IACR Transactions on Symmetric Cryptology
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–18. DOI:XXXXXXXX

Collision Attacks on Reduced RIPEMD-128

Zhengrong Lu[1], Hongbo Yu[1,2,3(✉)], Xiaoen Lin[1] and Sitong Yuan[1]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, People's Republic of China luzr20@mails.tsinghua.edu.cn,yuhongbo@mail.tsinghua.edu.cn,lxe21@mails.tsinghua.edu.cn,yuanst23@mails.tsinghua.edu.cn
[2] Zhongguancun Laboratory, Beijing, People's Republic of China
[3] State Key Laboratory of Cryptography and Digital Economy Security, Tsinghua University, Beijing, 100084, People's Republic of China

Abstract. RIPEMD-128 is an ISO/IEC standard hash function based on a double-branch Merkle-Damgård structure. Its compression function includes two branches with distinct Boolean functions and message expansion permutations. To perform a collision attack, differential characteristics must be constructed simultaneously for both branches under the same message word difference, and the message modification order must align with conditions in both branches. These factors make collision attacks on (reduced) RIPEMD-128 highly challenging.

In 2014, an attack on 40 steps of RIPEMD-128 was achieved by Wang with no state differences in round 3. In this work, we analyze message permutation properties and propose two new structures for creating message differences. These structures enable high-probability local collisions in both branches of round 3, extending the attack to more steps. Notably, the second structure can eliminate all state differences in round 3, allowing the attack to cover more than three whole rounds.

To ensure practical attacks, we limit the number of conditions based on our message modification strategy and use multi-step message modification techniques to control more conditions. As a result, we successfully generate colliding message pairs for 46-step and 54-step reduced RIPEMD-128, with time complexities of approximately 2^{42} and 2^{54}, respectively.

Keywords: hash function · RIPEMD-128 · collision attack · message difference · message modification

1 Introduction

Cryptanalysis of hash functions is an important subject in cryptography. In general, a hash function has three basic security properties: preimage resistance, second-preimage resistance, and collision resistance. In 2005, modular differential attack was introduced, which could break many hash functions in the MD-SHA family, such as MD4 [WLF+05], RIPEMD [WLF+05], MD5 [WY05], SHA-0 [WYY05b], and SHA-1 [WYY05a, SBK+17].

RIPEMD [BP95] is a double-branch hash function where the two branches differ only in their round constants. This implies that the same differential characteristic can be applied simultaneously to both branches. To address some weaknesses of RIPEMD, RIPEMD-128 [DBP96] was introduced as a strengthened version in 1996. In addition to round constants, the message word permutations and Boolean functions within the same round differ between the two branches. As a result, message word differences are introduced into different state words and propagate according to distinct state functions. This necessitates high-probability differential characteristics for the same message word differences in both branches simultaneously. Moreover, this property of RIPEMD-128 increases the

complexity of message modification, as modifying a message word to satisfy conditions in one branch simultaneously fixes it in the step function of the other branch.

Related Work. The first attempt at a collision attack on RIPEMD-128 was proposed by Mendel et al. in 2006 [MPRR06]. However, the differential characteristics they constructed were too complex to constitute a valid attack. At FSE 2012, Mendel et al. discovered a suitable message difference structure and utilized automatic tools to achieve a 38-step collision attack [MNS12]. In the same paper, they also presented a 44-step near-collision result and a 48-step free start collision result. Later, at EUROCRYPT 2013, a theoretical semi-free start collision attack for full RIPEMD-128 was proposed [LP13]. In 2014, Wang demonstrated a practical collision attack on the reduced 40-step RIPEMD-128 [Wan14, WY15], which remains the best collision attack result to date.

There have also been several preimage attacks on reduced RIPEMD-128. In 2011, preimage attacks on 33-step and 35-step (starting from intermediate steps) were proposed [OSS10]. Later, an improved result was presented, which extended the attack to 36 steps starting from intermediate steps [WSK+11].

Finally, several distinguishing attacks on the full RIPEMD-128 are presented in [MNS12, IPS13, WY15].

Our Contribution. To improve the collision attack results on reduced RIPEMD-128, we attempt to introduce differences in round 3. This approach presents two challenges. First, we need to find new message difference structures capable of generating local collisions in round 3 of both branches with high probabilities. Second, when selecting message differences suitable for round 3, their corresponding differential characteristics in rounds 1 and 2 become more complex relatively and involve a greater number of uncontrollable conditions. Therefore, we need to search for better differential characteristics and employ more advanced message modification techniques to reduce the complexity and achieve a practical attack. First, we analyze the message permutations and identify a specific property. Based on this property, we propose two types of message differences that can generate short local collisions in round 3 of both branches simultaneously. Next, we choose two specific message differences and search for proper differential characteristics based on the order of message modification to fully utilize the degrees of freedom in message words. Finally, we apply our message modification techniques to obtain practical colliding message pairs. The previous results and our results are summarized in Table 1.

Table 1: Summary of attack results on RIPEMD-128

attack type	steps	complexity	reference
collision	38	2^{14} (practical)	[MNS12]
collision	40	2^{35} (practical)	[Wan14, WY15]
collision	46	2^{42} (practical)	Subsection 4.2
collision	54	2^{54} (practical)	Subsection 5.2
near-collision	44	2^{32} (practical)	[MNS12]
free start collision	48	2^{40} (practical)	[MNS12]
semi-free start collision	64 (full)	$2^{61.57}$	[LP13]
preimage	33	$2^{124.5}$	[OSS10]
preimage	35	2^{121}	[OSS10]
preimage	36	$2^{126.5}$	[WSK+11]
distinguishing	64 (full)	$2^{105.4}$	[MNS12]
distinguishing	64 (full)	$2^{95.8}$	[IPS13]
distinguishing	64 (full)	$2^{90.4}$	[WY15]

The source code for generating and verifying colliding message pairs is available at https://github.com/usernamelzr/ripemd128_attack. The automatic differential characteristics search tool we utilized is provided at https://github.com/usernamelzr/cipher-auto-search-tool.

