n============================	7	41	========u======u=u==============	8

$X_{10}[11] = X_{11}[21], X_{11}[11] \neq X_{13}[21], X_{11}[18, 19] \neq X_{12}[28, 29]$,
$X_{12}[18] = X_{14}[28], X_{12}[19] \neq X_{14}[29], X_{12}[20] = X_{13}[30], X_{12}[27] \neq X_{13}[5]$,
$\overline{X_{39}[16]} \vee X_{40}[16] = 1$,
$Y_{16}[31] = Y_{17}[31], Y_{20}[9] = Y_{21}[9], Y_{24}[19, 20] = Y_{25}[19, 20], Y_{28}[29, 30, 31] = Y_{29}[29, 30, 31]$,
$\overline{Y_{32}[7]} \vee Y_{33}[7] = 1, \overline{Y_{35}[0]} \vee Y_{36}[0] = 1, \overline{Y_{36}[17]} \vee Y_{37}[17] = 1$,
$\overline{Y_{38}[16]} \vee Y_{39}[16] = 1, \overline{Y_{39}[1]} \vee Y_{40}[1] = 1, \overline{Y_{39}[10]} \vee Y_{40}[10] = 1$

Table 14: The 43-step differential characteristic, where $\delta m_{12} = 2^{15}$.

i	∇X_i	$\pi_l(i)$	i	∇Y_i	$\pi_r(i)$
-5	================================		-5	================================	
-4	================================		-4	================================	
-3	================================		-3	================================	
-2	================================		-2	================================	
-1	================================		-1	================================	
0	================================	0	0	================================	5
1	================================	1	1	================================	14
2	================================	2	2	================================	7
3	================================	3	3	================================	0
4	================================	4	4	================================	9
5	================================	5	5	================================	2
6	================================	6	6	================================	11
7	================================	7	7	================================	4
8	================================	8	8	================================	13
9	================================	9	9	================================	6
10	================================	10	10	================================	15
11	================================	11	11	================================	8
12	==========n=====================	12	12	================================	1
13	==nu============================	13	13	====================0===========	10
14	=u========0==============u=====	14	14	====================1===========	3
15	10=0==============n=====110n====	15	15	==========n====================	12
16	u0==============1=0====1=1=0====	7	16	================================	6
17	0=======1========1=====1=n=1====	4	17	==========1====================	11
18	1================0====0u=0======	13	18	==========1====================	3
19	===u==========1=n====00=0======	1	19	u===============================	7
20	===0=========1=1=0====u0=====n==	10	20	================================	0
21	===0===1====11=n=0====0==1===0==	6	21	1==============================0	13
22	=======n==unnn=0===1101==1===1==	15	22	1==============================1	5
23	=====1=n=11100=0===0=====n======	3	23	====================un=========	10
24	1111u1=0=nunn1=====n=====0===1==	12	24	================================	14
25	===01n=1=0111==1===0=====0===1==	0	25	====================001=========	15
26	111u10===1===0=1===1======1==0=1	9	26	====================111=========	8
27	==00=====0=====n====n=====01=n=1	5	27	==========nnn===================	12
28	=00u=====n=====0====1==011u==0=n	2	28	================================	4
29	===1=1===01==n11===1=1=1n10=1==0	14	29	============1===================	9
30	==0==1===11nnuu00=11=u0nnn=====0	11	30	==1=============================	1
31	10011u===n0unu0=0=n000=110=01==0	8	31	==u=============================	2
32	n00000==0uu00nu=u1nuun0=u==1=00=	3	32	==1=============================	15
33	unu1101=n00uu1n=1n0uuuu=u0=0=nn1	10	33	==1======0====================	5
34	0n110=0=0=0n00u==0110u011u=u0101	14	34	========n==============1=======	1
35	=10=1=0=1=n0=n1==111=011u0=0n==0	4	35	========1==============u=======	3
36	10=00=0=u1u=10u==01==u0=1===u==n	9	36	========1==============10======	7
37	0==u==n=001=n=0===0==nn=====0==1	15	37	========0================u=====1	14
38	==01==1=uu=11========n0==0==1==u	8	38	========n=====0==========1=====n	6
39	===1====u1=01=0======1======0=0u	1	39	===0====1=====u================1	9
40	========1=====n======1===1====00	2	40	===u====0=====10=0==============	11
41	==============1==========n======	7	41	===1====u======u=u==============	8
42	========n======n=n==============	0	42	==========u=========n========nn	12

$X_{10}[11] = X_{11}[21], X_{11}[11] \neq X_{13}[21], X_{11}[18,19] \neq X_{12}[28,29]$,
$X_{12}[18] = X_{14}[28], X_{12}[19] \neq X_{14}[29], X_{12}[20] = X_{13}[30], X_{12}[27] \neq X_{13}[5]$,
$Y_{16}[31] = Y_{17}[31], Y_{20}[9] = Y_{21}[9], Y_{24}[19,20] = Y_{25}[19,20], Y_{28}[29,30,31] = Y_{29}[29,30,31]$,
$\overline{Y_{32}[7]} \vee Y_{33}[7] = 1, \overline{Y_{35}[0]} \vee Y_{36}[0] = 1, \overline{Y_{36}[17]} \vee Y_{37}[17] = 1$,
$\overline{Y_{38}[16]} \vee Y_{39}[16] = 1, \overline{Y_{39}[1]} \vee Y_{40}[1] = 1, \overline{Y_{39}[10]} \vee Y_{40}[10] = 1, \overline{Y_{40}[27]} \vee Y_{41}[27] = 1$

Table 15: Implicit conditions on LQ_i and RQ_i for 44-step differential characteristic.

Implicit conditions on LQ_i and RQ_i:
$(LQ_i \boxplus \alpha_i^l)^{\lll s_i^l} = LQ_i^{\lll s_i^l} \boxplus \beta_i^l$
$(RQ_i \boxplus \alpha_i^r)^{\lll s_i^r} = RQ_i^{\lll s_i^r} \boxplus \beta_i^r$

i	α_i^l	β_i^l	s_i^l	i	α_i^r	β_i^r	s_i^r
0	0x0	0x0	11	0	0x0	0x0	8
1	0x0	0x0	14	1	0x0	0x0	9
2	0x0	0x0	15	2	0x0	0x0	9
3	0x0	0x0	12	3	0x0	0x0	11
4	0x0	0x0	5	4	0x0	0x0	13
5	0x0	0x0	8	5	0x0	0x0	15
6	0x0	0x0	7	6	0x0	0x0	15
7	0x0	0x0	9	7	0x0	0x0	5
8	0x0	0x0	11	8	0x0	0x0	7
9	0x0	0x0	13	9	0x0	0x0	7
10	0x0	0x0	14	10	0x0	0x0	8
11	0x0	0x0	15	11	0x0	0x0	11
12	0x8000	0x200000	6	12	0x0	0x0	14
13	0x200000	0x10000000	7	13	0x0	0x0	14
14	0xefe00000	0xbffffffe0	9	14	0x0	0x0	12
15	0x10000020	0x2010	8	15	0x8000	0x200000	6
16	0x0	0x0	7	16	0x0	0x0	9
17	0x0	0x0	6	17	0x0	0x0	13
18	0x80	0x8000	8	18	0x0	0x0	15
19	0xffff7c00	0xef800000	13	19	0x0	0x0	7
20	0x800000	0x4	11	20	0x0	0x0	12
21	0x4	0x800	9	21	0x0	0x0	8
22	0x10804	0x840200	7	22	0x0	0x0	9
23	0x7cfe00	0x7f00003f	15	23	0x0	0x0	11
24	0x2007fc0	0x3fe001	7	24	0x0	0x0	7
25	0x0	0x0	12	25	0x0	0x0	7
26	0x0	0x0	15	26	0x0	0x0	12
27	0x200002	0x40000400	9	27	0x8000	0x400000	7
28	0x80000000	0xffffffc00	11	28	0x0	0x0	6
29	0x400	0x20000	7	29	0x0	0x0	15
30	0x40080100	0x1fe801	13	30	0x0	0x0	13
31	0x80f	0x80f000	12	31	0x0	0x0	11
32	0xb80f4e00	0x7a6ffdc0	11	32	0x0	0x0	9
33	0x83a02fbe	0x5f7d074	13	33	0x0	0x0	7
34	0x7b0441c5	0xc110715f	6	34	0x80	0x400000	15
35	0xbdb27852	0xd93c28de	7	35	0x0	0x0	11
36	0x4dab71a	0xadc68136	14	36	0x0	0x0	8
37	0xeb2b9a07	0x57340fd6	9	37	0xffffffff	0xfffffc0	6
38	0xe069d15c	0x3a2b7c0d	13	38	0xfffe0000	0xff800000	6
39	0x7857febc	0xff5e3c2c	15	39	0x0	0x0	14
40	0x9234816	0xd2058249	14	40	0xffff0000	0xf0000000	12
41	0x22187fc7	0x187fc722	8	41	0xffffc02	0xff804000	13
42	0x108fb8de	0xf71bc212	13	42	0x8000000	0x1	5
43	0x3e43df0	0xf90f7c01	6	43	0x43e	0x10f8000	14

Table 16: Implicit conditions on LQ_i and RQ_i for 41-step differential characteristic.

