Improved Collision Attack on RIPEMD-160

Zhengrong Lu[1], Hongbo Yu[1,4,5(✉)], Yingxin Li[2], Xindi Zhang[3], and Xiaoen Lin[1]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China lu-zr@mail.tsinghua.edu.cn,yuhongbo@mail.tsinghua.edu.cn, linxe@mail.tsinghua.edu.cn

[2] School of Cryptology, East China Normal University, Shanghai, China liyx1140@163.com

[3] Key Laboratory of System Software (Chinese Academy of Sciences), Institute of Software, Beijing, China dezhangxd96@gmail.com

[4] Zhongguancun Laboratory, Beijing, China

[5] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, China

Abstract. RIPEMD-160 is an ISO/IEC hash function standard based on the Merkle-Damgård structure with a double-branch compression function. There have been many attempts at modular differential attacks on reduced RIPEMD-160, with the best previous result being a 40-step practical collision attack achieved in 2023. That attack constructs a simple local collision in round 2 of the left branch to minimize uncontrolled conditions. To achieve this, differences must be introduced into many message words, which constrains the maximum number of steps that can be attacked. To overcome this limitation and target more steps, we propose a new differential characteristic structure that abandons the sparse local collision in round 2 and instead uses a single continuous differential characteristic spanning rounds 1 to 2 for each branch. This structure allows us to inject a difference into only one message word. Using an automatic search tool based on the high-performance parallel SAT-solver PRS, we identify suitable differential characteristics by imposing more control over conditions, differences, and the probability of proper propagation. Based on the differential characteristics, we identify three colliding message pairs for 42-step RIPEMD-160 with theoretical time complexity of approximately $2^{47.4}$, thereby improving the best practical collision attack by 2 steps on this hash function.

Keywords: RIPEMD-160 · collision attack · modular differential attack.

1 Introduction

In cryptanalysis research, hash functions need to satisfy three fundamental security properties: collision resistance, preimage resistance, and second-preimage resistance. At EUROCRYPT 2005, Wang et al. proposed the modular differential attack strategy [19] and presented practical collision attacks on full MD4

and RIPEMD. This strategy introduced the concept of modular difference, enabling it to trace the propagation of differences through Boolean functions and modular addition more accurately. By constructing suitable modular differential characteristics and using conditions to control the propagation, it can effectively cancel out differences to obtain colliding message pairs. Since then, through improvements in differential characteristic searching and message modification methods to fulfill more conditions, many other hash functions in the MD-SHA family have been broken using this strategy, including MD5 [21], SHA-0 [14, 22], and SHA-1 [18, 20].

In the beginning, differential characteristics were searched manually. However, in complex situations, this approach could neither prove the optimality of the obtained results nor determine whether a valid solution exists. Moreover, hand-crafted work became difficult to apply to hash functions with more complex step functions. At ASIACRYPT 2006, De Canniére [3] first used automatic tools to search for differential characteristics of SHA-1. Since then, more research has focused on automatic differential characteristic searching, including collision attacks on RIPEMD-128, SHA-256, and SHA-512 [6, 8, 15, 16, 24]. Besides these, at SAT 2006, Mironov et al. first applied SAT-solver-based automatic tools to collision attacks on hash functions [17]. At ToSC 2025, Lu et al. utilized a SAT-solver-based automatic tool to find suitable differential characteristics to achieve 46-step and 54-step collision attacks on RIPEMD-128 [13].

RIPEMD [1] is a hash function that follows the Merkle-Damgård construction, and its compression function is double-branch but with both branches sharing the same message permutations, Boolean functions, and rotation constants. Therefore, the same differential characteristic can be applied to both branches to easily obtain a collision attack. Later, RIPEMD-160 [4] was proposed at FSE 1996 as a strengthened version of RIPEMD. As an ISO/IEC international standard, RIPEMD-160 is widely adopted in various cryptographic applications, such as in Bitcoin. In the step functions of RIPEMD-160, message words are introduced into state words through different message permutations and propagate via different Boolean functions in the two branches. This design enhances the security of RIPEMD-160 and poses challenges for collision attacks on this hash function.

Related Works. For collision attacks on RIPEMD-160, automatic searching for differential characteristics is necessary due to its complex step functions. At ASIACRYPT 2017, Liu et al. presented a theoretical collision attack on 30-step RIPEMD-160 [11]. Afterwards, at CRYPTO 2019, they improved their result and achieved a practical collision attack on 31-step RIPEMD-160 and a theoretical collision attack on 34-step RIPEMD-160 [9]. Subsequently, at EUROCRYPT 2023, a 36-step theoretical collision attack [12] was proposed using MILP-based automatic tools for differential characteristic searching. The previous best collision attack result is the 40-step practical collision attack [7] proposed by Li et al. at ToSC 2023.

For semi-free-start collision attacks, the limitation on degrees of freedom is looser; therefore, attacks on more steps exist. In the work where Liu et al. pre-

sented their 30-step theoretical collision attack, they also proposed a theoretical semi-free-start collision attack on 36-step RIPEMD-160 [11]. At ToSC 2019, Liu et al. improved their previous result and demonstrated a practical semi-free-start collision attack on 37-step RIPEMD-160, as well as a theoretical semi-free-start collision attack on 40-step RIPEMD-160 [10]. Afterwards, Li et al. reported theoretical semi-free-start collision attacks on reduced RIPEMD-160 up to 43 steps [7]. The best semi-free-start collision attack to date is a 42-step practical attack and a 44-step theoretical attack [25], proposed by Zhang et al. at ASIACRYPT 2025.

Our Contributions. To achieve improved collision attacks on reduced RIPEMD-160, we adopt a new differential characteristic structure. While this structure allows us to attack more steps, it may increase uncontrolled conditions in round 2. To address this, we employ an automatic search tool based on the high-performance parallel SAT-solver PRS, which enables more precise control over conditions and yields suitable differential characteristics within a reasonable time. Using these characteristics, we mount a practical collision attack on 42-step RIPEMD-160 and successfully identify three colliding message pairs on multi-core computers. This is the first practical collision attack on 42-step RIPEMD-160, matching the step count of the best practical semi-free-start collision attacks while using the more restrictive setting.

Previous (semi-free-start) collision attack results on RIPEMD-160 together with our result, are summarized in Table 1.