The remainder of the paper is organized as follows. We describe the hash function RIPEMD-128 in Section 2. Some techniques we used to improve the collision attack results is presented in Section 3. Next, in Section 4 and Section 5, we detail our differential characteristics and the message modification steps for collision attacks on the reduced 46-step and 54-step RIPEMD-128, respectively. Finally, we summarize our findings and conclude the paper in Section 6.

2 Description of RIPEMD-128

RIPEMD-128 is a 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 algorithm begins with a 128-bit initial value $CV_0 = IV$ and iteratively applies the compression function $CV_{i+1} = H(CV_i, M_i)$ to each message block to produce the final output. The initial value IV is defined as follows:

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

The compression function $H(CV, M) = (H_0, H_1, H_2, H_3)$ of RIPEMD-128 consists of two branches, each containing 4 rounds with 16 steps per round. While the structure of both branches is similar, they differ in their message expansion permutations and the order of Boolean functions. Within the compression function, X_i denotes the state word in the left branch, and Y_i represents the state word in the right branch.

The input CV of the compression function is divided into 4 32-bit words, denoted as cv_0 to cv_3. The initial state words for both branches are derived by CV as follows:

$$\begin{aligned} X_{-4} &= Y_{-4} = cv_0, \\ X_{-3} &= Y_{-3} = cv_3, \\ X_{-2} &= Y_{-2} = cv_2, \\ X_{-1} &= Y_{-1} = cv_1. \end{aligned}$$

The 512-bit input message block M is also divided into 16 message words, denoted as m_0 to m_{15}. Each step utilizes one message word, and each round employs all 16 message words exactly once. The step function of RIPEMD-128 at step i is defined as follows, where $i = (0, 1, 2, ..., 63)$. In the description, $+$ denotes addition modulo 2^{32}, $\lll$ denotes a circular left shift.

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

Boolean functions and round constants used in the step function are listed in Table 2. In the Boolean functions, $\vee$, $\wedge$, and $\oplus$ represent bit-wise OR, AND, and XOR, respectively. Message permutation constants π_i^x and π_i^y, and rotation constants s_i^x and s_i^y in the step function, are listed in Table 3.

Finally, a feed-forward operation is applied to compute the output (H_0, H_1, H_2, H_3) of the compression function:

Table 2: Boolean functions and round constants in RIPEMD-128

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

Table 3: Message permutation constants and rotation constants in RIPEMD-128

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

$$H_0 = cv_1 + X_{62} + Y_{61},$$
$$H_1 = cv_2 + X_{61} + Y_{60},$$
$$H_2 = cv_3 + X_{60} + Y_{63},$$
$$H_3 = cv_0 + X_{63} + Y_{62}.$$

3 Improve the Collision Attack for RIPEMD-128

The modular differential attack strategy [WLF+05, WY05] for hash functions based on Merkle-Damgård structure, consists of the following steps:

Step 1: Identify suitable message differences to construct simple local collisions in the later rounds.

Step 2: Search for suitable modular differential characteristics in preceding rounds and compute their corresponding conditions. These modular differential characteristics are consisted by modular differences in each state word and conditions controlling the propagation of these differences. In Step 3, we must identify message pairs satisfying all differential characteristic conditions. Consequently, the time complexity depends on the number of uncontrolled conditions, which should be minimized during the search process.

Step 3: Randomly generate message pairs and apply message modification techniques to satisfy some conditions. Then, verify whether the modified message pair results in a collision for the hash function. If not, repeat this step until a colliding message pair is found.

In the remainder of this section, we will describe our techniques to improve the above three steps to achieve a better collision attack result.

3.1 New Message Difference Structures

RIPEMD-128 is a double-branch hash function with distinct message word permutations and Boolean functions in its two branches. Consequently, selecting suitable message word differences that can generate high-probability local collisions in both branches simultaneously is essential.

The theoretical semi-free start collision attack on full RIPEMD-128 [LP13] utilizes a single-bit difference in m_{14} to construct state differences cancellation in round 4. This approach results in numerous conditions in rounds 2 and 3. Their method involves starting from the midpoint of both branches, performing backward modification, and searching for a suitable IV to merge the branches. However, for collision attacks where the IV cannot be arbitrarily selected, we face significantly constrained degrees of freedom for message modification.

Previous collision attack results, such as the 38-step [MNS12] and 40-step [Wan14, WY15] collision attacks, primarily focused on message words appearing late in round 3. This approach allowed them to avoid introducing state differences in round 3 and instead focus on finding differential characteristics in rounds 1 and 2 to achieve collisions. However, to attack reduced versions of the hash function with more steps, it becomes necessary to introduce differences in round 3. This introduces additional complexity, as it may lead to more intricate differential characteristics with higher number of uncontrollable conditions. Therefore, identifying new message difference structures capable of generating short local collisions in round 3 is crucial. In the remainder of this subsection, we present two new

types of message differences that can produce local collisions in both branches of round 3 with high probabilities.

To identify short differential characteristics simultaneously in round 3, a key idea is to introduce differences in message words that appear in both branches with the same distances. From Table 3, we observe that within the same round, when $\pi_i^x = \pi_j^y$, it always holds that $s_i^x = s_j^y$. This implies that the same message word bit affects the same state word bit in both branches. Based on the message permutations in round 3, we can categorize all the message words (excluding m_5) into three groups such that if two message words belong to the same group, they exhibit the same distance in both branches during round 3. Each group is highlighted in the same color, as shown in Table 4, and summarized in Table 5. This approach enables similar difference cancellation structure in both branches when introducing differences to message words in the same group. That helps us generate local collisions in both branches simultaneously in round 3 with high probabilities.