Implicit conditions on LQ_i and RQ_i:
$(LQ_i \boxplus \alpha_i^l)^{\lll s_i^l} = LQ_i^{\lll s_i^l} \boxplus \beta_i^l$
$(RQ_i \boxplus \alpha_i^r)^{\lll s_i^r} = RQ_i^{\lll s_i^r} \boxplus \beta_i^r$

i	α_i^l	β_i^l	s_i^l	i	α_i^r	β_i^r	s_i^r
0	0x0	0x0	11	0	0x0	0x0	8
1	0x0	0x0	14	1	0x0	0x0	9
2	0x0	0x0	15	2	0x0	0x0	9
3	0x0	0x0	12	3	0x0	0x0	11
4	0x0	0x0	5	4	0x0	0x0	13
5	0x0	0x0	8	5	0x0	0x0	15
6	0x0	0x0	7	6	0x0	0x0	15
7	0x0	0x0	9	7	0x0	0x0	5
8	0x0	0x0	11	8	0x0	0x0	7
9	0x0	0x0	13	9	0x0	0x0	7
10	0x0	0x0	14	10	0x0	0x0	8
11	0x0	0x0	15	11	0x0	0x0	11
12	0x8000	0x200000	6	12	0x0	0x0	14
13	0x200000	0x10000000	7	13	0x0	0x0	14
14	0xefe00000	0xbffffe0	9	14	0x0	0x0	12
15	0x10000020	0x2010	8	15	0x8000	0x200000	6
16	0x0	0x0	7	16	0x0	0x0	9
17	0x0	0x0	6	17	0x0	0x0	13
18	0x80	0x8000	8	18	0x0	0x0	15
19	0xffff7c00	0xef800000	13	19	0x0	0x0	7
20	0x800000	0x4	11	20	0x0	0x0	12
21	0x4000	0x800000	9	21	0x0	0x0	8
22	0x0	0x0	7	22	0x0	0x0	9
23	0x800000	0x40	15	23	0x0	0x0	11
24	0x7fc0	0x3fe000	7	24	0x0	0x0	7
25	0x0	0x0	12	25	0x0	0x0	7
26	0x0	0x0	15	26	0x0	0x0	12
27	0x0	0x0	9	27	0x8000	0x400000	7
28	0x14	0xa000	11	28	0x0	0x0	6
29	0xbb818000	0xc0bfffde	7	29	0x0	0x0	15
30	0x3e807ff8	0xfff07d0	13	30	0x0	0x0	13
31	0x82c01fee	0x1fee82c	12	31	0x0	0x0	11
32	0x1023b19f	0x1d8cf881	11	32	0x0	0x0	9
33	0x26e14f03	0x29e044dd	13	33	0x0	0x0	7
34	0x4f0eeb	0x13c3bac0	6	34	0x80	0x400000	15
35	0xd00c3ea0	0x61f4fe8	7	35	0x0	0x0	11
36	0xfdff3050	0xcc13ff80	14	36	0x0	0x0	8
37	0xac000040	0x7f58	9	37	0xffffffff	0xfffffc0	6
38	0x3c0a0	0x78140000	13	38	0x20000	0x800000	6
39	0x77ec0000	0x3bf6	15	39	0x0	0x0	14
40	0xffffc008	0xf0020000	14	40	0xffff0000	0xf0000000	12

Table 17: Implicit conditions on LQ_i and RQ_i for 42-step differential characteristic.

Implicit conditions on LQ_i and RQ_i:
$(LQ_i \boxplus \alpha_i^l)^{\lll s_i^l} = LQ_i^{\lll s_i^l} \boxplus \beta_i^l$
$(RQ_i \boxplus \alpha_i^r)^{\lll s_i^r} = RQ_i^{\lll s_i^r} \boxplus \beta_i^r$

i	α_i^l	β_i^l	s_i^l	i	α_i^r	β_i^r	s_i^r
0	0x0	0x0	11	0	0x0	0x0	8
1	0x0	0x0	14	1	0x0	0x0	9
2	0x0	0x0	15	2	0x0	0x0	9
3	0x0	0x0	12	3	0x0	0x0	11
4	0x0	0x0	5	4	0x0	0x0	13
5	0x0	0x0	8	5	0x0	0x0	15
6	0x0	0x0	7	6	0x0	0x0	15
7	0x0	0x0	9	7	0x0	0x0	5
8	0x0	0x0	11	8	0x0	0x0	7
9	0x0	0x0	13	9	0x0	0x0	7
10	0x0	0x0	14	10	0x0	0x0	8
11	0x0	0x0	15	11	0x0	0x0	11
12	0x8000	0x200000	6	12	0x0	0x0	14
13	0x200000	0x10000000	7	13	0x0	0x0	14
14	0xefe00000	0xbffffffe0	9	14	0x0	0x0	12
15	0x10000020	0x2010	8	15	0x8000	0x200000	6
16	0x0	0x0	7	16	0x0	0x0	9
17	0x0	0x0	6	17	0x0	0x0	13
18	0x80	0x8000	8	18	0x0	0x0	15
19	0xffff7c00	0xef800000	13	19	0x0	0x0	7
20	0x800000	0x4	11	20	0x0	0x0	12
21	0x0	0x0	9	21	0x0	0x0	8
22	0xfffe00	0x7fff0000	7	22	0x0	0x0	9
23	0x7fff0000	0x7fffc000	15	23	0x0	0x0	11
24	0x0	0x0	7	24	0x0	0x0	7
25	0x0	0x0	12	25	0x0	0x0	7
26	0x7000000	0x380	15	26	0x0	0x0	12
27	0xfc000080	0x101f8	9	27	0x8000	0x400000	7
28	0xeefeff80	0xf7fc0778	11	28	0x0	0x0	6
29	0x800	0x40000	7	29	0x0	0x0	15
30	0x800794	0xf28010	13	30	0x0	0x0	13
31	0xed3e0001	0xe0000ed4	12	31	0x0	0x0	11
32	0x11f7af4c	0xbd7a608f	11	32	0x0	0x0	9
33	0x2dfe9f3d	0xd3e785c0	13	33	0x0	0x0	7
34	0x4a364410	0x8d910412	6	34	0x80	0x400000	15
35	0x6fff8c01	0xffc600b8	7	35	0x0	0x0	11
36	0x15a53c01	0x4f004569	14	36	0x0	0x0	8
37	0x60b07491	0x60e920c1	9	37	0xffffffff	0xffffffc0	6
38	0x5415ff7e	0xbfefaa82	13	38	0x20000	0x800000	6
39	0x3ffff4fe	0xfa7f2000	15	39	0x0	0x0	14
40	0x5001fbe	0x7ef8140	14	40	0xffff0000	0xf0000000	12
41	0xf7ec0	0xf7ec000	8	41	0xfffffbfe	0xff7fc000	13

Table 18: Implicit conditions on LQ_i and RQ_i for 43-step differential characteristic.

Implicit conditions on LQ_i and RQ_i:
$(LQ_i \boxplus \alpha_i^l)^{\lll s_i^l} = LQ_i^{\lll s_i^l} \boxplus \beta_i^l$
$(RQ_i \boxplus \alpha_i^r)^{\lll s_i^r} = RQ_i^{\lll s_i^r} \boxplus \beta_i^r$

i	α_i^l	β_i^l	s_i^l	i	α_i^r	β_i^r	s_i^r
0	0x0	0x0	11	0	0x0	0x0	8
1	0x0	0x0	14	1	0x0	0x0	9
2	0x0	0x0	15	2	0x0	0x0	9
3	0x0	0x0	12	3	0x0	0x0	11
4	0x0	0x0	5	4	0x0	0x0	13
5	0x0	0x0	8	5	0x0	0x0	15
6	0x0	0x0	7	6	0x0	0x0	15
7	0x0	0x0	9	7	0x0	0x0	5
8	0x0	0x0	11	8	0x0	0x0	7
9	0x0	0x0	13	9	0x0	0x0	7
10	0x0	0x0	14	10	0x0	0x0	8
11	0x0	0x0	15	11	0x0	0x0	11
12	0x8000	0x200000	6	12	0x0	0x0	14
13	0x200000	0x10000000	7	13	0x0	0x0	14
14	0xefe00000	0xbffffe0	9	14	0x0	0x0	12
15	0x10000020	0x2010	8	15	0x8000	0x200000	6
16	0x0	0x0	7	16	0x0	0x0	9
17	0x0	0x0	6	17	0x0	0x0	13
18	0x80	0x8000	8	18	0x0	0x0	15
19	0xffff7c00	0xef800000	13	19	0x0	0x0	7
20	0x800000	0x4	11	20	0x0	0x0	12
21	0x0	0x0	9	21	0x0	0x0	8
22	0x20000	0x1000000	7	22	0x0	0x0	9
23	0x1000000	0x80	15	23	0x0	0x0	11
24	0x1f08000	0xf8400000	7	24	0x0	0x0	7
25	0x0	0x0	12	25	0x0	0x0	7
26	0xfff80000	0xfffffffc	15	26	0x0	0x0	12
27	0x4	0x800	9	27	0x8000	0x400000	7
28	0x20000	0x10000000	11	28	0x0	0x0	6
29	0xe0000801	0x40070	7	29	0x0	0x0	15
30	0xf0000090	0x11fe00	13	30	0x0	0x0	13
31	0xffff8141	0xf8141000	12	31	0x0	0x0	11
32	0xf7eff440	0x7fa207bf	11	32	0x0	0x0	9
33	0xc348341	0x90682186	13	33	0x0	0x0	7
34	0xbfe05bef	0xf816fbb0	6	34	0x80	0x400000	15
35	0x2e5f47ff	0x2fa3ff97	7	35	0x0	0x0	11
36	0xe7e9e4bf	0x792ff9fa	14	36	0x0	0x0	8
37	0xbea4c0f7	0x4981ef7e	9	37	0xffffffff	0xffffffc0	6
38	0x17fe3a8a	0xc75142ff	13	38	0x20000	0x800000	6
39	0xbffedf03	0x6f81dfff	15	39	0x0	0x0	14
40	0x70062048	0x88121c02	14	40	0xffff0000	0xf0000000	12
41	0x77dfe800	0xdfe80078	8	41	0xffffffbfe	0xff7fc000	13
42	0x2018038a	0x714403	13	42	0x7ff8000	0xfff00001	5

E Rationale Behind the Bit Conditions on X_{35}

In this section, we provide a detailed explanation of why the specific bit conditions on X_{35} (highlighted in blue in Table 7) are sufficient to ensure that the required condition on LQ_{39} is always satisfied.