Table 1. Summary of (semi-free-start) collision attack results on RIPEMD-160

attack type	steps	time complexity	reference	year
collision	30	2^{67}	[11]	2017
collision	31	practical	[9]	2019
collision	34	$2^{74.3}$	[9]	2019
collision	36	$2^{64.5}$	[12]	2023
collision	40	practical	[7]	2023
collision	**42**	**practical**	**Section 3.3**	**2026**
semi-free-start collision	36	$2^{55.1}$	[11]	2017
semi-free-start collision	37	practical	[10]	2019
semi-free-start collision	40	$2^{74.6}$	[10]	2019
semi-free-start collision	43	$2^{74.8}$	[7]	2023
semi-free-start collision	42	practical	[25]	2025
semi-free-start collision	44	$2^{76.9}$	[25]	2025

Organization. The remainder of this paper is organized as follows. In Section 2, we describe the notations, the algorithm of RIPEMD-160, and some related works. Next, in Section 3, we discuss our attack method, including the new differential characteristic structure and message difference, the search for

differential characteristics, and how to find colliding message pairs. Finally, we conclude our results in Section 4.

2 Preliminaries

2.1 Notations

In this paper, we will use the following notations. $\boxplus$ and $\boxminus$ denote modular addition and subtraction modulo 2^{32}, respectively. $\lll$ and $\ggg$ denote a circular left shift and a circular right shift. $\neg$, $\vee$, $\wedge$, and $\oplus$ represent bitwise NOT, OR, AND, and XOR, respectively, in the Boolean functions. $x[i]$ denotes the i-th bit of x, where $x[0]$ is the least significant bit of x. $x[i \sim j]$ $(i \geq j)$ denotes the value consisting of bits $x[i]$, $x[i-1]$, $\ldots$, $x[j]$. Δx represents the XOR difference of x, i.e., $x \oplus x'$, and δx represents the modular subtraction difference of x, i.e., $x' \boxminus x$. In addition, for the description of conditions in differential characteristics, Table 2 introduces the symbols, where A_i and A'_i represent two colliding state words, and the indices of bits are taken modulo 32. All symbols except 'u' and 'n' indicate no difference on the corresponding bits.

Table 2. Symbols and corresponding conditions

symbol	condition	symbol	condition
.	no other condition	e	$A_i[j] = A_{i-1}[j-10]$
0	$A_i[j] = 0$	f	$A_i[j] \neq A_{i-1}[j-10]$
1	$A_i[j] = 1$	g	$A_i[j] = A_{i-2}[j-10]$
u	$A_i[j] = 1, A'_i[j] = 0$	h	$A_i[j] \neq A_{i-2}[j-10]$
n	$A_i[j] = 0, A'_i[j] = 1$	i	$A_i[j] = A_i[j-10]$
a	$A_i[j] = A_{i-1}[j]$	j	$A_i[j] \neq A_i[j-10]$
b	$A_i[j] \neq A_{i-1}[j]$	k	$A_i[j] = 0$ or $A_{i+1}[j+10] = 1$[1]

[1] The probability equals $\frac{3}{4} \approx 2^{-0.4}$; hence, we regard it as 0.4 conditions.

2.2 Description of RIPEMD-160

RIPEMD-160 [4] is a double-branch hash function based on the Merkle-Damgård structure. The input message is padded and then divided into 512-bit message blocks, denoted as M_i. The padding procedure does not influence the validity of a collision attack; therefore, we will ignore it in the following description. The hash algorithm starts with a 160-bit initial value $CV_0 = IV$ and iteratively applies the compression function $CV_{i+1} = H(CV_i, M_i)$. The constant initial value IV is as follows:

IV=(0x67452301,0xEFCDAB89,0x98BADCFE,0x10325476,0xC3D2E1F0).

The compression function $H(CV, M)$ of RIPEMD-160 can be divided into two branches, each consisting of 80 steps with every 16 steps forming one round. To describe the state words of both branches, we denote X_i as the state word of the left branch and Y_i as the state word of the right branch. In addition, Q_i^x and Q_i^y are two important intermediate values in the step function.

The input message block M can be divided into 16 32-bit message words $(m_0, \ldots, m_{15})$. The input CV of the compression function consists of 5 32-bit words $(cv_0, \ldots, cv_4)$. The initial state words for both branches are defined as follows:

$$\begin{aligned} X_{-5} = Y_{-5} &= (cv_0 \ggg 10), \\ X_{-4} = Y_{-4} &= (cv_4 \ggg 10), \\ X_{-3} = Y_{-3} &= (cv_3 \ggg 10), \\ X_{-2} = Y_{-2} &= cv_2, \\ X_{-1} = Y_{-1} &= cv_1. \end{aligned}$$

The step function at step i of both branches is shown as follows, where $i = 0, 1, \ldots, 79$.

$$\begin{aligned} Q_i^x &= X_{i-5} \lll 10 \boxplus f_i^x(X_{i-1}, X_{i-2}, X_{i-3} \lll 10) \boxplus m_{\pi_i^x} \boxplus K_i^x, \\ X_i &= Q_i^x \lll s_i^x \boxplus X_{i-4} \lll 10, \\ Q_i^y &= Y_{i-5} \lll 10 \boxplus f_i^y(Y_{i-1}, Y_{i-2}, Y_{i-3} \lll 10) \boxplus m_{\pi_i^y} \boxplus K_i^y, \\ Y_i &= Q_i^y \lll s_i^y \boxplus Y_{i-4} \lll 10. \end{aligned}$$

The structure of the left branch step function is illustrated in Fig. 1:

The Boolean functions f_i^x and f_i^y, along with the round constants K_i^x and K_i^y, are listed in Table 3. The message permutation constants π_i^x and π_i^y, as well as the rotation constants s_i^x and s_i^y used in the step functions, are listed in Table 4.