Table 4: Message permutations in round 3

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
π_i^y	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13

Table 5: Message words with the same distance in both branches in round 3

Group	Message words
1	m_3,m_{14},m_9,m_8,m_2,m_0,m_{13}
2	m_{10},m_4
3	m_{15},m_1,m_7,m_6,m_{11},m_{12}

Another challenge we need to address is how to handle the Boolean functions used in round 3. For the right branch, if only one input bit of the function $(a \wedge b) \vee (\neg a \wedge c)$ has a difference, we can use conditions on the other input bits to control the difference of its output. However, for the left branch, the Boolean function is $(a \vee \neg b) \oplus c$. When c has a difference and the input bits a and b do not, the output of the Boolean function will always exhibit a difference. Therefore, we need to introduce an additional message word difference to cancel it out. In this way, we obtain the message differences type 1 and their corresponding local collisions in round 3. For the message differences type 1, we introduce two differences into message words within the same group so that they can cancel each other in the right branch. We then use an additional message difference to cancel the difference introduced by the Boolean function in the left branch. For example, the message differences $\Delta m_3 = +2^{21}$, $\Delta m_4 = +2^0$, and $\Delta m_9 = -2^0$ constitute a suitable choice because m_3 and m_9 belong to the same group. The corresponding local collisions and conditions are illustrated in Table 6 and Table 7. The conditions on X_i and Y_i regulate the modular differences and Boolean functions to ensure proper differential propagation in local collisions. Especially, conditions $X_{32}[0] = 0$ and $Y_{35}[0] = 0$ restrict the modular differences to bit 0 exclusively, while the remaining conditions control the output of the Boolean functions. To demonstrate how these conditions control differential propagation, we provide the following example. In the Boolean function $(a \vee \neg b) \oplus c$, when $a = 1$, the output becomes independent of b and when $b = 0$, the output becomes independent of a. Thus, the conditions $X_{31}[0] = 0$ and $X_{33}[0] = 1$ in Table 6 prevent difference propagation from step 32 to steps 33 and 34. Conditions in step 34 of the left branch and steps 34, 36, and 37 of the right branch employ a similar mechanism to control the output differences

of the Boolean functions.

However, since the extra difference cannot be canceled in the right branch, this type of message difference is limited to collision attacks on fewer than 48 steps.

Table 6: Local collision of the left branch for message differences type 1

step i	π_i^x	s_i^x	$\Delta m_{\pi_i^x}$	ΔX_i	conditions on X_i
31					$X_{31}[0] = 0$
32	3	11	$+2^{21}$	$+2^0$	$X_{32}[0] = 0$
33					$X_{33}[0] = 1$
34					$X_{34}[0] = 1$
35	4	7	$+2^0$		
36	9	14	-2^0		

Table 7: Local collision of the right branch for message differences type 1

step i	π_i^y	s_i^y	$\Delta m_{\pi_i^y}$	ΔY_i	conditions on Y_i
34					$Y_{34}[0] = Y_{33}[0]$
35	3	11	$+2^{21}$	$+2^0$	$Y_{35}[0] = 0$
36					$Y_{36}[0] = 0$
37					$Y_{37}[0] = 1$
38					
39	9	14	-2^0		

To attack more than 48 steps, we must ensure that each state difference in round 3 is canceled, meaning that each message difference must be part of a local collision. This requires an alternative approach to cancel the extra difference. By analyzing the left branch Boolean function $(a \vee \neg b) \oplus c$, we observe that when two adjacent input bits (a and b, or b and c) have the same difference, the output can still be controlled to remain unchanged. This approach allows us to cancel all extra differences. Consequently, we derive the message differences type 2 by introducing differences into three message words within the same group, which can mutually cancel each other in round 3. For example, the message differences $\Delta m_{15} = +2^{26}$, $\Delta m_7 = -2^3 + 2^9$, and $\Delta m_{11} = -2^{17}$ is a proper choice because m_{15}, m_7, and m_{11} belong to the same group. The corresponding local collisions and conditions are illustrated in Table 8 and Table 9.

Table 8: Local collision of the left branch for message differences type 2

step i	π_i^x	s_i^x	$\Delta m_{\pi_i^x}$	ΔX_i	conditions on X_i
36					$X_{36}[3] = 0$
37	15	9	$+2^{26}$	$+2^3$	$X_{37}[3] = 0$
38					$X_{38}[3] = 1$
39					$X_{39}[3] = 0, X_{39}[17] = 0$
40		14		$+2^{17}$	$X_{40}[17] = 0$
41	7	8	$-2^3 + 2^9$	$+2^{17}$	$X_{41}[17] = 0$
42					$X_{42}[17] = 0$
43					
43					
44	11	12	-2^{17}		

For round 4, we can adopt a method from [LP13] to cancel differences in the last four steps, thereby improving the attack result. If $\pi_a^x = \pi_{a+3}^y$, and there are no other differences

Table 9: Local collisions of the right branch for message differences type 2

step i	π_i^y	s_i^y	$\Delta m_{\pi_i^y}$	ΔY_i	conditions on Y_i
31					$Y_{31}[3] = Y_{30}[3]$
32	15	9	$+2^{26}$	$+2^3$	$Y_{32}[3] = 0$
33					$Y_{33}[3] = 0$
34					$Y_{34}[3] = 1$
35					$Y_{35}[17] = Y_{35}[17]$
36	7	8	$-2^3 + 2^9$	$+2^{17}$	$Y_{36}[17] = 0$
37					$Y_{37}[17] = 0$
38					$Y_{38}[17] = 1$
39					
40	11	12	-2^{17}		

on $m_{\pi^x_{a+1}}$, $m_{\pi^x_{a+2}}$, $m_{\pi^x_{a+3}}$, $m_{\pi^y_a}$, $m_{\pi^y_{a+1}}$, and $m_{\pi^y_{a+2}}$, we can select an appropriate differential bit for $m_{\pi^x_a}$ and leverage the feed-forward structure to cancel its difference with additional conditions. For example, when $\Delta m_{15} = +2^{26}$, $\Delta m_7 = -2^3 + 2^9$, and $\Delta m_{11} = -2^{17}$, we have $\pi^x_{50} = \pi^y_{53} = 11$. By introducing five conditions, we can ensure $\Delta X_{50} = \Delta Y_{53} = -2^{31}$ and $\Delta X_{51} = \Delta X_{52} = \Delta X_{53} = \Delta Y_{50} = \Delta Y_{51} = \Delta Y_{52} = 0$. When these equations are satisfied, the feed-forward operation of the reduced 54-step RIPEMD-128 ensures the output differences $\Delta H_0 = \Delta H_1 = \Delta H_2 = \Delta H_3 = 0$.