According to Table 15, LQ_{39} must satisfy the equation $(LQ_{39} \boxplus \texttt{0x7857febc})^{\lll 15} = LQ_{39}^{\lll 15} \boxplus \texttt{0xff5e3c2c}$, which holds with a probability of $2^{-0.92}$. To make this equation always valid, i.e., to ensure that $H = L'$ and $L = H'$, it is sufficient that any bit in $LQ_{39}[16 \sim 9]$ is 1 and that $LQ_{39}[31] = 0$, as shown in Table 19. Note that $LQ_{39}^{\lll 15} = X_{39} - X_{35}^{\lll 10}$, we illustrate the computation of this expression in Table 20, which clarifies how the extra bit conditions on X_{35} are derived. Specifically, setting $X_{35}[2] = 0$ and $X_{35}[4] = 1$ ensures that $LQ_{39}[31] = 0$, while $X_{35}[21] = 0$ guarantees that at least one bit in $LQ_{39}[16 \sim 12]$ is 1. In summary, if all the bit conditions on X_{35} and X_{39}, including the extra ones, are satisfied, then the equation $(LQ_{39} \boxplus \texttt{0x7857febc})^{\lll 15} = LQ_{39}^{\lll 15} \boxplus \texttt{0xff5e3c2c}$ will always hold.

Table 19: Deduce conditions on LQ_{39}.

LQ_{35}	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LQ_{39}																1	1	1	1	1	1	1	1									
a_{39}^{l}	0	1	1	1	1	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0
	H															L																
$LQ_{39}^{\ggg 15}$	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
$LQ_{39}^{\ggg 15}$																		0														
β_{39}^{l}	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	1	0	1	1	0	0
	H'																	L'														

Table 20: Deduce extra conditions on X_{35}.

X_{39}	1	=	=	=	=	1	=	=	=	=	=	=	=	0	n	1	=	u	=	=	1	u	=	0	=	=	n	1	=	=	1	=
$X_{35}^{\lll 10}$	0	0	0	0	=	1	0	=	n	0	n	=	0	n	=	0	u	1	0	0	0	1	1	0	=	1	1	u	1	n	1	0
$LQ_{39}^{\lll 15}$	1	1	1	1	1													0														

By doing so, we incorporate part of the conditions from Phase 2 of the first stage into Phase 1 of the message modification process, thereby significantly reducing the time complexity of generating starting points. To the best of our knowledge, this is the first application of such a technique for complexity reduction in this context. However, we did not observe similarly favorable properties in the other attacks presented in this paper.

F Detailed Clustering Procedures

As follows, we present the detailed procedures for applying differential clustering to the 41-, 42-, and 43-step differential characteristics.

On the 41-step Differential Characteristic. We randomly select 2^{38} valid tuples $(Y_{22}, Y_{23}, Y_{24}, Y_{25}, Y_{26})$ along with the corresponding message words to estimate the probabilities that $(\delta Y_{36}, \delta Y_{37}, \delta Y_{38}, \delta Y_{39}, \delta X_{40})$ are satisfied. This experiment is repeated 200 times. The estimated probabilities are $2^{-14.397}$, $2^{-2.373}$,

$2^{-2.941}$, $2^{-4.937}$ and $2^{-4.290}$, respectively. Additionally, to ensure cancellation between the left and right branches, we evaluate the probability that the equation $\delta X_{39} \boxplus \delta Y_{38}^{\lll 10} = 0$ holds. The experimental results suggest that this probability is approximately $2^{-0.003}$.

On the 42-step Differential Characteristic. Similarly, we randomly select 2^{38} valid tuples $(Y_{25}, Y_{26}, Y_{27}, Y_{28}, Y_{29})$ and corresponding message words to estimate the probabilities that $(\delta Y_{37}, \delta Y_{38}, \delta Y_{39}, \delta Y_{40}, \delta Y_{41})$ are satisfied. The estimated probabilities are $2^{-8.146}$, $2^{-3.917}$, $2^{-5.703}$, $2^{-4.415}$ and $2^{-3.817}$, respectively. For the cancellation conditions, the probabilities that $\delta X_{39}^{\lll 10} \boxplus \delta Y_{38}^{\lll 10} = 0$ and $\delta X_{40} \boxplus \delta Y_{39}^{\lll 10} = 0$ hold are approximately $2^{-0.003}$ and $2^{-0.046}$, respectively.

On the 43-step Differential Characteristic. We randomly select 2^{38} valid tuples $(Y_{29}, Y_{30}, Y_{31}, Y_{32}, Y_{33})$ and corresponding message words to estimate the probability that $(\delta Y_{38}, \delta Y_{39}, \delta Y_{40}, \delta Y_{41}, \delta Y_{42})$ are satisfied. The probabilities are $2^{-5.716}$, $2^{-6.191}$, $2^{-4.529}$, $2^{-4.379}$ and $2^{-6.696}$, respectively. And the probabilities that the equations $\delta X_{39}^{\lll 10} \boxplus \delta Y_{38}^{\lll 10} = 0$, $\delta X_{40}^{\lll 10} \boxplus \delta Y_{39}^{\lll 10} = 0$ and $\delta X_{41} \boxplus \delta Y_{40}^{\lll 10} = 0$ hold are $2^{-0.003}$, $2^{-0.046}$ and $2^{-0.093}$, respectively.

G Solutions of Initial Starting Points

In this section, we provide the initial starting points for the 42-step and 43-step SFS collision attacks, as listed in Table 21.

H Detailed Attacks Procedures

In this section, we present the detailed procedures and complexity analyses of the 41-, 42-, and 43-step SFS collision attacks, as summarized in Table 22, Table 23, and Table 24, respectively.

I More Colliding Message Pairs

In this section, we give more colliding message pairs for 41 steps and 42 steps of RIPEMD-160, as shown in Table 25, Table 26 and Table 27.

Table 21: Solution for the initial starting points $X_i(12 \leq i \leq 40)$.

	43-step	44-step
X_{12}	1011110100n10100101010001111111 10	1011011100n00101001101011111 0010
X_{13}	11nu10000001101010010010000001	00nu01011010010011001100000101011
X_{14}	1u11110000010100110000110 0u00111	0u11111100111001101111 0000u11110
X_{15}	10100000100011000 0n00101110n0101	0000100000001100 11n11010110n1000
X_{16}	u00111101111101011010110100 0101	u01000000000111110010011011010 01
X_{17}	01001001100010000010000111n010010	01110010110110010101001 10n110111
X_{18}	10001111100101010001110u10000100	11100100011111111010100u10111011
X_{19}	000u1000000011111n11110010100100	010u1011101100011n11000010000100
X_{20}	10100010101111110000 11u011000n10	11001111101111001000 10u101110n10
X_{21}	001011011011111n1011010111010010	111101111010101n1010n10111001110
X_{22}	0101100n01unnn000001101011001100	1001110nu0101101010100n100100010
X_{23}	1110011n1111001000000000 1n110001	n1001101011000110001100011101 00u
X_{24}	1111u1101nunn111000n011010100111	011110100n01u010110u10110000111n
X_{25}	00101n1110111001000000100011111 0	01000n0010n1011111100111101010 00
X_{26}	111u10101100100110011000 10100011	111110111110n00100111111110101n1
X_{27}	010010101000000n0101n10000010n01	0n10110000100101101100n110011000
X_{28}	000u11111n1110101000101011u0000n	1un010100u00111100011001110 0100n
X_{29}	0101110110101n1101010101n1001010	n1000000101001n010101110011n1000
X_{30}	01011111011nnuu001110u0nnn111110	1nu1001000n01101010u01011011011n
X_{31}	10011u000n0unu0001n0001110001100	00101110n11nu00n10un100n01000011
X_{32}	n00000000uu00nu0u1nuun00u0111001	n1111u1uu1un1011110010nu0n11100u
X_{33}	unu11011n00uu1n11n0uuuu0u0001nn1	101n10u00000u100100n01n1n011uu00
X_{34}	0n110000000n00u100110u011u1u0101	1n010001nn1n1001n000u1u11un0100u
X_{35}	1100110010n01n1011110011u010n010	10111u1n1000001100n0n00n10u10001
X_{36}	10100100u1u010u000100u011111u11n	u01u00u01u1n1u00u000001u11n11101
X_{37}	001u01n00010n10011001nn100110001	1n11u111u01100u101u00010110n10u0
X_{38}	00010010uu11100000001n010000110u	u1100u11010110nn1010u01n100n0010
X_{39}	01111111u10011011000111111001000u	11011101011010n10u011u1011n11010
X_{40}	00110011100111n00000111101101000	10011110111101n10011111111110101
$m_0 = 0$x5adb5817, $m_1 = 0$x19a02241, $m_2 = 0$x4bc27a1a, $m_3 = 0$x856fb365, $m_4 = 0$x5cca826e, $m_5 = 0$x31b88c52, $m_6 = 0$xbbc7ab66, $m_8 = 0$x6818ce71, $m_9 = 0$x8e876ac6, $m_{10} = 0$x5557fc9e, $m_{11} = 0$xc6145578, $m_{12} = 0$x482a8579, $m_{13} = 0$x96d1b084, $m_{14} = 0$x1796261b, $m_{15} = 0$x6d790c70.		$m_0 = 0$xa2b302bb, $m_1 = 0$xc754c76c, $m_2 = 0$x996a8bfb, $m_3 = 0$xbf2518fa, $m_4 = 0$xd127634a, $m_5 = 0$xe17bb553, $m_6 = 0$xed462a4a, $m_8 = 0$x8d518273, $m_9 = 0$x993a14a5, $m_{10} = 0$xabe6a658, $m_{11} = 0$x5dd204fc, $m_{12} = 0$xd1161be2, $m_{13} = 0$xb11f359f, $m_{14} = 0$xfb1e00ea, $m_{15} = 0$x9b8423fd.