Table 3. Boolean functions and round constants in RIPEMD-160

round	step i	$f_i^x(a,b,c)$	$f_i^y(a,b,c)$	K_i^x	K_i^y
1	$0 \le i \le 15$	$a \oplus b \oplus c$	$a \oplus (b \vee \neg c)$	0x00000000	0x50A28BE6
2	$16 \le i \le 31$	$(a \wedge b) \vee (\neg a \wedge c)$	$(c \wedge a) \vee (\neg c \wedge b)$	0x5A827999	0x5C4DD124
3	$32 \le i \le 47$	$(a \vee \neg b) \oplus c$	$(a \vee \neg b) \oplus c$	0x6ED9EBA1	0x6D703EF3
4	$48 \le i \le 63$	$(c \wedge a) \vee (\neg c \wedge b)$	$(a \wedge b) \vee (\neg a \wedge c)$	0x8F1BBCDC	0x7A6D76E9
5	$64 \le i \le 79$	$a \oplus (b \vee \neg c)$	$a \oplus b \oplus c$	0xA953FD4E	0x00000000

Table 4. Message permutation constants and rotation constants in RIPEMD-160

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

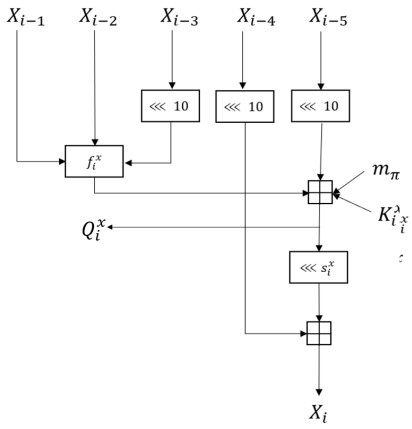

Fig. 1. left branch step function of RIPEMD-160

After all step functions, the output of the compress function $(H_0, H_1, H_2, H_3, H_4)$ is obtained by feed-forward expressions:

$$
\begin{aligned}
H_0 &= cv_1 \boxplus X_{78} \boxplus Y_{77} \lll 10,\\
H_1 &= cv_2 \boxplus X_{77} \lll 10 \boxplus Y_{76} \lll 10,\\
H_2 &= cv_3 \boxplus X_{76} \lll 10 \boxplus Y_{75} \lll 10,\\
H_3 &= cv_4 \boxplus X_{75} \lll 10 \boxplus Y_{79},\\
H_4 &= cv_0 \boxplus X_{79} \boxplus Y_{78}.
\end{aligned}
$$

2.3 SAT-solver-based Automatic Search Tool

Due to the double-branch structure and complex step functions of RIPEMD-160, the evaluation of differential characteristics involves many aspects, such as the degrees of freedom of message words across both branches and the extra conditions on intermediate values. Therefore, we choose to employ a SAT-solver-based automatic search tool to find suitable differential characteristics, allowing us to take into account more factors that influence the time complexity and thereby ensure a valid collision attack.

In addition, if we optimize the sparse part of the differential characteristics, the dense part may contain a large number of conditions, making it difficult to find corresponding message words and state words due to the lack of degrees of freedom. To address this issue, we also employ an automatic search tool to obtain partial message word and state word solutions that satisfy all conditions in the dense part.

The SAT-solver used in this work is based on PRS [2, 23], a high-performance parallel SAT-solver designed for both shared-memory and distributed parallel computing environments.

PRS adopts a Master-Worker architecture, where the Master is responsible for task scheduling and coordination, while the Workers perform the actual SAT solving. It follows the idea of diversity portfolio parallel search, in which multiple Workers solve the same SAT instance in parallel. Different Workers use complementary solver configurations, such as different branching heuristics and restart strategies, so that they explore the search space from different directions. Once any Worker finds a satisfying assignment or proves the instance unsatisfiable, the original SAT problem is solved.

To reduce redundant exploration of the same search space, PRS allows Workers to exchange selected learned clauses. During the search, when a Worker reaches a conflict, it analyzes the conflict and derives a new clause that rules out a class of assignments leading to the same conflict. Sharing such clauses enables other Workers to avoid revisiting similar infeasible regions, thereby propagating useful conflict information across Workers and pruning the subsequent search. By combining diversity portfolio parallel search with learned-clause sharing, PRS can effectively exploit multi-core and distributed computing resources to improve the solving efficiency of large-scale SAT instances.

In recent years, several top-performing solvers in the parallel track of the International SAT Competition have adopted frameworks similar to the PRS series, some of them were directly developed based on PRS or its subsequent variants.

We use a recent derivative of PRS, available at `https://anonymous.4open.science/r/PRS-sc26/`. This version incorporates a broader set of complementary techniques and parameter configurations, aiming to improve the diversity of the underlying portfolio and achieve better overall solving performance.

2.4 Difference Propagation of Intermediate Values

For RIPEMD-160, we not only need to consider the modular differences of state words δX_i and δY_i, but also focus on the difference propagation of intermediate values Q_i^x and Q_i^y. Take the left branch as an example: when δX_{i-5}, δX_{i-3}, δX_{i-2}, and δX_{i-1} are determined, we can derive δQ_i^x from them; similarly, when δX_{i-4} and δX_i are determined, we can derive $\delta(Q_i^x \lll s_i^x)$. Thus, we can summarize the form of the requirement for intermediate values, as follows:

$$(Q \boxplus \alpha) \lll s = Q \lll s \boxplus \beta. \tag{1}$$

where Q represents the intermediate value, α and β represent the input and output modular differences respectively.

In 2007, Fouque et al. [5] analyzed the modular difference propagation of circular rotation. For a fixed α, they identified all possible cases of β along with their corresponding conditions. Furthermore, in 2017, Liu et al. [11] adopted this analysis for collision attacks on RIPEMD-160 and provided a more specific method to calculate the probability and conditions for correct difference propagation. Here, we will describe their method and conclusions, and then present a simple way to ensure sufficiently high probability that is suitable for SAT-solver-based automatic search tools.

First, we split $(Q \boxplus \alpha)$ into two parts $A\|B$,and $(Q \lll s \boxplus \beta)$ into two parts $B'\|A'$, where A and A' are s-bit variables, B and B' are $(32-s)$-bit variables. From Equation (1), we know that $A = A'$ and $B = B'$.