3.2 Search for Differential Characteristics

After identifying suitable message differences, we need to search for proper differential characteristics in rounds 1 and 2 to complete the attack. In the compress function of RIPEMD-128, the same message word is used in the step functions of both branches in one round. This implies that in most cases, if we utilize the degrees of freedom of a message word to modify conditions in one branch of round 1, we cannot simultaneously use it to control the corresponding state word in the other branch, as other parameters in the state function have not yet been determined at that point. Consequently, only partial state words in round 1 can be modified directly, while conditions in other state words will impact the time complexity.

Different message modification strategies can directly modify different state words, resulting in varying time complexities. Therefore, our approach is as follows: first, we determine the order of message modification based on the structure of differential characteristics. This allows us to identify which state words are controlled by which message words and which state words are not directly modified by their corresponding message words, thus the conditions associated with the latter should be minimized. Based on this information, we can then search for optimal differential characteristics by automatic tools. In addition, for a single message word, we only have 32 degrees of freedom. Therefore, if multiple state words are controlled by a single message word, the total number of conditions across these state words must not exceed 32 to avoid additional time complexity caused by insufficient degrees of freedom. Because the restriction of differential characteristics in our strategy is complex, we utilized an automated search tool based on SAT-solver. Based on the above properties, we outline our high-level method for finding differential characteristics in rounds 1 and 2:

Step 1: Determine the order of message modification based on the structure of the differential characteristics. State words near the first state difference are likely to have more conditions and thus should be given higher priority for controlled directly.

Step 2: Identify which state words are controlled by each message word and determine which state words cannot be directly modified. If a single message word controls multiple state words, we must impose a limitation in step 3.

Step 3: Use automatic tool to search for differential characteristics based on the above strategy. The limitations are that the number of conditions controlled by the same message word cannot exceed 32 to avoid additional time complexity, and the conditions in state words that are not directly modified by their corresponding message words should be minimized. Finally, state words with a large number of conditions will be fully fixed to simplify subsequent message modification.

3.3 Multi-step Message Modification

To reduce the time complexity of the collision attack, we can employ more precise multi-step message modification techniques to satisfy more conditions in round 1. In brief, when $m_{\pi_i^x}$ is predetermined, we cannot directly modify its corresponding state word X_i (the same for $m_{\pi_i^y}$ and Y_i). However, we can adjust X_{i-1} or X_{i-2} to modify the conditions in X_i using the Boolean functions.

For the left branch, the Boolean function in round 1 is $a \oplus b \oplus c$. This implies that if any bit of its input is modified, the output will also change. Consequently, we can adjust X_{i+1} or X_{i+2} by altering the corresponding bit in X_i by the step function. In addition, the conditions required to control the output difference of this Boolean function include $a = b$, $a \neq c$, and similar expressions (conditions described by symbols a, b, c and d in Table 14, will be introduced in Appendix A). Although these conditions pertain to later steps, we can modify the bit in the previous step to satisfy these conditions. For example, when the condition is $X_{i+1}[j] = X_i[j]$, modifying $X_i[j]$ is also a valid approach to satisfy the condition.

For the right branch, the Boolean function in round 1 is $(c \wedge a) \vee (\neg c \wedge b)$. This implies that when $c = 0$ or $c = 1$, the output of this function depends solely on b or a, respectively. Therefore, if Y_i and Y_{i+1} have degrees of freedom in the same bit j, we can impose the condition $Y_i[j] = 0$ and use Y_{i+1} to control the conditions in Y_{i+3}. Similarly, if Y_i and Y_{i+2} both have degrees of freedom in bit j, we can impose the condition $Y_i[j] = 1$ and utilize Y_{i+2} to control the conditions in Y_{i+3}.

4 Collision Attack on RIPEMD-128 Reduced to 46 Steps

For the collision attack, we select the message differences $\Delta m_3 = +2^{21}$, $\Delta m_4 = +2^0$, and $\Delta m_9 = -2^0$ in type 1. The corresponding local collisions in round 3 is shown in Table 6 and Table 7. These message differences enable us to use a single state difference in $Y_{25}[0]$ to cancel two differences introduced by m_4 and m_9 in round 2 of the right branch, resulting in a differential characteristic with fewer uncontrolled conditions. The overall structure of the differential characteristic generated by these message differences is illustrated in Figure 1. The curves in the figure represent differences cancel out. Specifically, the curves in rounds 1 and 2 depict the differential characteristics to be identified, while the curves in round 3 correspond to the local collisions described in Table 6 and Table 7.

4.1 Differential Characteristic

Based on our message modification strategy outlined in Table 10, we can employ automated tools to search for the differential characteristic in rounds 1 and 2. The total number of conditions in the state words controlled by the same message word must not

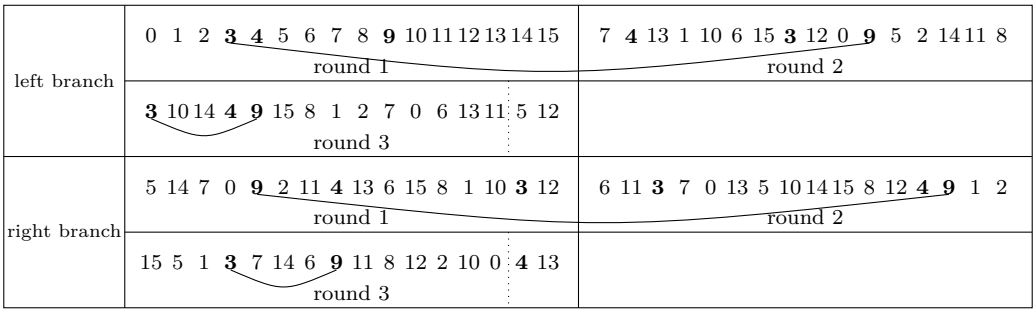

Figure 1: Structure of differential characteristic for 46-step RIPEMD-128

exceed 32. State words highlighted in red or not listed in Table 10 are not directly modified by their corresponding message words; therefore, the conditions in these state words should be minimized. The detailed results and corresponding conditions are presented in Table 15.