Table 22: Detailed computation of complexity for the 41-step attack.

$n_1 = 3, p_1 = 9.005, p_2 = 0, p_3 = 52.634$					
The first stage: the probability to be verified is $2^{-p_1} = 2^{-9.005}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	X_{12}	$X_{12}[21] = 0$ $X_{12}[18] = X_{14}[28]$ $X_{12}[19] \neq X_{14}[29]$ $X_{12}[20] = X_{13}[30]$ $X_{12}[27] \neq X_{13}[5]$	2^{-5}	$2^{-4.005}$	$\frac{1}{2\times 41} \times 2^{9.005}$
2	m_{13}, m_1 X_{39}	LQ_{39} $X_{39}[1, 6, 10] = 1$	$2^{-3.912}$	$2^{-0.093}$	$\frac{3}{2\times 41} \times 2^{4.005}$
3	X_{40}	LQ_{40}	$2^{-0.093}$	-	$\frac{1}{2\times 41} \times 2^{0.093}$
Total	-	-	-	-	$2^{2.779}$
The time complexity of the first stage is $2^{(p_2+p_3)-(32-n_1)} \times 2^{2.779} = 2^{23.634} \times 2^{2.779} \approx 2^{26.41}$					
The second stage: the probability to be verified is $2^{-p_2-p_3} = 2^{-52.634}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	m_7	LQ_{15}	$2^{-0.093}$	$2^{-52.541}$	$\frac{1}{2\times 41} \times 2^{52.634}$
2	X_{10}	LQ_{14} $X_{10}[11] = X_{11}[21]$	$2^{-1.508}$	$2^{-51.033}$	$\frac{1}{2\times 41} \times 2^{52.541}$
3	X_9	LQ_{13}	$2^{-0.093}$	$2^{-50.940}$	$\frac{1}{2\times 41} \times 2^{51.033}$
4	X_8	LQ_{12}	$2^{-0.001}$	$2^{-50.939}$	$\frac{1}{2\times 41} \times 2^{50.940}$
5	Y_{13}	$Y_{13}[11] = 0$	2^{-1}	$2^{-49.939}$	$\frac{27}{2\times 41} \times 2^{50.939}$
6	Y_{14}	$Y_{14}[11] = 1$	2^{-1}	$2^{-48.939}$	$\frac{1}{2\times 41} \times 2^{49.939}$
7	Y_{15}	RQ_{15} $Y_{15}[21] = 0$	$2^{-1.001}$	$2^{-47.938}$	$\frac{1}{2\times 41} \times 2^{48.939}$
8	Y_{16}, Y_{17}	$Y_{17}[21] = 1$ $Y_{16}[31] = Y_{17}[31]$	2^{-2}	$2^{-45.938}$	$\frac{2}{2\times 41} \times 2^{47.938}$
9	Y_{18}	$Y_{18}[21] = 1$	2^{-1}	$2^{-44.938}$	$\frac{1}{2\times 41} \times 2^{45.938}$
10	Y_{19}	$Y_{19}[31] = 1$	2^{-1}	$2^{-43.938}$	$\frac{1}{2\times 41} \times 2^{44.938}$
11	Y_{20}, Y_{21}	$Y_{21}[0] = 0$ $Y_{21}[31] = 1$ $Y_{20}[9] = Y_{21}[9]$	2^{-3}	$2^{-40.938}$	$\frac{2}{2\times 41} \times 2^{43.938}$
12	Y_{22}	$Y_{22}[0, 31] = 1$	2^{-2}	$2^{-38.938}$	$\frac{1}{2\times 41} \times 2^{40.938}$
13	Y_{23}	$Y_{23}[9] = 0$ $Y_{23}[10] = 1$	2^{-2}	$2^{-36.938}$	$\frac{1}{2\times 41} \times 2^{38.938}$
14	Y_{24}, Y_{25}	$Y_{25}[9] = 1$ $Y_{25}[10, 11] = 0$ $Y_{24}[19, 20] = Y_{25}[19, 20]$	2^{-5}	$2^{-31.938}$	$\frac{2}{2\times 41} \times 2^{36.938}$
15	Y_{26}	$Y_{26}[9, 10, 11] = 1$	2^{-3}	$2^{-28.938}$	$\frac{1}{2\times 41} \times 2^{31.938}$
16	Y_{36}	δY_{36} $\delta X_{37}^{\lll 10} \boxplus \delta Y_{36}^{\lll 10} = 0$	$2^{-14.397}$	$2^{-14.541}$	$\frac{10}{2\times 41} \times 2^{28.938}$
17	Y_{37}	δY_{37} $\delta X_{38}^{\lll 10} \boxplus \delta Y_{37}^{\lll 10} = 0$	$2^{-2.373}$	$2^{-12.168}$	$\frac{1}{2\times 41} \times 2^{14.541}$
18	Y_{38}	δY_{38} $\delta X_{39} \boxplus \delta Y_{38}^{\lll 10} = 0$	$2^{-2.941}$	$2^{-9.227}$	$\frac{1}{2\times 41} \times 2^{12.168}$
19	Y_{39}	δY_{39}	$2^{-4.937}$	$2^{-4.290}$	$\frac{1}{2\times 41} \times 2^{9.227}$
20	Y_{40}	δY_{40}	$2^{-4.290}$	-	$\frac{1}{2\times 41} \times 2^{4.290}$
Total	-	-	-	-	$2^{49.77}$
The time complexity of the second stage is $2^{49.77}$					
The total time complexity of the whole attack is $2^{49.8}$					

† Time complexity is evaluated by one 41-step compression of RIPEMD-160.

Table 23: Detailed computation of complexity for the 42-step attack.

$n_1 = 3, p_1 = 13.936, p_2 = 0.090, p_3 = 56.744$					
The first stage: the probability to be verified is $2^{-p_1} = 2^{-13.936}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	X_{12}	$X_{12}[21] = 0$ $X_{12}[18] = X_{14}[28]$ $X_{12}[19] \neq X_{14}[29]$ $X_{12}[20] = X_{13}[30]$ $X_{12}[27] \neq X_{13}[5]$	2^{-5}	$2^{-8.936}$	$\frac{1}{2\times 42} \times 2^{13.936}$
2	m_{13}, m_1 X_{39}, X_{40}	$X_{40}[27] = 0$ $X_{39}[0, 23] = 1$ $X_{40}[0, 23] = 1$ $X_{39}[6, 14, 27] = 0$ $\overline{X_{39}[16]} \vee X_{40}[16] = 1$ $LQ_{39}(pro \approx 2^{-0.447})$ $LQ_{40}(pro \approx 2^{-0.074})$	$2^{-8.936}$	–	$\frac{4}{2\times 42} \times 2^{8.936}$
Total	-	-	-	-	$2^{7.714}$
The time complexity of the first stage is $2^{(p_2+p_3)-(32-n_1)} \times 2^{7.714} = 2^{27.834} \times 2^{7.714} \approx 2^{35.55}$					
The second stage: the probability to be verified is $2^{-p_2-p_3} = 2^{-56.834}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	m_7	LQ_{15}	$2^{-0.093}$	$2^{-56.741}$	$\frac{1}{2\times 42} \times 2^{56.834}$
2	X_{10}	LQ_{14} $X_{10}[11] = X_{11}[21]$	$2^{-1.508}$	$2^{-55.233}$	$\frac{1}{2\times 42} \times 2^{56.741}$
3	X_9	LQ_{13}	$2^{-0.093}$	$2^{-55.140}$	$\frac{1}{2\times 42} \times 2^{55.233}$
4	X_{41}	LQ_{41}	$2^{-0.090}$	$2^{-55.050}$	$\frac{1}{2\times 42} \times 2^{55.140}$
5	X_8	LQ_{12}	$2^{-0.001}$	$2^{-55.049}$	$\frac{1}{2\times 42} \times 2^{55.050}$
6	Y_{13}	$Y_{13}[11] = 0$	2^{-1}	$2^{-54.049}$	$\frac{27}{2\times 42} \times 2^{55.049}$
7	Y_{14}	$Y_{14}[11] = 1$	2^{-1}	$2^{-53.049}$	$\frac{1}{2\times 42} \times 2^{54.049}$
8	Y_{15}	RQ_{15} $Y_{15}[21] = 0$	$2^{-1.001}$	$2^{-52.048}$	$\frac{1}{2\times 42} \times 2^{53.049}$
9	Y_{16}, Y_{17}	$Y_{17}[21] = 1$ $Y_{16}[31] = Y_{17}[31]$	2^{-2}	$2^{-50.048}$	$\frac{2}{2\times 42} \times 2^{52.048}$
10	Y_{18}	$Y_{18}[21] = 1$	2^{-1}	$2^{-49.048}$	$\frac{1}{2\times 42} \times 2^{50.048}$
11	Y_{19}	$Y_{19}[31] = 1$	2^{-1}	$2^{-48.048}$	$\frac{1}{2\times 42} \times 2^{49.048}$
12	Y_{20}, Y_{21}	$Y_{21}[0] = 0$ $Y_{21}[31] = 1$ $Y_{20}[9] = Y_{21}[9]$	2^{-3}	$2^{-45.048}$	$\frac{2}{2\times 42} \times 2^{48.048}$
13	Y_{22}	$Y_{22}[0, 31] = 1$	2^{-2}	$2^{-43.048}$	$\frac{1}{2\times 42} \times 2^{45.048}$
14	Y_{23}	$Y_{23}[9] = 0$ $Y_{23}[10] = 1$	2^{-2}	$2^{-41.048}$	$\frac{1}{2\times 42} \times 2^{43.048}$
15	Y_{24}, Y_{25}	$Y_{25}[9] = 1$ $Y_{25}[10, 11] = 0$ $Y_{24}[19, 20] = Y_{25}[19, 20]$	2^{-5}	$2^{-36.048}$	$\frac{2}{2\times 42} \times 2^{41.048}$
16	Y_{26}	$Y_{26}[9, 10, 11] = 1$	2^{-3}	$2^{-33.048}$	$\frac{1}{2\times 42} \times 2^{36.048}$
17	Y_{27}	RQ_{27} $Y_{27}[19, 20, 21] = 0$	$2^{-3.001}$	$2^{-30.047}$	$\frac{1}{2\times 42} \times 2^{33.048}$
18	Y_{28}, Y_{29}	$Y_{29}[19] = 1$ $Y_{28}[31 \sim 29] = Y_{29}[31 \sim 29]$	2^{-4}	$2^{-26.047}$	$\frac{2}{2\times 42} \times 2^{30.047}$
19	Y_{37}	δY_{37} $\delta X_{38}^{\lll 10} \boxplus \delta Y_{37}^{\lll 10} = 0$	$2^{-8.146}$	$2^{-17.901}$	$\frac{8}{2\times 42} \times 2^{26.047}$
20	Y_{38}	δY_{38} $\delta X_{39}^{\lll 10} \boxplus \delta Y_{38}^{\lll 10} = 0$	$2^{-3.920}$	$2^{-13.981}$	$\frac{1}{2\times 42} \times 2^{17.901}$
21	Y_{39}	δY_{39} $\delta X_{40} \boxplus \delta Y_{39}^{\lll 10} = 0$	$2^{-5.749}$	$2^{-8.232}$	$\frac{1}{2\times 42} \times 2^{13.981}$
22	Y_{40}	δY_{40}	$2^{-4.415}$	$2^{-3.817}$	$\frac{1}{2\times 42} \times 2^{8.232}$
23	Y_{41}	δY_{41}	$2^{-3.817}$	-	$\frac{1}{2\times 42} \times 2^{3.817}$
Total	-	-	-	-	$2^{53.90}$
The time complexity of the second stage is $2^{53.90}$					
The total time complexity of the whole attack is $2^{53.9}$					