Next, we will derive the specific expressions of A, A', B, and B'. From their definitions and after eliminating the common parts Q, we obtain

$$A - A' = (\alpha[31 \sim (32-s)] - \beta[(s-1) \sim 0] + carry_0) \mod 2^s = 0, \quad (2)$$

$$B' - B = (\beta[31 \sim s] - \alpha[(32-s-1) \sim 0] + carry_1) \mod 2^{32-s} = 0. \quad (3)$$

where $carry_0$ represents the carry from bit $(Q \boxplus \alpha)[32-s-1]$ to bit $(Q \boxplus \alpha)[32-s]$, and $carry_1$ represents the carry from bit $(Q \lll s \boxplus \beta)[s-1]$ to bit $(Q \lll s \boxplus \beta)[s]$.

From Equations (2) and (3), we know that when $\alpha[(32-s-1) \sim 0]$ and $\beta[(s-1) \sim 0]$ are determined, $\beta[31 \sim s]$ and $\alpha[31 \sim (32-s)]$ each have only two possible values, depending solely on $carry_0$ and $carry_1$ respectively. We denote κ as the event that $carry_0 = 1$, and ω as the event that $carry_1 = 1$. Then the probabilities $P(\kappa)$ and $P(\omega)$ can be calculated as follows:

$$P(\kappa) = \sum_{i=0}^{32-s-1} \alpha[i] \times 2^{-(32-s-i)},$$

$$P(\omega) = \sum_{i=0}^{s-1} \beta[i] \times 2^{-(s-i)}.$$

For a given differential characteristic, we can derive α, β, $carry_0$, and $carry_1$ to calculate the probability of proper difference propagation. However, when searching for differential characteristics using SAT-solver-based automatic search tools, it is difficult to model this probability because it depends on numerous bits of the input and output modular differences. Therefore, we present Property 1 to identify sufficiently high probability:

Property 1. Given α, β, $carry_0$, and $carry_1$, for $0 \le t < 32-s$, if there exists a valid Q such that $Q[t] = (Q \boxplus \alpha)[t]$, then $P(A = A') \ge 2^{-(32-s-t)}$; for $0 \le t < s$, if there exists a valid Q such that $(Q \lll s)[t] = (Q \lll s \boxplus \beta)[t]$, then $P(B = B') \ge 2^{-(s-t)}$.

Proof. We employ proof by contradiction. For the first part, assume $P(A = A') < 2^{-(32-s-t)}$. When $carry_0 = 0$, we obtain $P(\kappa) \ge 1 - 2^{-(32-s-t)}$; consequently, all bits in $\alpha[(32-s-1) \sim t]$ are equal to 1. In this case, for a valid Q, the conditions $Q[t] = 0$ and $(Q \boxplus \alpha)[t] = 1$ always hold, which contradicts the premise $Q[t] = (Q \boxplus \alpha)[t]$. When $carry_0 = 1$, we obtain $P(\kappa) < 2^{-(32-s-t)}$; consequently, all bits in $\alpha[(32-s-1) \sim t]$ are equal to 0. In this case, for a valid Q, the conditions $Q[t] = 1$ and $(Q \boxplus \alpha)[t] = 0$ always hold, again leading to a contradiction. The proof for the second part $P(B = B')$ can be completed in a similar manner.

By Property 1, a single condition on each of ΔQ and $\Delta(Q \lll s)$ suffices to control the probability of proper difference propagation of intermediate values.

3 Improved Collision Attack on Reduced RIPEMD-160

3.1 Message Differences and Differential Characteristic Structures

First, we review the structure of previous collision attacks. The 36-step collision attack [12] introduces differences into message words m_0, m_6, and m_9 to avoid state word difference in round 3, and obtains a sparse local collision in round 2 of the left branch with few conditions. The attack then searches for suitable differential characteristics in round 1 of the left branch and in rounds 1 and 2 of the right branch to obtain the final collision. The corresponding differential characteristic structure is illustrated in Fig. 2, where each curve represents an independent local collision.

The previous best result [7] presents a 40-step collision attack using a similar strategy. It introduces differences into message words m_0, m_2, m_{11}, and m_{12}, which appear later in round 3, enabling the attack to cover more steps. The differential characteristic structure of this attack is shown in Fig. 3.

We summarize the common structure of the differential characteristics employed in these attacks as follows:

Round 3 contains no state word differences. The left branch contains two local collisions (one in round 1 and another in round 2), with the later local collision being as simple as possible. The right branch contains a single local collision spanning rounds 1 and 2. Since the time complexity of the collision attack mainly depends on uncontrolled conditions in round 2, and this structure contains few such conditions in the left branch due to its simple local collision, a feasible collision attack can be achieved based on this structure.

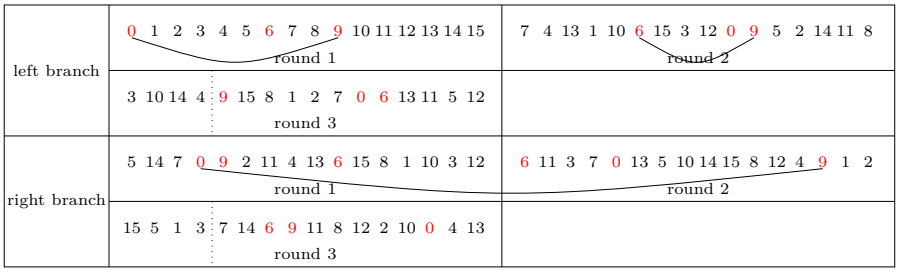

Fig. 2. Differential characteristic structure for 36-step RIPEMD-160

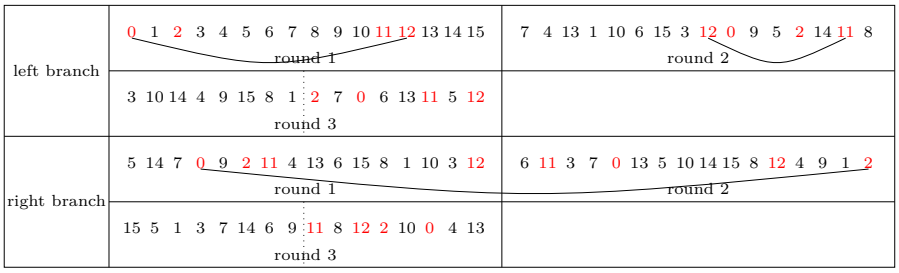

Fig. 3. Differential characteristic structure for 40-step RIPEMD-160

To attack more steps of reduced RIPEMD-160, we need to identify new message word differences. If we still wish to avoid state word differences in round 3, only message words m_0, m_{12} and m_{13} can have non-zero differences, as they appear after the 40-th step in round 3 of both branches. However, paper [7] indicates that if differences are introduced into these message words using the same differential characteristic structure, the local collision in round 2 of the left branch requires at least 44 conditions. Consequently, the overall probability of the differential characteristics is too low to yield a valid collision attack.