Table 10: Message modification order for 46-step RIPEMD-128

message words	controlled state words
m_0	X_0
m_1	X_1
m_2	X_2
m_3	X_3
m_4	X_4
m_5	X_5 and Y_0
m_6	X_6
m_7	X_7
m_{14}	Y_1, Y_2, and Y_3
m_9	Y_4 and Y_5
m_{11}	Y_6 and Y_7
m_{13}	Y_8 and Y_9
m_{15}	Y_{10}
m_8	Y_{11} and Y_{12}
m_{10}	X_{10}, X_{11}, Y_{13}, and Y_{14}
m_{12}	X_{12} and Y_{15}

4.2 Find Colliding Message Pair

After obtaining the differential characteristics, we can implement message modification according to Table 10. The procedure can be divided into the following steps, as illustrated in Figure 2.

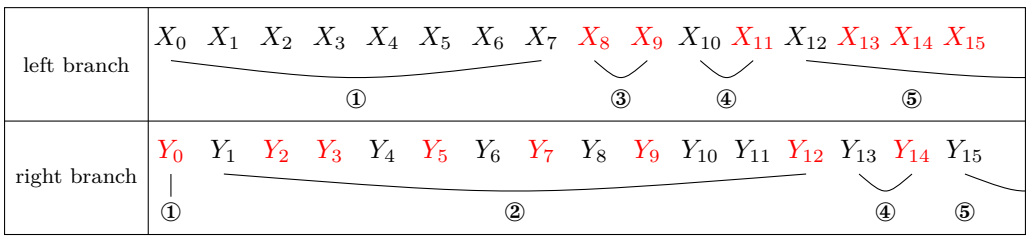

Figure 2: Steps of message modification for 46-step RIPEMD-128

Step 1: We calculate the message words from m_0 to m_7, based on the fixed state words X_0 to X_7 in the left branch.

Step 2: We begin modifying other message words to satisfy the conditions in the right branch. Since Y_0 has no conditions, it can be calculated directly using m_5. We then utilize the degrees of freedom of m_{14} to fulfill the conditions from Y_1 to Y_3. Here, the conditions in Y_1 can be modified directly, while those in Y_2 and Y_3 are adjusted indirectly. We repeat the random selection process until all conditions in these steps are satisfied. Next, we employ a similar approach to randomly generate and modify m_9, m_{11}, m_{13}, m_{15}, and m_8 to meet the conditions from Y_4 to Y_{12}.

Step 3: We use m_8 and m_9 to calculate X_8 and X_9, and then verify whether the conditions in these step words are satisfied. If not, we restart steps 2 and 3.

Step 4: There are 66 remaining conditions, but we only have 64 free bits in m_{10} and m_{12}. This implies that we need to find 2^2 solutions of step 3. Subsequently, we use the message word m_{10} to modify X_{10}, X_{11}, Y_{13}, and Y_{14}.

Step 5: Finally, we use the remaining message word m_{12} to modify X_{12} and Y_{15}. Additionally, we can implement the multi-step message modification techniques described in Subsection 3.3 to satisfy more conditions, thereby reducing the complexity. Specifically, we can use bits $10, 23, 25, 31$ of X_{12} to influence $X_{13}[17]$, $X_{14}[0]$, $X_{13}[0]$ and $X_{14}[8]$, thereby modifying four conditions. We then test whether the message pair constitutes a 46-step collision attack. If not, we repeat steps 4 and 5 by selecting alternative values for m_{10} and m_{12}.

Theoretical Time Complexity. The time complexity of step 2 depends on the number of conditions that cannot be directly modified. Specifically, Y_2 and Y_3 collectively have 14 conditions, Y_5 has 10 conditions, Y_7 has 11 conditions, Y_9 has 12 conditions, and Y_{12} has 11 conditions. Therefore, the overall complexity is $2^{14} + 2^{10} + 2^{11} + 2^{12} + 2^{11} \approx 2^{15}$. Additionally, since there are 24 conditions in X_8 and X_9, the expected time complexity for once step 3 is $2^{24} \times 2^{15} = 2^{39}$. There are 40 uncontrolled conditions remaining, resulting in a complexity of 2^{40} for steps 4 and 5. Since we need 2^2 solutions from step 3, the overall complexity of the entire attack is $2^2 \times 2^{39} + 2^{40} \approx 2^{42}$.

Using this approach, we successfully identified a colliding message pair for the reduced 46-step RIPEMD-128 on a typical PC within 20 hours. The results are presented in Table 11.

Table 11: 46-step colliding message pair

CV_0	67452301	efcdab89	98badcfe	10325476				
M_0	2d9942f6 d673ac60	8116c0fa b89cd050	ec95290c 50cacb0d	0f23a553 fd249d61	2a34e00f 08e0ba16	84797a0c ee28d84c	94c87810 3bfab38c	b389dfb7 045080be
M_0'	2d9942f6 d673ac60	8116c0fa b89cd04f	ec95290c 50cacb0d	0f43a553 fd249d61	2a34e010 08e0ba16	84797a0c ee28d84c	94c87810 3bfab38c	b389dfb7 045080be
H	10a518f7	44a599d3	cae0e914	bd0d53b4				

5 Collision Attack on RIPEMD-128 Reduced to 54 Steps

For the 54-step collision attack, we select the message differences $\Delta m_{15} = +2^{26}$, $\Delta m_7 = -2^3 + 2^9$, and $\Delta m_{11} = -2^{17}$ in type 2. The corresponding local collisions in round 3 is

shown in Table 8 and Table 9. This allows us to apply the method in round 4 to cancel two differences using the feed-forward structure. The overall structure of the differential characteristic generated by these message differences is illustrated in Figure 3. The curves in this figure carry similar significance to those presented in Section 4.