† Time complexity is evaluated by one 42-step compression of RIPEMD-160.

Table 24: Detailed computation of complexity for the 43-step attack.

$n_1 = 3, p_1 = 25.567, p_2 = 3.302, p_3 = 63.765$

The first stage: the probability to be verified is $2^{-p1} = 2^{-25.567}$ in total

Step	Computed States	Verify Conditions	Verify Probabilities	Remaining Probabilities	Time Complexity†
1	X_{12}	$X_{12}[21] = 0$ $X_{12}[18] = X_{14}[28]$ $X_{12}[19] \neq X_{14}[29]$ $X_{12}[20] = X_{13}[30]$ $X_{12}[27] \neq X_{13}[5]$	2^{-5}	$2^{-20.567}$	$\frac{1}{2\times 43} \times 2^{25.567}$
2	m_{13}, m_1 X_{39}	$X_{39}[1, 3, 17, 20] = 0$ $X_{39}[0, 10, 19, 22, 23, 28] = 1$ $LQ_{39}(pro \approx 2^{-2.825})$	$2^{-12.825}$	$2^{-7.742}$	$\frac{3}{2\times 43} \times 2^{20.567}$
3	X_{40}	$X_{40}[0, 1, 17] = 0$ $X_{40}[6, 10, 23] = 1$ $LQ_{40}(pro \approx 2^{-1.742})$	$2^{-7.742}$	-	$\frac{1}{2\times 43} \times 2^{7.742}$
Total	-	-	-	-	$2^{19.270}$

The time complexity of the first stage is $2^{(p_2+p_3)-(32-n_1)} \times 2^{19.270} = 2^{38.067} \times 2^{19.270} \approx 2^{57.34}$

The second stage: the probability to be verified is $2^{-p_2-p_3} = 2^{-67.067}$ in total

Step	Computed States	Verify Conditions	Verify Probabilities	Remaining Probabilities	Time Complexity†
1	m_7	LQ_{15}	$2^{-0.093}$	$2^{-66.974}$	$\frac{1}{2\times 43} \times 2^{67.067}$
2	X_{41}	$X_{41}[6] = 0$ $X_{41}[17] = 1$ $LQ_{41}(pro \approx 2^{-1.106})$	$2^{-3.106}$	$2^{-63.868}$	$\frac{1}{2\times 43} \times 2^{66.974}$
3	X_{10}	LQ_{14} $X_{10}[11] = X_{11}[21]$	$2^{-1.508}$	$2^{-62.360}$	$\frac{1}{2\times 43} \times 2^{63.868}$
4	X_{42}	LQ_{42}	$2^{-0.196}$	$2^{-62.164}$	$\frac{1}{2\times 43} \times 2^{62.360}$
5	X_9	LQ_{13}	$2^{-0.093}$	$2^{-62.071}$	$\frac{1}{2\times 43} \times 2^{62.164}$
6	X_8	LQ_{12}	$2^{-0.001}$	$2^{-62.070}$	$\frac{1}{2\times 43} \times 2^{62.071}$
7	Y_{13}	$Y_{13}[11] = 0$	2^{-1}	$2^{-61.070}$	$\frac{27}{2\times 43} \times 2^{62.070}$
8	Y_{14}	$Y_{14}[11] = 1$	2^{-1}	$2^{-60.070}$	$\frac{1}{2\times 43} \times 2^{61.070}$
9	Y_{15}	RQ_{15} $Y_{15}[21] = 0$	$2^{-1.001}$	$2^{-59.069}$	$\frac{1}{2\times 43} \times 2^{60.070}$
10	Y_{16}, Y_{17}	$Y_{17}[21] = 1$ $Y_{16}[31] = Y_{17}[31]$	2^{-2}	$2^{-57.069}$	$\frac{2}{2\times 43} \times 2^{59.069}$
11	Y_{18}	$Y_{18}[21] = 1$	2^{-1}	$2^{-56.069}$	$\frac{1}{2\times 43} \times 2^{57.069}$
12	Y_{19}	$Y_{19}[31] = 1$	2^{-1}	$2^{-55.069}$	$\frac{1}{2\times 43} \times 2^{56.069}$
13	Y_{20}, Y_{21}	$Y_{21}[0] = 0$ $Y_{21}[31] = 1$ $Y_{20}[9] = Y_{21}[9]$	2^{-3}	$2^{-52.069}$	$\frac{2}{2\times 43} \times 2^{55.069}$
14	Y_{22}	$Y_{22}[0, 31] = 1$	2^{-2}	$2^{-50.069}$	$\frac{1}{2\times 43} \times 2^{52.069}$
15	Y_{23}	$Y_{23}[9] = 0$ $Y_{23}[10] = 1$	2^{-2}	$2^{-48.069}$	$\frac{1}{2\times 43} \times 2^{50.069}$
16	Y_{24}, Y_{25}	$Y_{25}[9] = 1$ $Y_{25}[10, 11] = 0$ $Y_{24}[19, 20] = Y_{25}[19, 20]$	2^{-5}	$2^{-43.069}$	$\frac{2}{2\times 43} \times 2^{48.069}$
17	Y_{26}	$Y_{26}[9, 10, 11] = 1$	2^{-3}	$2^{-40.069}$	$\frac{1}{2\times 43} \times 2^{43.069}$
18	Y_{27}	RQ_{27} $Y_{27}[19, 20, 21] = 0$	$2^{-3.001}$	$2^{-37.068}$	$\frac{1}{2\times 43} \times 2^{40.069}$
19	Y_{28}, Y_{29}	$Y_{29}[19] = 1$ $Y_{28}[31 \sim 29] = Y_{29}[31 \sim 29]$	2^{-4}	$2^{-33.068}$	$\frac{2}{2\times 43} \times 2^{37.068}$
20	Y_{30}	$Y_{30}[29] = 1$	2^{-1}	$2^{-32.068}$	$\frac{1}{2\times 43} \times 2^{33.068}$
21	Y_{31}	$Y_{31}[29] = 1$	2^{-1}	$2^{-31.068}$	$\frac{1}{2\times 43} \times 2^{32.068}$
22	Y_{32}, Y_{33}	$Y_{32}[29] = 1$ $Y_{33}[22] = 0$ $Y_{33}[29] = 1$ $\overline{Y_{32}[7]} \vee Y_{33}[7] = 1$	$2^{-3.415}$	$2^{-27.653}$	$\frac{2}{2\times 43} \times 2^{31.068}$
23	Y_{38}	δY_{38} $\delta X_{39}^{\lll 10} \boxplus \delta Y_{38}^{\lll 10} = 0$	$2^{-5.719}$	$2^{-21.934}$	$\frac{5}{2\times 43} \times 2^{27.653}$
24	Y_{39}	δY_{39} $\delta X_{40}^{\lll 10} \boxplus \delta Y_{39}^{\lll 10} = 0$	$2^{-6.237}$	$2^{-15.697}$	$\frac{1}{2\times 43} \times 2^{21.934}$
25	Y_{40}	δY_{40} $\delta X_{41} \boxplus \delta Y_{40}^{\lll 10} = 0$	$2^{-4.622}$	$2^{-11.075}$	$\frac{1}{2\times 43} \times 2^{15.697}$
26	Y_{41}	δY_{41}	$2^{-4.379}$	$2^{-6.696}$	$\frac{1}{2\times 43} \times 2^{11.075}$
27	Y_{42}	δY_{42}	$2^{-6.696}$	-	$\frac{1}{2\times 43} \times 2^{6.696}$
Total	-	-	-	-	$2^{62.24}$

The time complexity of the second stage is $2^{62.24}$

The total time complexity of the whole attack is $2^{62.3}$

† Time complexity is evaluated by one 43-step compression of RIPEMD-160.