Therefore, we need to identify a new differential characteristic structure to attack more steps. Since no simple local collision exists in round 2, we need to use a single continuous differential characteristic spanning rounds 1 to 2 to cover all message differences in one branch. This structure has the advantage that we can introduce a difference into only one message word to obtain suitable differential characteristics.

If we inject a difference into message word m_{13}, no valid differential characteristic exists in the left branch because state words X_{13} and X_{18} are too close. Comparing message words m_0 and m_{12}, m_0 is used to calculate X_{25} and Y_{20} in round 2, while m_{12} is used for X_{24} and Y_{27}. When summed over both branches, m_0 affects fewer steps in round 2; therefore, its corresponding differential characteristic may contain fewer uncontrolled conditions. After several attempts, we select the message difference where only m_0 contains a difference $\delta m_0 = 0 \boxminus 2^{17}$. The corresponding differential characteristic structure is illustrated in Fig. 4.

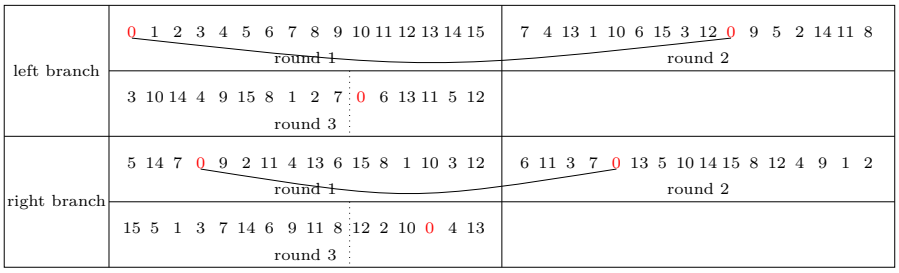

Fig. 4. Differential characteristic structure for 42-step RIPEMD-160

3.2 Search for Differential Characteristics

To optimize the total time complexity of our collision attack, when searching for differential characteristics, we should limit the number of uncontrolled conditions in later steps. Since the double-branch structure of RIPEMD-160 constrains the degrees of freedom of message words to satisfy conditions in state words of both branches, we must first determine our message modification strategy and identify which state words will remain uncontrolled.

We employ a similar method in this step as [13], with additional refinements to handle the more complex step functions of RIPEMD-160. First, based on our differential characteristic structure shown in Fig. 4, we observe that the left branch contains more conditions in the later steps of round 1; therefore, we choose m_{15} as the last message word to modify. In this way, state words after X_{15} and Y_{10} become uncontrolled and conditions in them should be minimized. Next, we need to control conditions in middle steps to fulfill the limitation of degrees of freedom of message words. After a few trials, we choose state words from X_0 to X_7 and Y_0 to Y_9 as the dense part, from X_8 to X_{15} and Y_{10} as the sparse part. When all state words in the dense part and corresponding message words are fixed, only message words m_8, m_{10}, m_{12}, and m_{15} remain undetermined, hence the number of conditions in sparse part should be as few as possible. In addition, for the dense part, we also need to control their differences to ensure the existence of valid partial message word and state word solutions.

For RIPEMD-160, we also need to consider the intermediate values Q_i^x and Q_i^y. In Section 2.4, we discussed how to use simple conditions on ΔQ and $\Delta(Q \lll s)$ to control the probability when searching for differential characteristics. Adding conditions on higher bits can guarantee higher probability but also restricts the search space. After overall consideration and several trials, we decided to impose the conditions $\Delta Q[32-s-2] = 0$ and $\Delta(Q \lll s)[s-2] = 0$ in the sparse part during the search. In this case, according to Property 1, we can ensure that the probability of proper difference propagation of the intermediate value is at least 2^{-4} (In fact, from Table 6, we can calculate that for the final differential characteristics we identified, this probability is at least $2^{-2.5}$).

In [25], the authors also proposed a method to analyze the probability of intermediate values. They introduced more than sixty SMT statements per step

to enumerate all possible positions of the first '1' in order to approximate the probability more accurately. However, this model is overly complicated, and most of these cases are unnecessary in practice. For instance, if the probability for a single step is lower than 2^{-10}, the resulting differential characteristics are typically unusable. Our method introduces only two SAT constraints per step and still ensures sufficiently high probability. Thus, our approach is more suitable for SAT-solver-based automatic tools.

In the following, we describe our approach to searching for suitable differential characteristics using our SAT-solver-based automatic search tool in detail:

Step 1: For the right branch, search for the solution with the minimum number of conditions in the sparse part from Y_{10} to Y_{20}, subject to the constraints that state words Y_{16} to Y_{20} have no difference, and that $\Delta Q_i^r[32 - s_i^r - 2] = 0$ and $\Delta(Q_i^r \lll s_i^r)[s_i^r - 2] = 0$ for $10 \leq i \leq 20$.

Step 2: Search for a differential characteristic whose number of conditions in the sparse part matches the result obtained in Step 1, while minimizing the differences in the dense part from Y_3 to Y_9.

Step 3: For the left branch, the differential characteristic is longer. Search for state word differences from X_{13} to X_{16} with the minimum number of differences; these differences are then fixed for the subsequent steps., so that the running time of the SAT-solver remains reasonable.

Step 4: Based on the state word differences obtained above, similar to Step 1, search for the differential characteristic with the minimum number of conditions in the sparse part from X_8 to X_{15} that satisfy the intermediate value constraints.

Step 5: Similar to Step 2, search for the final differential characteristic for the left branch, where the number of conditions in the sparse part matches that of Step 4, while minimizing the differences in the dense part from X_0 to X_7.

The differential characteristics we searched for that meet the above requirements are listed in Table 5, where the meaning of the symbols used to describe conditions is explained in Table 2. The corresponding extra conditions on Q_i^x and Q_i^y are shown in Table 6.