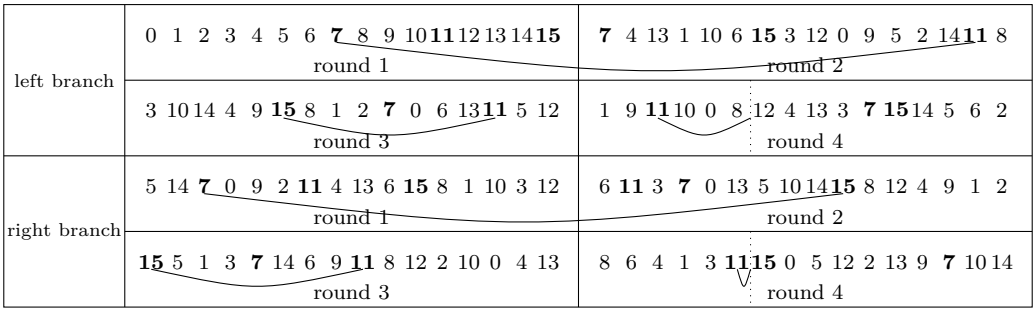

Figure 3: Structure of differential characteristic for 54-step RIPEMD-128

5.1 Differential Characteristic

For the 54-step collision attack, the message modification strategy outlined in Table 12 is more complex. This is because we first determine some intermediate state words (X_5 to X_{11}) in the left branch and then utilize other message words (m_1, m_3, m_4, m_6, and m_8) to connect them with previous state words X_{-3} to X_0. Since there are no conditions in the state words from X_0 to X_4, this strategy enables better control over conditions in the middle of the left branch. For the right branch, we fix the state words Y_0 to Y_4 to ensure that message word m_9 is consistent in both branches. Other constraints are similar to those in the 46-step collision attack. The detailed results and corresponding conditions are presented in Table 16.

Table 12: Message modification order for 54-step RIPEMD-128

message words	controlled state words
m_5	Y_0
m_{14}	Y_1
m_7	Y_2
m_0	Y_3 and X_0
m_9, m_{10} and m_{11}	X_5, X_6, X_7, X_8, X_9, X_{10}, X_{11}, and Y_4
m_2	Y_5 and Y_6
m_1, m_3, m_4, m_6, and m_8	X_1, X_2, X_3, and X_4
m_{13}	Y_8 and Y_9
m_{15}	Y_{10}, Y_{11}, Y_{12} , Y_{13}, and Y_{14}
m_{12}	X_{12} and Y_{15}

5.2 Find Colliding Message Pair

By employing the multi-step message modification techniques described in Subsection 3.3, and following the strategy outlined in Table 12, we can complete our 54-step collision attack using the steps illustrated in Figure 4.

Step 1: We randomly generate the first message block to obtain its hash value as the initial state words. Using these initial state words along with the fixed state words Y_0 to

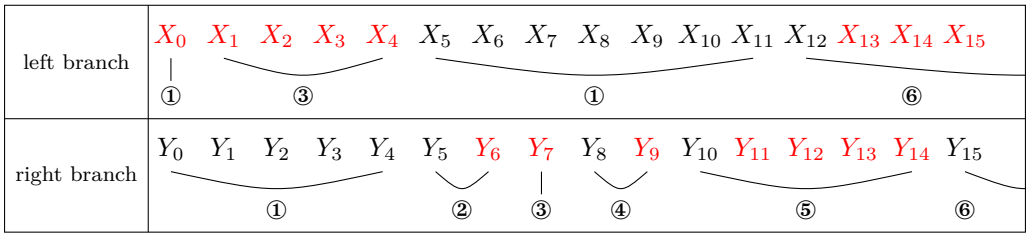

Figure 4: Steps of message modification for 54-step RIPEMD-128

Y_3 and X_5 to X_{11}, we derive the message words m_5, m_{14}, m_7, m_0, m_9, m_{10}, and m_{11}. Finally, we compute X_0 using the step function by message word m_0.

Step 2: We employ the same method used in the 46-step collision attack, utilizing m_2 to satisfy the conditions in Y_5 and Y_6.

Step 3: We will connect the left branch with the fixed state words while ensuring the correctness of all step functions. This is the most complex and crucial step. Specifically, we need to set the message words m_1, m_3, m_4, m_6, and m_8 to derive the state words X_1 to X_4. Since there are no conditions in these state words, we only need to ensure they are consistent with the predetermined message words m_2, m_5, and m_7. By observation, when m_8 is fixed, we can compute X_4 and then use m_7 to compute X_3. Therefore, we can enumerate m_8 and store the corresponding relationship between it and $X_4 \oplus X_3$. Next, we can enumerate m_1 to obtain X_1 and X_2. From the step function, we know that $X_5 = (X_1 + m_5 + (X_2 \oplus X_3 \oplus X_4)) \lll 8$. This implies that $X_3 \oplus X_4 = ((X_5 \ggg 8) - X_1 - m_5) \oplus X_2$. Consequently, we can calculate the right-hand side of the equation and retrieve m_8 from the stored data. Now that we have X_1 to X_4, we can use them to derive m_3, m_4, and m_6. After obtaining a solution, we compute Y_7 by m_4 and verify the 11 conditions in it. Since we have 5 free message words to determine 4 state words, we can obtain 2^{32} solutions for the left branch on average, and $2^{32-11} = 2^{21}$ solutions will be retained.

Step 4: Similar to step 2, we utilize the degrees of freedom in m_{13} to satisfy the conditions in Y_8 and Y_9.

Step 5: We use m_{15} to fulfill conditions in Y_{10}, and then apply multi-step message modification techniques to satisfy 8 conditions in Y_{11} and Y_{12}. Specifically, we set $Y_8[6] = Y_8[12] = Y_8[18] = Y_8[24] = Y_8[26] = Y_8[30] = 1$ to modify 6 conditions in corresponding bit of Y_{11} by Y_{10}, and $Y_9[15] = Y_9[21] = 0$ to modify 2 conditions in corresponding bit of Y_{12} by Y_{10}. We then verify and retain only those results where all other conditions in the state words Y_{10}, Y_{11}, Y_{12} , Y_{13}, and Y_{14} are satisfied.