Table 25: Colliding message pairs (M_0, M_1) and CV_0 for 41-step RIPEMD-160 using our characteristic in Table 12.

1	CV_0	00a86cd4 f214ff2e af7cf0ba 66322cf7 406fa28a
	M_0	22fd841e 3af4f9cc f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 2fce0222
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d108c 8c415f74 970b8ed5 4ca0cc22
	M_1	22fd841e 3af4f9cc f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 2fce0222
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d908c 8c415f74 970b8ed5 4ca0cc22
	hash	8199dda9 2eec60cf ebb6061f c951137a 3b231d4a
2	CV_0	c8ee9f0a 95e9cde3 24853364 273f96ef ac0b149d
	M_0	22fd841e eba5f97d f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 224c4cf7
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d108c 9b9c439b 970b8ed5 4ca0cc22
	M_1	22fd841e eba5f97d f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 224c4cf7
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d908c 9b9c439b 970b8ed5 4ca0cc22
	hash	28331db2 2ee74bcd 7f4afb11 ec2250b2 794cdf88
3	CV_0	aca9a75f 78347e77 2b70e72c d31cf83f 9117c91b
	M_0	22fd841e 0bd4d5ef f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 638d4bbd
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d108c 00aae277 970b8ed5 4ca0cc22
	M_1	22fd841e 0bd4d5ef f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 638d4bbd
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d908c 00aae277 970b8ed5 4ca0cc22
	hash	6914c185 494e7094 cdd869e2 66332a41 c504befb
4	CV_0	fb64d063 87fe05cb 4c458e87 53498553 1f85072a
	M_0	22fd841e 37ecc555 f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 79795d7d
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d108c 9c5aaa53 970b8ed5 4ca0cc22
	M_1	22fd841e 37ecc555 f1fc8a55 7768fef9 2f880f34 b928267f f77922a3 79795d7d
		c4b57f6d f8b8c9df 2f1d6959 8257bcc0 427d908c 9c5aaa53 970b8ed5 4ca0cc22
	hash	e78ef361 8966fed8 a849832d f64ae097 1f90b879

Table 26: Colliding message pairs (M_0, M_1) and CV_0 for 42-step RIPEMD-160 using our characteristic in Table 13.

CV_0	e4780cd1 38f312e0 b4d5b752 91ff78d5 345d75ee
M_0	79f928a9 6977d62e 8353ba65 e2e8af98 06a68efd 4fe043f5 efccef7e 99cee9a0
	c1703e5e 64ee7ec9 ae7ad7b5 078d7695 6ceb3df9 cce8d3f5 5001508f eaff152f
M_1	79f928a9 6977d62e 8353ba65 e2e8af98 06a68efd 4fe043f5 efccef7e 99cee9a0
	c1703e5e 64ee7ec9 ae7ad7b5 078d7695 6cebbdf9 cce8d3f5 5001508f eaff152f
hash	950593df d7ac8fad bfc5bcc5 f3a4496e 3427911e

Table 27: Colliding message pairs (M_0, M_1) and CV_0 for 41-step RIPEMD-160 using characteristic from [9].

CV_0	df20b559 45d1ec9f b3c30a61 3c3c66d5 8bcff5c7
M_0	c408cf53 1e1f5ecf d523de93 8ba64003 e5b58a85 b99d6243 54e6af34 fc2ca034
	27c09f39 e8300da6 a4fff494 3efb77d4 04adaef9 b61cacd2 8ae1c0fa bc002a65
M_1	c408cf53 1e1f5ecf d523de93 8ba64003 e5b58a85 b99d6243 54e6af34 fc2ca034
	27c09f39 e8300da6 a4fff494 3efb77d4 04ae2ef9 b61cacd2 8ae1c0fa bc002a65
hash	a0f35e5f e483b17e 82ab7372 f00df6c9 d72a0a94

J 48-Step Attacks from the Middle

In the following, we detail our improved 48-step semi-free-start (SFS) collision attack on RIPEMD-160 and discuss the limitations that prevent extending the attack to more steps.

J.1 Search for a New 48-Step Characteristic

At ASIACRYPT 2013, Mendel et al. [17] selected $m_7 = 2^{31}$ as the message word for the single difference insertion. the shape of the differential characteristic is shown in Figure 8, which the local collision spans from X_{37} to X_{57} in left branch and from Y_{32} to Y_{60} in right branch. Then they employed a guess-and-determine (GnD) technique to construct a 48-step differential characteristic for RIPEMD-160, covering steps 16 to 63. However, the GnD technique requires pre-fixing the differential constraint on M_7, i.e., specifying the bit position(s) of the difference. Therefore, the original 48-step differential characteristic with $m_7 = 2^{31}$ is likely not optimal.

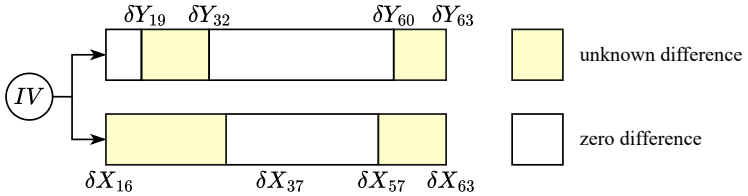


Fig. 8: Attack 48 steps of RIPEMD-160 by injecting difference at m_7.

We set all 32 bits of m_7 as free variables to search for a new 48-step differential characteristic. The detailed procedure is as follows:

Step 1: Set the 32 bits of m_7 as free variables and search for an optimal solution of $(\nabla X_{30}, \ldots, \nabla X_{63})$ such that $\delta X_i = 0$ for $37 \leq i \leq 57$. The objective function is to minimize $\sum_{i=35}^{i=63} \mathbf{H}(\nabla X_i) + \sum_{i=35}^{i=63} \sum_{j=0}^{j=31} (t1_j^i + t2_j^i) + \sum_{i=35}^{i=63} (W_0^i + W_1^i)$. This step also determines the specific value of m_7.

Step 2: Find a valid solution of $(\nabla X_{11}, \ldots, \nabla X_{25})$ such that $\delta X_i = 0$ for $11 \leq i \leq 15$ and minimize $\sum_{i=16}^{i=22} \mathbf{H}(\nabla X_i) + \sum_{i=16}^{i=25} \sum_{j=0}^{j=31} (t1_j^i + t2_j^i) + \sum_{i=16}^{i=22} (W_0^i + W_1^i)$.

Step 3: Determine a solution of $(\nabla X_{26}, \ldots, \nabla X_{29})$ such that the two partial solutions $(\nabla X_{11}, \ldots, \nabla X_{25})$ and $(\nabla X_{30}, \ldots, \nabla X_{63})$ can be connected.

Step 4: Find an optimal solution of $(\nabla Y_{23}, \ldots, \nabla Y_{63})$ such that $\delta Y_i = 0$ for $32 \leq i \leq 60$, and the differences $(\nabla Y_{59}, \ldots, \nabla Y_{63})$ cancel with $(\nabla X_{59}, \ldots, \nabla X_{63})$. The objective function is to minimize $\sum_{i=28}^{i=63} \mathbf{H}(\nabla X_i) + \sum_{i=28}^{i=63} \sum_{j=0}^{j=31} (t1_j^i + t2_j^i) + \sum_{i=28}^{i=63} (W_0^i + W_1^i)$.

Step 5: Find a valid solution of $(\nabla Y_{11}, \ldots, \nabla Y_{22})$ such that it can propagate to $\nabla Y_{23}, \ldots, \nabla Y_{63}$.

The new 48-step differential characteristic and the corresponding implicit conditions on (LQ_i, RQ_i) are displayed in Table 28 and Table 29, respectively. Notably, δm_7 is now set to 2^{16}, instead of the previously used value 2^{31}.

J.2 Apply Differential Clustering

In this attack, we consider the differential clustering effect only in the left branch. Specifically, we randomly select 2^{24} valid $(X_{30}, X_{31}, X_{32}, X_{33}, X_{34})$ along with the corresponding message words involved in the subsequent computations and repeat this experiment 200 times. We then evaluate the probabilities that $(\delta X_{59}, \delta X_{60}, \delta X_{61}, \delta X_{62}, \delta X_{63})$ are satisfied. The probabilities are $(2^{-9.542}, 2^{-1.222}, 2^{-1.099}, 2^{-0.094}, 2^{-0.998})$, respectively.

Since only $\delta X_{62} \neq 0$, it is sufficient to ensure that $\delta X_{62} \boxplus (\delta Y_{61} \lll 10) = 0$, so that the differences in the left and right branches cancel each other. As differential clustering is not applied to the right branch, this cancellation holds with probability 1, without introducing additional cancellation costs as in previous attacks.

J.3 Perform Message Modification Process

Mendel et al. [17] and Wang et al. [21] both adopted a four-stage message modification framework, which we summarize into a two-stage framework as illustrated in Figure 9. Specifically, in Stage 1 (corresponding to Part 1), we generate a sufficient number of starting points. Then, we divide the remaining uncontrolled conditions into three independent parts. By applying our dedicated message modification technique, we minimize the overall attack complexity.

In what follows, we provide a detailed description of the attack process.

The first stage: generate starting points. The target of this stage is to find sufficient starting points, which are the solutions for $(X_{12}, \ldots, X_{34})$ and $(Y_{12}, \ldots, Y_{27})$. We employ an effective three-phase strategy:

Phase 1: Find a solution for $(X_{15}, \ldots, X_{34})$ and $Y_{16}, \ldots, Y_{27}$ using the model that describes value transitions [13, 19].