3.3 Find Colliding Message Pairs

After obtaining the differential characteristics, we can find corresponding colliding message pairs according to the following steps:

Step 1: Generate the first message block M_0 randomly to obtain initial state words CV_1 that satisfy the conditions on the initial state word X_{-1} shown in Table 5 and the extra conditions on intermediate values shown in Table 6.

Step 2: After obtaining suitable initial state words, use them as parameters in the SAT-solver-based automatic search tool to derive the message

Table 5. 42-step collision differential characteristics and conditions

step i	X_i	conditions	Y_i	conditions
−5	.....	0	.....	0
−4	.....	0	.....	0
−3	.....	0	.....	0
−2	.....	0	.....	0
−1	.eff.....	3	.....	0
0	.unn.....b.ffee.ef.....	9	.....	0
1	.ghhf.f.....nuuufnu..ib.efeee	19	......kk11...k.....	3.2
2	u.fnnauaaa...g.fhnhnug.babeunnnn	26	1.....1100...0.....1k..00	8.4
3	..n.ug.annbnen.u.g.uu.jenangggghg	24	1..u.....k1111k01....0kk...unnn	14.6
4	nju.nufnguugngjh.banenbugb.nafun	29	k0000..01k110000nn111nu001.k01.1	26.2
5	uauah.ug..u.h.anbj.hngnnu.agenfj	25	uuuunnnu00.11uu..0.1.01kk..01000	23.8
6	nn..u.uuennb.bun.aenea.nu.nuu.ub	24	.1.11.0n.....1k1.nuu.uuuu01101nn	21.4
7	uh..uahnuu..aunh.nu.n.au..ggu.hu	22	k.k0.0.100.00un1..u.nn..01..u0.k	18.2
8	....ub.nn.....u.n..jn...n..u..	10	1.0nuu..11.11.n...1.k.....0.0	13.4
9	...n.....u..u..n..i.a.b.....na	9	...01.....1.....1un.n111....	10
10	...h..u.n.n.be....aa.u....u.b..	11	....0k....n01.1.....00.....	7.4
11	...a...ff.h..n.....nbg...ah.....	10	1...n0....10.....k.....	5.4
12	.....uu..b.g..a..g.....b.a.	7	....1....k.....0.u1...	4.4
13	u..a...hga.....	5	..0.....1....n..1.....	4
14	.....1e.....bb.	4	..1.....u	2
15	..e.....u.....1.....01	5	....a.....n.....	2
16	..n.....0.....0.....n.	4	.....a.....	1
17	..0.....e.....u.....01	5	.....a.....	1
18	..1.....n.....10..1.....e	6	.....	0
19	.....1..0.....0.....u	4	.....	0
20	....1.....n.....0	3	.....	0
21	.....1..0.....	2	.....	0
22	.....1.....	1	.....	0
23	.....	0	.....	0
24	.....	0	.....	0
25	.....	0	.....	0
26	.....	0	.....	0
27	.....	0	.....	0
28	.....	0	.....	0
29	.....	0	.....	0
30	.....	0	.....	0
31	.....	0	.....	0
32	.....	0	.....	0
33	.....	0	.....	0
34	.....	0	.....	0
35	.....	0	.....	0
36	.....	0	.....	0
37	.....	0	.....	0
38	.....	0	.....	0
39	.....	0	.....	0
40	.....	0	.....	0
41	.....	0	.....	0

Table 6. Extra conditions on intermediate values of 42-step collision differential characteristics

Extra conditions:
$(Q_i^x \boxplus \alpha_i^x) \lll s_i^x = Q_i^x \lll s_i^x \boxplus \beta_i^x$,
$(Q_i^y \boxplus \alpha_i^y) \lll s_i^y = Q_i^y \lll s_i^y \boxplus \beta_i^y$.

step i	s_i^x	α_i^x	β_i^x	s_i^y	α_i^y	β_i^y
0	11	0xfffe0000	0xf0000000	8	0x0	0x0
1	14	0x90000000	0x2400	9	0x0	0x0
2	15	0x8fff2c00	0x960047ff	9	0x0	0x0
3	12	0x8a018d2f	0x18d2e8a0	11	0xfffe0000	0xefffffff
4	5	0x7325409b	0x64a8134f	13	0x10000007	0xc200
5	8	0x845d510b	0x5d510a84	15	0x39f4	0x1cfa0000
6	7	0x5b687e2c	0xb43f162e	15	0x11060200	0x1000883
7	9	0x4a164dd3	0x2c9ba694	5	0xbfffef81	0xfffdf038
8	11	0x912b266c	0x59336489	7	0x1f400	0xfa0000
9	13	0x944c5d93	0x8bb2728a	7	0x18300004	0x1800020c
10	14	0x842a0977	0x825de10b	8	0xfbfffdf4	0xfffdf3fc
11	15	0x6c3e0ed1	0x768361f	11	0x20a04	0x10502000
12	6	0x67ffde41	0xfff7901a	14	0xff7fe000	0xf7ffffe0
13	7	0x801bfff0	0xdfff7c0	14	0x20	0x80000
14	9	0x2c00840	0x80108006	12	0xfff80000	0x7fffffff
15	8	0xffef8000	0xef800000	6	0x80000001	0x60
16	7	0x10400000	0x20000008	9	0x20	0x4000
17	6	0xffffffff8	0xffffffe00	13	0xffffc000	0xf8000000
18	8	0x200	0x20000	15	0x8000000	0x400
19	13	0x0	0x0	7	0xffffffc00	0xfffe0000
20	11	0xfffffffff	0xfffff800	12	0x0	0x0
21	9	0x800	0x100000	8	0x0	0x0
22	7	0xfff00000	0xf8000000	9	0x0	0x0
23	15	0x8000000	0x400	11	0x0	0x0
24	7	0xffffffc00	0xfffe0000	7	0x0	0x0
25	12	0x0	0x0	7	0x0	0x0
26	15	0x0	0x0	12	0x0	0x0
27	9	0x0	0x0	7	0x0	0x0
28	11	0x0	0x0	6	0x0	0x0
29	7	0x0	0x0	15	0x0	0x0
30	13	0x0	0x0	13	0x0	0x0
31	12	0x0	0x0	11	0x0	0x0
32	11	0x0	0x0	9	0x0	0x0
33	13	0x0	0x0	7	0x0	0x0
34	6	0x0	0x0	15	0x0	0x0
35	7	0x0	0x0	11	0x0	0x0
36	14	0x0	0x0	8	0x0	0x0
37	9	0x0	0x0	6	0x0	0x0
38	13	0x0	0x0	6	0x0	0x0
39	15	0x0	0x0	14	0x0	0x0
40	14	0x0	0x0	12	0x0	0x0
41	8	0x0	0x0	13	0x0	0x0