Step 6: Finally, we utilize the free bits in the message word m_{12} to modify 5 conditions in X_{12} and 3 conditions in different bits (bits $5, 14, 17$) of Y_{15}. We then apply multi-step message modification techniques to satisfy 10 conditions in X_{13} and X_{14}. Specifically, we use bits $j = 11, 12, 13, 15, 16, 26$ of X_{12} to fulfill conditions $X_{13}[j] = X_{12}[j]$ or $X_{13}[j] \neq X_{12}[j]$. Then we use bits $0, 1, 20, 28$ of X_{12} to influence $X_{14}[9]$, $X_{14}[10]$, $X_{13}[27]$ and $X_{13}[3]$ thereby modifying 4 conditions (note that $X_{13}[3]$ satisfy the condition on $X_{14}[3]$). After this step, all message words are determined, and we verify whether this constitutes a 54-step colliding message pair. If all results are tested and no collision is found, we generate another first message block and restart the entire search process.

Theoretical Time Complexity. The time complexity of once step 3 is 2^{32}, yielding 2^{21} solutions. Step 4 requires $2^{21+32-15} = 2^{38}$ time to obtain $2^{21+32-15-10} = 2^{28}$ partial message pairs. The time complexity of step 5 is $2^{28+32-8-8} = 2^{44}$, and since the number of degrees of freedom matches the number of conditions, we retain 2^{28} results. Finally, the time complexity of step 6 is $2^{28+32-5-3-10} = 2^{42}$, producing 2^{42} message pairs for verification. Given that there are still 52 uncontrolled conditions, the expected retry count is 2^{10}. Overall, the complexity of the entire attack is $2^{10} \times (2^{32} + 2^{38} + 2^{44} + 2^{42}) \approx 2^{54}$.

Using the above method, we successfully identified a colliding message pair for the reduced 54-step RIPEMD-128 in approximately 2 hours using 8000 core-groups. The results are presented in Table 13.

Table 13: 54-step colliding message pair

CV_0	67452301	efcdab89	98badcfe	10325476				
M_0	3196321c e6344b00	c183bda8 35f1aec2	5948a4a6 c6eb8b42	b9897a72 5c135c23	952d2821 bf669982	49e99518 916f3793	837f0108 4edc7b52	a78bcb84 066be09b
CV_1	dc0f547e	c91d0a6b	f4664fff	34a8eb58				
M_1	bcc32b21 baf8e6fa	21c52df4 a8c64220	56958c72 2fade74a	355a5a6c ac323113	3f69982d 57446d32	0cc0147a 361f18a8	eaa3ffe9 71313ab1	31d75d55 01ddb358
M_1'	bcc32b21 baf8e6fa	21c52df4 a8c64220	56958c72 2fade74a	355a5a6c ac303113	3f69982d 57446d32	0cc0147a 361f18a8	eaa3ffe9 71313ab1	31d75f4d 05ddb358
H	f465f1aa	dba390ce	a1aaf407	ffe138ab				

6 Conclusion

In this paper, we present new results on collision attacks against the double-branch hash function RIPEMD-128. We introduce new message differences to achieve practical collision attacks for reduced 46-step and 54-step RIPEMD-128, surpassing the previous best results. The 54-step collision attack represents the first collision instance that includes the entire round 3. Specifically, we leverage the properties of message expansion permutations and identify new types of message difference structures to construct high-probability local collisions in round 3. We then search for their corresponding differential characteristics in rounds 1 and 2 based on our message modification strategy. This strategy involves arranging proper order of message modification, allowing us to use the degrees of freedom in one message word to influence many state words. Next, we employ multi-step message modification techniques to control additional conditions and reduce the time complexity of the attack to a feasible level. Overall, the time complexity of 46-step and 54-step collision attacks are approximately 2^{42} and 2^{54}, respectively. Finally, we utilize multi-core computation to obtain the actual colliding message pairs.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (Grant No. 2024YFA1013003), National Cryptographic Science Foundation of China (Grant No. 2025NCSF02014), and Zhongguancun Laboratory.

References

[BP95] Antoon Bosselaers and Bart Preneel, editors. *Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evalu-*

ation RIPE-RACE 1040, volume 1007 of *Lecture Notes in Computer Science*. Springer, 1995.

[DBP96] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In Dieter Gollmann, editor, *FSE'96*, volume 1039 of *LNCS*, pages 71–82. Springer, Berlin, Heidelberg, February 1996.

[IPS13] Mitsugu Iwamoto, Thomas Peyrin, and Yu Sasaki. Limited-birthday distinguishers for hash functions - collisions beyond the birthday bound can be meaningful. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 504–523. Springer, Berlin, Heidelberg, December 2013.

[LP13] Franck Landelle and Thomas Peyrin. Cryptanalysis of full RIPEMD-128. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 228–244. Springer, Berlin, Heidelberg, May 2013.

[MNS12] Florian Mendel, Tomislav Nad, and Martin Schläffer. Collision attacks on the reduced dual-stream hash function RIPEMD-128. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 226–243. Springer, Berlin, Heidelberg, March 2012.

[MPRR06] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. On the collision resistance of RIPEMD-160. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006*, volume 4176 of *LNCS*, pages 101–116. Springer, Berlin, Heidelberg, August / September 2006.

[OSS10] Chiaki Ohtahara, Yu Sasaki, and Takeshi Shimoyama. Preimage attacks on step-reduced RIPEMD-128 and RIPEMD-160. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 6th International Conference, Inscrypt 2010, Shanghai, China, October 20-24, 2010, Revised Selected Papers*, volume 6584 of *Lecture Notes in Computer Science*, pages 169–186. Springer, 2010.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 570–596. Springer, Cham, August 2017.

[Wan14] Gaoli Wang. Practical collision attack on 40-step RIPEMD-128. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 444–460. Springer, Cham, February 2014.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 1–18. Springer, Berlin, Heidelberg, May 2005.