Table 28: The 48-step differential characteristic, where $\delta m_7 = 2^{16}$.

i	∇X_i	$\pi_l(i)$	i	∇Y_i	$\pi_r(i)$
11	================================		11	================================	
12	================================		12	================================	
13	================================		13	================================	
14	================================		14	================================	
15	================================		15	================================	
16	========n============================	7	16	================================	6
17	========0============================	4	17	==================1=============	11
18	========1=====================1=	13	18	0000010==========0===0010001101	3
19	================================0=	1	19	10100==1n0=011100=10110=0010=010	7
20	================================n=	10	20	==0=1=010=01uuuuuuuuuuuuuuuu00=	0
21	================================0=	6	21	0nnnnnn0unnu0u=001un00111=nuuuuu	13
22	====================1===========	15	22	uuu01nuu1nuunu=n01nuu111n11u011u	5
23	==================1=============	3	23	0nn101nuu10111un1nu1n1u1n1n0u00u	10
24	=0=1====0==========n===========	12	24	n1100nnnnn1nn1111n0nn01nnn0uu111	14
25	0======1n0=1========0011=00=====	0	25	unnu0u=00un=nn0nnn10=0000=nu0000	15
26	10======1=0001=100===01unn==0001	9	26	=010n==0=00==100=110n=10n1=01nn1	8
27	0n===0111==unnn1u11011u111=10nuu	5	27	==1u1====u0==u0==1=u==100u01=0=1	12
28	0u00=nuuuuuu01u110u1n0u11=101111	2	28	n==1n=====0===0====0=====0n==0==	4
29	1uuu1u=uun0unu1=0nuuuuuuu1=unnn0	14	29	===1==============u=====u======	9
30	nn0u0n=1un1101unnu0111u=1=nnnnnn	11	30	====n====================0======	1
31	10=nnn101111011011=1110=01n1n0u0	8	31	====1====================u======	2
32	===u1100====u=u0n01=00u==u111=0=	3	32	=========================1======	15
33	===n=n=u====0=0=0=0===0==10=====	10	33	================================	5
34	===1=0=0==============u==u=0=0==	14	34	================================	1
35	=======n==============1==1======	4	35	================================	3
36	=======1================u======	9	36	================================	7
37	========================1======	15	37	================================	14
38	================================	8	38	================================	6
39	================================	1	39	================================	9
40	================================	2	40	================================	11
41	================================	7	41	================================	8
42	================================	0	42	================================	12
43	================================	6	43	================================	2
44	================================	13	44	================================	10
45	================================	11	45	================================	0
46	================================	5	46	================================	4
47	================================	12	47	================================	13
48	================================	1	48	================================	8
49	================================	9	49	================================	6
50	================================	11	50	================================	4
51	================================	10	51	================================	1
52	================================	0	52	================================	3
53	================================	8	53	================================	11
54	================================	12	54	================================	15
55	================================	4	55	================================	0
56	====================0===========	13	56	================================	5
57	====================1===========	3	57	================================	12
58	==========n====================	7	58	================================	2
59	================================	15	59	================================	13
60	==========1====================	14	60	================================	9
61	================================	5	61	==========n====================	7
62	n===============================	6	62	==========0====================	10
63	================================	2	63	================================	14

$X_{59}[31] = X_{60}[31]$,
$\overline{X_{35}[16]} \vee X_{36}[16] = 1, \overline{X_{35}[19]} \vee X_{36}[19] = 1, \overline{X_{36}[2]} \vee X_{37}[2] = 1, \overline{X_{37}[16]} \vee X_{38}[16] = 1$,
$Y_{27}[11] = Y_{28}[11], Y_{28}[0, 16, 22] = Y_{29}[0, 16, 22], Y_{29}[5, 9, 15] = Y_{30}[5, 9, 15]$,
$\overline{Y_{30}[16]} \vee Y_{31}[16] = 1, \overline{Y_{30}[22]} \vee Y_{31}[22] = 1, \overline{Y_{31}[5]} \vee Y_{32}[5] = 1, \overline{Y_{32}[16]} \vee Y_{33}[16] = 1$,
$Y_{59}[11] = Y_{60}[11]$

Table 29: Implicit conditions on LQ_i and RQ_i for 48-step differential characteristic.

Implicit conditions on LQ_i and RQ_i:
$(LQ_i \boxplus \alpha_i^l)^{\lll s_i^l} = LQ_i^{\lll s_i^l} \boxplus \beta_i^l$
$(RQ_i \boxplus \alpha_i^r)^{\lll s_i^r} = RQ_i^{\lll s_i^r} \boxplus \beta_i^r$

i	α_i^l	β_i^l	s_i^l	i	α_i^r	β_i^r	s_i^r
16	0x10000	0x800000	7	16	0x0	0x0	9
17	0x0	0x0	6	17	0x0	0x0	13
18	0x0	0x0	8	18	0x0	0x0	15
19	0x0	0x0	13	19	0x10000	0x800000	7
20	0x0	0x0	11	20	0x800000	0x8	12
21	0x0	0x0	9	21	0x7dcbf0	0x7dcbf001	8
22	0x0	0x0	7	22	0x37908a84	0x2115086f	9
23	0x0	0x0	15	23	0xd2ac0fe5	0x607f2695	11
24	0x0	0x0	7	24	0x510fb073	0x87d839a8	7
25	0x800	0x800000	12	25	0x33385b74	0x9c2dba19	7
26	0xff800000	0xfffffffc0	15	26	0xb3de5	0xb3de5001	12
27	0x9ffec0	0x3ffd7e01	9	27	0x7be64324	0xf321923e	7
28	0xbff7fdbd	0xbfede600	11	28	0x49a6578	0x26995e01	6
29	0xf9156801	0x8ab4007c	7	29	0x612291ff	0x48ffb091	15
30	0xf1f59e02	0xb3c03e3f	13	30	0x3f003eef	0x7dde7e0	13
31	0xb262607f	0x2607fb26	12	31	0x220820	0x10410001	11
32	0xd7f70bd0	0xb85e7ebf	11	32	0xefffffbf	0xffff7de0	9
33	0x4ea217f0	0x42fe09d5	13	33	0x8200	0x410000	7
34	0xc40c1bfb	0x306fef1	6	34	0xffc00000	0xfffffffe0	15
35	0x2001fecf	0xff6790	7	35	0x20	0x10000	11
36	0x9824	0x26090000	14	36	0x0	0x0	8
37	0xda000000	0xfffffffb4	9	37	0x0	0x0	6
38	0x48	0x90000	13	38	0x0	0x0	6
39	0xfff80000	0xffffffffc	15	39	0x0	0x0	14
40	0x4	0x10000	14	40	0x0	0x0	12
41	0x0	0x0	8	41	0x0	0x0	13
42	0x0	0x0	13	42	0x0	0x0	5
43	0x0	0x0	6	43	0x0	0x0	14
44	0x0	0x0	5	44	0x0	0x0	13
45	0x0	0x0	12	45	0x0	0x0	13
46	0x0	0x0	7	46	0x0	0x0	7
47	0x0	0x0	5	47	0x0	0x0	5
48	0x0	0x0	11	48	0x0	0x0	15
49	0x0	0x0	12	49	0x0	0x0	5
50	0x0	0x0	14	50	0x0	0x0	8
51	0x0	0x0	15	51	0x0	0x0	11
52	0x0	0x0	14	52	0x0	0x0	14
53	0x0	0x0	15	53	0x0	0x0	14
54	0x0	0x0	9	54	0x0	0x0	6
55	0x0	0x0	8	55	0x0	0x0	14
56	0x0	0x0	9	56	0x0	0x0	6
57	0x0	0x0	14	57	0x0	0x0	9
58	0x10000	0x200000	5	58	0x0	0x0	12
59	0x0	0x0	6	59	0x0	0x0	9
60	0x0	0x0	8	60	0x0	0x0	12
61	0x0	0x0	6	61	0x10000	0x200000	5
62	0x0	0x0	5	62	0x0	0x0	15
63	0x0	0x0	12	63	0x0	0x0	8

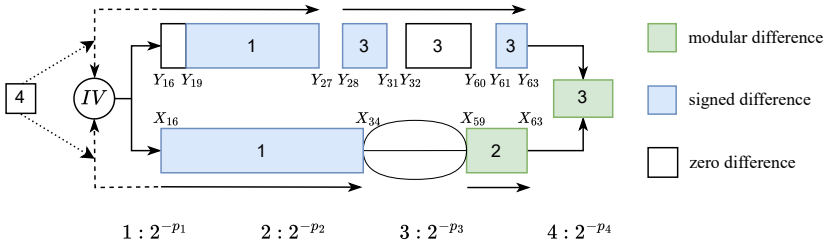


Fig. 9: Message modification framework for 48-step SFS collision attack.

Phase 2: In Phase 1, (m_1, m_4, m_7, m_{13}) remain undetermined. Similar to [17, 21], we use these free message words to ensure a perfect match on the values of the four initial words of both branches, i.e., $X_i = Y_i$ for $i = 12, 13, 14, 15$. In this phase, to ensure $X_{13} = Y_{13}$ and $X_{14} = Y_{14}$, we pre-compute m_1 and m_7 and store them in a table for all the 2^{64} possible values, requiring 2^{73} time and 2^{64} memory. Each table lookup costs about 4 steps of the step function and $2^{-2.1}$ calls of the 48-step RIPEMD-160. The costs of $X_{15} = Y_{15}$ and $X_{12} = Y_{13}$ are both estimated as a single step of the step function.