words $m_0, \ldots, m_7$, m_9, m_{11}, m_{13}, and m_{14}, as well as the state words $X_0, \ldots, X_7$ and $Y_0, \ldots, Y_9$. However, to increase the success rate of later steps, only those partial message words and state words that can satisfy all conditions and differences on state words from X_0 to X_{15} and from Y_0 to Y_{15} are considered valid. If the tool returns no solution, return to Step 1 and repeat the process.

Step 3: Enumerate message words m_8. For the current m_8, if there exist m_{10}, m_{12}, and m_{15} satisfying all conditions on the state words $X_8, \ldots, X_{15}$, $Y_{10}, \ldots, Y_{15}$, and on the intermediate values $Q^x_8, \ldots, Q^x_{15}$ and $Q^y_{10}, \ldots, Q^y_{15}$, store the value of m_8 in a table `Tab_m8`. If the size of `Tab_m8` is at least 560, the stored values are ready for further enumeration in Step 4; otherwise, the degrees of freedom are considered insufficient, and return to Step 1 and regenerate a new initial state words.

Step 4: For each m_8 stored in `Tab_m8`, enumerate m_{10} and directly modify the conditions on X_{10} and Q^x_{10}, then verify whether the conditions on X_{11} and Q^x_{11} are satisfied. Apply the same process to m_{12} to satisfy all conditions on X_{12}, X_{13}, X_{14}, Q^x_{12}, Q^x_{13}, and Q^x_{14}.

Step 5: Utilize the remaining message word m_{15} to satisfy conditions on X_{15}, Q^x_{15}, Y_{10}, and Q^y_{10}. After this step, all message words are determined and verify whether they constitute a 42-step collision. If all possible values of m_8, m_{10}, m_{12}, and m_{15} have been tested and no solution is found, return to Step 1 and attempt another first message block.

Theoretical Time Complexity. There are 3 conditions on the initial state words and 2 extra conditions related to them. Hence, the time complexity of a single execution of Step 1 is 2^5. For Step 2 and Step 3, based on our examination, the success rates for obtaining valid partial message words are approximately 2^{-5} and 2^{-4}, respectively, and the cost for our automatic search tool to obtain partial message words or return no solution is equivalent to approximately 2^{30} RIPEMD-160 computations. Considering all corresponding conditions, for $560 \approx 2^{9.1}$ values of m_8, we can obtain about $2^{16.8}$ values of m_{10}, and then about $2^{28.4}$ values of m_{12} in Step 4. For Step 5, the degrees of freedom of m_{15} are sufficient; therefore, the time complexity of the overall attack is dominated by the uncontrolled conditions after X_{15} and Y_{10}. The number of uncontrolled conditions after X_{15} is 25, and after Y_{10} is 19.8. In addition, the total probability for intermediate values after Q^x_{15} is $2^{-0.3}$, and that after Q^y_{10} is $2^{-2.3}$. Assuming that the corresponding state words and intermediate values are uniformly random, the probability that all of these conditions are satisfied is approximately $2^{-47.4}$; consequently, the total time complexity of the collision attack is $2^{47.4}$.

After a more precise analysis, we find that when partial message words and state words are predetermined, other state words may not be uniformly random; hence, the probability of some uncontrolled conditions may differ from 2^{-1}. For example, in the left branch, predetermined state words and extra conditions constrain bit $m_8[20]$. Then, for the condition $Y_{11}[31] = 1$ in the right branch, its corresponding bit mainly depends on bits $Y_{10}[20]$, $Y_9[20]$, $Y_8[10]$, $Y_7[21]$, $Y_6[10]$,

and $m_8[20]$. We observe that all these bits are either predetermined or constrained; therefore, the probability of this condition may be close to 0 or 1. If we select the first message block that can make this probability close to 1, we can achieve a more efficient collision attack. After several attempts, we found two superior first message blocks (the cost of rejecting unsuitable first message blocks is included in the success probability of Step 3). Using these message blocks together with our attack strategy, we successfully identified three colliding message pairs for 42-step RIPEMD-160 using 560 cores (on 2.7 GHz, Intel Xeon 6258R CPUs). The time to obtain the first result for each message block was 30 minutes and 2 hours, respectively. Considering that each core can execute approximately 2^{22} reduced 42-step RIPEMD-160 compression function calls per second, the actual running time corresponds to around $2^{41.9}$ and $2^{43.9}$ compression function calls, which is lower than the theoretical estimate. The resulting colliding message pairs are presented in Table 7.

4 Conclusion

In this paper, we have presented practical collision attacks against a 42-step reduced version of RIPEMD-160. Unlike previous work that relied on sparse local collisions, we introduced a new differential characteristic structure that connects message differences across rounds 1 and 2 with a single continuous path. This design allows us to introduce only a one-bit difference into the message word m_0. Using an automatic search tool built on the high-performance SAT-solver PRS, we successfully identified three colliding message pairs within practical runtime on a multi-core platform, confirming the feasibility of our approach.

To the best of our knowledge, this result pushes the boundary of practical collision attacks on RIPEMD-160 beyond the previous 40-step limit. While the 42-step attack does not break the full hash function, it demonstrates that the security margin of reduced-round RIPEMD-160 may be narrower than previously thought.