Phase 3: Phases 1 and 2 yield a solution of $(X_{12}, \ldots, X_{34})$ and $(Y_{12}, \ldots, Y_{27})$, which serves as the initial starting point. We then use the available degrees of freedom in (m_0, m_6, m_9) to generate sufficient starting points. First, we exhaust all possible values of X_{21} and recompute m_9 to ensure that X_{26} remains unchanged:

$$m_9 = (X_{26} \boxminus X_{22}^{\lll 10})^{\ggg s_{26}^l} \boxminus IFX(X_{25}, X_{24}, X_{23}^{\lll 10}) \boxminus X_{21}^{\lll 10} \boxminus K_1^l.$$

Since there is only one bit condition on X_{21}, we can generate 2^{31} starting points. We then traverse all possible values of m_0 and modify $(X_{20}, X_{19}, X_{18}, X_{17})$ to keep $(X_{25}, X_{24}, X_{23}, X_{22})$ unchanged. There are 5 conditions on $(X_{20}, X_{19}, X_{18}, X_{17})$, allowing us to generate 2^{27} starting points:

$$X_{20} = ((X_{25} \boxminus X_{21}^{\lll 10})^{\ggg s_{25}^l} \boxminus IFX(X_{24}, X_{23}, X_{22}^{\lll 10}) \boxminus m_0 \boxminus K_1^l)^{\ggg 10},$$
$$X_{19} = ((X_{24} \boxminus X_{20}^{\lll 10})^{\ggg s_{24}^l} \boxminus IFX(X_{23}, X_{22}, X_{21}^{\lll 10}) \boxminus m_{12} \boxminus K_1^l)^{\ggg 10},$$
$$X_{18} = ((X_{23} \boxminus X_{19}^{\lll 10})^{\ggg s_{23}^l} \boxminus IFX(X_{22}, X_{21}, X_{20}^{\lll 10}) \boxminus m_3 \boxminus K_1^l)^{\ggg 10},$$
$$X_{17} = ((X_{22} \boxminus X_{18}^{\lll 10})^{\ggg s_{22}^l} \boxminus IFX(X_{21}, X_{20}, X_{19}^{\lll 10}) \boxminus m_{15} \boxminus K_1^l)^{\ggg 10}.$$

Finally, we exhaust all possible values of X_{16} and recompute m_6 and X_{15} to keep X_{21} unchanged:

$$m_6 = (X_{21} \boxminus X_{17}^{\lll 10})^{\ggg s_{21}^l} \boxminus IFX(X_{20}, X_{19}, X_{18}^{\lll 10}) \boxminus X_{16}^{\lll 10} \boxminus K_1^l,$$
$$X_{15} = ((X_{20} \boxminus X_{16}^{\lll 10})^{\ggg s_{20}^l} \boxminus IFX(X_{19}, X_{18}, X_{17}^{\lll 10}) \boxminus m_{10} \boxminus K_1^l)^{\ggg 10}.$$

Similarly, the condition $X_{26}[23] = 0$ allows us to generate 2^{31} valid starting points. We then utilize the degrees of freedom provided by (m_1, m_4, m_7, m_{13}) to deterministically merge the initial states $(X_{12}, \ldots, X_{15})$ and $(Y_{12}, \ldots, Y_{15})$.

The second stage: validate the uncontrolled conditions. During the first stage, all message words are fully determined. Therefore, in this stage, we verify the remaining conditions, including $(\delta X_{59}, \ldots, \delta X_{63})$ (Part 2), $(\nabla Y_{28}, \ldots, \nabla Y_{63}, RQ_{28}, \ldots, RQ_{63})$ (Part 3), and $X_{11} = Y_{11}$ (Part 4). All these conditions hold with probability $2^{-(p_2+p_3+p_4)}$.

Complexity Evaluation. According to Table 28 and Table 29 and above analysis, we have

$$p_2 = 12.955,\ p_3 = 30.636,\ p_4 = 32.$$

For the first stage, the total number of starting points required is $2^{p_2+p_3+p_4} = 2^{75.591}$. The overall time complexity of the first stage can be estimated as

$$\begin{aligned} T_1 =& 2^{75.591} \times \frac{2}{2 \times 48} + 2^{75.591-31} \times \frac{4}{2 \times 48} \times 2^5 + 2^{75.591-31-27} \times \frac{1}{2 \times 48} + \\ & 2^{75.591} \times (\frac{1}{2 \times 48} + \frac{4}{2 \times 48} + 2^{-2.1} + \frac{1}{2 \times 48}) + 2^{73} \approx 2^{74.54}. \end{aligned}$$

To optimize the time complexity of the second stage, we utilize our automated tool to determine the optimal verification order. The detailed complexity of this stage is presented in Table 30.

Consequently, the total time complexity of the 48-step SFS collision attack on RIPEMD-160 is approximately $2^{74.54} + 2^{69.00} \approx 2^{74.6}$. In comparison, the complexity given in [21] is estimated as $2^{78.5} \times 2^{-2.1}$, where $2^{-78.5}$ corresponds to the uncontrolled probability. We note that this estimation primarily reflects the differential probability, and may not fully account for the actual cost associated with the message modification process. In contrast, our analysis offers a more precise estimation by explicitly considering all relevant factors, leading to a refined total complexity from the previously reported $2^{76.4}$ to a more accurate value of $2^{77.0}$.

Challenges in Extending the Attack to More Steps. First, long-step local collisions have already been established in the middle part of both the left and right branches. In the final steps of the characteristic, once message differences are introduced, the differences propagate purely linearly. To ensure successful cancellation between the two branches, step 63 is already the furthest step to which the characteristic can be extended in the forward direction. When attempting to extend it to step 64, collisions can no longer be guaranteed regardless of the message differences introduced.

Moreover, only the degrees of freedom provided by (m_1, m_4, m_7, m_{13}) are available to ensure a match on the initial state words $X_i = Y_i$ for $12 \leq i \leq 15$.

The remaining word $X_{11} = Y_{11}$ falls under uncontrolled conditions and contributes a probability factor of 2^{-32}. Therefore, extending the characteristic further backward to step 10 would introduce an additional probability factor of 2^{-32}, rendering the attack infeasible unless a more effective method for merging the two branches is discovered.

In addition, a similar bottleneck arises with the injection of other message word differences discussed by Mendel et al. [17], particularly when attempting to achieve results better than those obtained in 48 steps.

Table 30: Complexity computation for the Second Stage of the 48-Step Attack.

The probability to be verified is $2^{-p_2-p_3-p_4} = 2^{-75.591}$ in total					
Step	**Computed States**	**Verify Conditions**	**Verify Probabilities**	**Remaining Probabilities**	**Time Complexity†**
1	Y_{28}	$Y_{28}[28] = 1, RQ_{28}$ $Y_{28}[11] = Y_{27}[11]$ $Y_{28}[2,5,6,12,17,21,27,31] = 0$	$2^{-10.153}$	$2^{-65.438}$	$\frac{1}{2\times48} \times 2^{75.591}$
2	Y_{29}	$Y_{29}[6,12,28] = 1, RQ_{29}$ $Y_{29}[0,16,22] = Y_{28}[0,16,22]$	$2^{-7.179}$	$2^{-58.259}$	$\frac{1}{2\times48} \times 2^{65.438}$
3	X_{11}, Y_{11}	$X_{11} = Y_{11}$	2^{-32}	$2^{-26.259}$	$\frac{2}{2\times48} \times 2^{58.259}$
4	Y_{30}, Y_{31} Y_{32}, Y_{33}	$Y_{32}[6] = 1$ $Y_{30}[6,27] = 0$ $Y_{31}[6,27] = 1$ $\overline{Y_{31}[5]} \vee Y_{32}[5] = 1$ $\overline{Y_{30}[16]} \vee Y_{31}[16] = 1$ $\overline{Y_{30}[22]} \vee Y_{31}[22] = 1$ $\overline{Y_{32}[16]} \vee Y_{33}[16] = 1$ $Y_{29}[5,9,15] = Y_{30}[5,9,15]$ $RQ_{30}, RQ_{31}, RQ_{32}, RQ_{33}$	$2^{-10.302}$	$2^{-15.957}$	$\frac{4}{2\times48} \times 2^{26.259}$
5	X_{59}	δX_{59}	$2^{-9.542}$	$2^{-6.415}$	$\frac{25}{2\times48} \times 2^{15.957}$
6	X_{60}	δX_{60}	$2^{-1.222}$	$2^{-5.193}$	$\frac{1}{2\times48} \times 2^{6.415}$
7	X_{61}	δX_{61}	$2^{-1.099}$	$2^{-4.094}$	$\frac{1}{2\times48} \times 2^{5.193}$
8	X_{62}	δX_{62}	$2^{-0.094}$	2^{-4}	$\frac{1}{2\times48} \times 2^{4.094}$
9	X_{63}	δX_{63}	$2^{-0.998}$	$2^{-3.002}$	$\frac{1}{2\times48} \times 2^{4}$
10	Y_{34}	RQ_{34}	$2^{-0.001}$	$2^{-3.001}$	$\frac{1}{2\times48} \times 2^{3.002}$
11	Y_{59}, Y_{60}	$Y_{59}[11] = Y_{60}[21]$	2^{-1}	$2^{-2.001}$	$\frac{26}{2\times48} \times 2^{3.001}$
12	Y_{61}	$Y_{61}[21] = 0$	$2^{-1.001}$	2^{-1}	$\frac{1}{2\times48} \times 2^{2.001}$
13	Y_{62}	$Y_{62}[21] = 0$	2^{-1}	-	$\frac{1}{2\times48} \times 2^{1}$
Total	-	-	-	-	$2^{69.00}$
The total time complexity of the second stage is $2^{69.00}$					

† Time complexity is evaluated by one 48-step compression of RIPEMD-160.