Table 7. 42-step colliding message pairs

Colliding message pair 1:	
CV_0	67452301 efcdab89 98badcfe 10325476 c3d2e1f0
M_0	b81b8973 1e8ab91e a282f1f6 617f6250 4248c299 32a091ed 60768902 38925c35 638e77cd 751723be 2180b60e 9434e64b d0c1334b 8bcf0e5c 7076426f 5303df65
CV_1	2c243ff5 1cba8764 4428a148 c49a9ffb d4935df7
M_1	5dd2288b d55f3928 5c58085d 2f89a75e fba5e680 ce84e293 007d7ed3 4dca9955 d5c29fcb 88d983e0 fb942d85 3ac676eb fe09a334 3d1b5a6e 0a2e1242 35cd36ef
M_1'	5dd0288b d55f3928 5c58085d 2f89a75e fba5e680 ce84e293 007d7ed3 4dca9955 d5c29fcb 88d983e0 fb942d85 3ac676eb fe09a334 3d1b5a6e 0a2e1242 35cd36ef
H	e2a4eef4 dc101472 e680ab40 4b6b2303 42951202

Colliding message pair 2:	
CV_0	67452301 efcdab89 98badcfe 10325476 c3d2e1f0
M_0	47046b41 bd565fd3 08a80cc9 eb0ceeb5 8ca535af 47f4d2c4 e698fa06 33017a62 ca2985fa 94366ca7 a45d259c 639cf15d 9d9a164f c5500897 9dc008c1 e7ca71b8
CV_1	f4749026 9a36d415 4428a148 1d64284c f536b36c
M_1	fedba86c d4064ab8 005670a5 8af0a84c e23ca74b f16db237 7ff759e6 bd27ea2d 08d42416 a296eceb 47be5fe3 d1374a66 8071fda9 8817d9b6 57c544dc a7c0e967
M_1'	fed9a86c d4064ab8 005670a5 8af0a84c e23ca74b f16db237 7ff759e6 bd27ea2d 08d42416 a296eceb 47be5fe3 d1374a66 8071fda9 8817d9b6 57c544dc a7c0e967
H	860e4522 8fd73037 0838d91e dcea2291 9ccf1e23

Colliding message pair 3:	
CV_0	67452301 efcdab89 98badcfe 10325476 c3d2e1f0
M_0	47046b41 bd565fd3 08a80cc9 eb0ceeb5 8ca535af 47f4d2c4 e698fa06 33017a62 ca2985fa 94366ca7 a45d259c 639cf15d 9d9a164f c5500897 9dc008c1 e7ca71b8
CV_1	f4749026 9a36d415 4428a148 1d64284c f536b36c
M_1	fedba86c d4064ab8 005670a5 8af0a84c e23ca74b f16db237 7ff759e6 bd27ea2d ffd631d8 a296eceb 144a9cbd d1374a66 cbf5e046 8817d9b6 57c544dc 5a5fa436
M_1'	fed9a86c d4064ab8 005670a5 8af0a84c e23ca74b f16db237 7ff759e6 bd27ea2d ffd631d8 a296eceb 144a9cbd d1374a66 cbf5e046 8817d9b6 57c544dc 5a5fa436
H	0a809a52 857bd090 601f80de ec0be1ba ff7548ed

References

1. 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
2. Chen, Z., Zhang, X., Qian, Y., Zhang, C., Cai, S.: PRS: An efficient parallel SAT framework. In: Proceedings of the 63rd ACM/IEEE The Chips to Systems Conference (DAC) (2026), to appear
3. De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006. pp. 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
4. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of ripemd. In: Gollmann, D. (ed.) Fast Software Encryption. pp. 71–82. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
5. Fouque, P., Leurent, G., Nguyen, P.Q.: Automatic search of differential path in MD4. IACR Cryptol. ePrint Arch. **2007**, 206 (2007), http://eprint.iacr.org/2007/206
6. Landelle, F., Peyrin, T.: Cryptanalysis of full RIPEMD-128. J. Cryptol. **29**(4), 927951 (Oct 2016). https://doi.org/10.1007/s00145-015-9213-5, https://doi.org/10.1007/s00145-015-9213-5
7. Li, Y., Liu, F., Wang, G.: Automating collision attacks on RIPEMD-160. IACR Transactions on Symmetric Cryptology **2023**(4), 112142 (Dec 2023). https://doi.org/10.46586/tosc.v2023.i4.112-142, https://tosc.iacr.org/index.php/ToSC/article/view/11282
8. 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. pp. 158–186. Springer Nature Switzerland, Cham (2024)
9. 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. pp. 117–149. Springer International Publishing, Cham (2019)
10. Liu, F., Dobraunig, C., Mendel, F., Isobe, T., Wang, G., Cao, Z.: New semi-free-start collision attack framework for reduced RIPEMD-160. IACR Transactions on Symmetric Cryptology pp. 169–192 (09 2019). https://doi.org/10.46586/tosc.v2019.i3.169-192
11. 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. pp. 158–186. Springer International Publishing, Cham (2017)
12. 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. pp. 189–219. Springer Nature Switzerland, Cham (2023)
13. Lu, Z., Yu, H., Lin, X., Yuan, S.: Collision attacks on reduced RIPEMD-128. IACR Transactions on Symmetric Cryptology **2025**, 16–33 (06 2025). https://doi.org/10.46586/tosc.v2025.i2.16-33
14. Manuel, S., Peyrin, T.: Collisions on SHA-0 in one hour. In: Nyberg, K. (ed.) Fast Software Encryption. pp. 16–35. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

15. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. pp. 288–307. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
16. Mendel, F., Nad, T., Schläffer, M.: Improving local collisions: New attacks on reduced SHA-256. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. pp. 262–278. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
17. 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. pp. 102–115. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
18. 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. pp. 570–596. Springer International Publishing, Cham (2017)
19. 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. pp. 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
20. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. pp. 17–36. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
21. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. pp. 19–35. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
22. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) Advances in Cryptology – CRYPTO 2005. pp. 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
23. Zhang, X., Chen, Z., Cai, S.: Parkissat: Random shuffle based and pre-processing extended parallel solvers with clause sharing. SAT Competition **51** (2022)
24. Zhang, Z., Li, M., Gao, L., Wang, M.: Collision attacks on SHA-256 up to 37 steps with improved trail search. In: Daemen, J., Thomé, E. (eds.) Advances in Cryptology – EUROCRYPT 2026. pp. 91–120. Springer Nature Switzerland, Cham (2026)
25. Zhang, Z., Li, M., Wang, H., Hou, S., Wang, W., Wang, M.: Improved semi-free-start collision attacks on RIPEMD-160. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025. pp. 221–252. Springer Nature Singapore, Singapore (2026)