[WSK+11] Lei Wang, Yu Sasaki, Wataru Komatsubara, Kazuo Ohta, and Kazuo Sakiyama. (Second) preimage attacks on step-reduced RIPEMD/RIPEMD-128 with a new local-collision approach. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 197–212. Springer, Berlin, Heidelberg, February 2011.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 19–35. Springer, Berlin, Heidelberg, May 2005.

[WY15] Gaoli Wang and Hongbo Yu. Improved cryptanalysis on RIPEMD-128. *IET Inf. Secur.*, 9(6):354–364, 2015.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 17–36. Springer, Berlin, Heidelberg, August 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 1–16. Springer, Berlin, Heidelberg, August 2005.

A Differential Characteristics and Conditions

Table 14 introduces the symbols used to describe differential characteristics and conditions in Table 15 and Table 16. Here, A and A' represent two colliding state words.

Table 14: Symbols and corresponding differential characteristics and conditions

symbol	condition
.	no condition
0	$A_i[j] = A'_i[j] = 0$
1	$A_i[j] = A'_i[j] = 1$
u	$A_i[j] = 1, A'_i[j] = 0$
n	$A_i[j] = 0, A'_i[j] = 1$
a	$A_i[j] = A'_i[j] = A_{i-1}[j]$
b	$A_i[j] = A'_i[j] \neq A_{i-1}[j]$
c	$A_i[j] = A'_i[j] = A_{i-2}[j]$
d	$A_i[j] = A'_i[j] \neq A_{i-2}[j]$

Table 15 and Table 16 describe our differential characteristics and conditions for the 46-step and 54-step collision attacks, respectively. State words that are fully fixed in the search for differential characteristics are considered to have 32 conditions.

Table 15: 46-step differential characteristics and conditions

step i	X_i	conditions	Y_i	conditions
0	00011100010001111100011111100001	32	.....	0
1	01110001100000011011111100011110	32	.....	0
2	11101100110000000000001110101101	32	.....1110.....	4
3	01111100110000101110nuuuuuuuuu1	32	110.....0000.....000	10
4	nnnnnnnnnnnnnnnnnuun01uuunn0010u	32	1110.....unnn.....000	11
5	11111u0u10111u0unnnn00nuunn00101	32	nnn1.....1....1..0.....unn	10
6	10000n11uuuuu1u011011110nnnnnnnu	32	..1u.....0....0aau..00.....0..	10
7	0010000nuuuuuuu10110111101nuuu01	32	aa11.....n.....11.0....1aa	11
8	.....bud.....c.....abb....aa	9	..nu111111u.....a..un.1.....	13
9	......uaabbbbna.....u..aaab.u	15	..0n1011111.....11.u.....	12
10	.....nu.....n.....u	4	..1unnnnnnn.....0101.....	13
11	.....b.....n.....	2	...0.....0n.....n1u..0....	7
12	.....b.....a.....b	3	...1aaaaa10.....1n...1....	11
13	.....b.....u	2	.....u.....01a..n....	5
14	.....n.....a.....c	3	.....u.....0....0...1....	4
15	.....0.....u.....1	3	.....0...0...n.....0....	4
16	.....1.....0.....	2	.....n...n.....	2
17	.....a.....1.....	2	.....n...0.....	2
18	.....n...a.....	2	.....0.....	1
19	.....0...u.....	2	.....0.....	1
20	.....1...0.....	2	.....n.....	1
21	.....1.....a	2	.....n.....	1
22	.....n	1	.....0.....	1
23	0	1	.....	0
24	1	1	0	1
25	.....	0	.....n	1
26	.....	0	1	1
27	.....	0	1	1
28	.....	0	.....	0
29	.....	0	.....	0
30	.....	0	.....	0
31	0	1	.....	0
32	.....n	1	.....	0
33	1	1	.....	0
34	1	1	.....a	1
35	.....	0	.....n	1
36	.....	0	0	1
37	.....	0	1	1
38	.....	0	.....	0
39	.....	0	.....	0
40	.....	0	.....	0
41	.....	0	.....	0
42	.....	0	.....	0
43	.....	0	.....	0
44	.....	0	.....	0
45	.....	0	.....	0

Table 16: 54-step differential characteristics and conditions

step i	X_i	conditions	Y_i	conditions
0	.....	0	11000000000000111100110111111001	32
1	.....	0	00101111111111111101000101111111	32
2	.....	0	uuuuuuuuuuuuuu1001un101nuuuuuuuu	32
3	.....	0	00001000110100111n00uuu000110100	32
4	.....	0	00001n00110u01100111n0u001001000	32
5	00100101011001111101001111011001	32	..1.u0.n....u.u.11..000..0....u.	13
6	11000101011010110010111101100110	32	..0.1u.0...a0.1.n...0.u..1.0....	12
7	uuuuuuuuuuuuuuunnnnn100uuuuuuuuu	32	..n.11.u....n.u........0.u.10.a.	11
8	nnn0uuuuu0101u1100unnnn101101001	32	.10.u1.10....1n.a..1..a1.1.n1...	15
9	nnnnn0111nuuuu0100unnnn001101101	32	..0....11.0.a.1.0......u.1..n...	10
10	01000u1111101nunuuuuuunu001000001	32	..1.a...n.....u.......00...a1...	8
11	1001un1100101nuuuuuuuun00001n100	32	..n.....1.....u.......1n..0.u...	7
12	....d........u.......un.....n...	5	..1.....0.....n.......u...1.0...	6
13	....bb........aabbbbb...........	9	..0...........n.......0a..u.0...	6
14	..............n.......ab.....b...	4	..............1..0....u...1.n...	5
15	............1..................	1	............01..u....u...0.n...	6
16	............1..................	1	............u...u....0.....0...	4
17	.....a........................	1	............u...0..............	2
18	.....n........................	1	............0..................	1
19	.....0........................	1	.....0.........................	1
20	.....1........................	1	.....u.........................	1
21	............................a..	1	.....u.........................	1
22	............................n..	1	.....0.........................	1
23	............................0..	1	....	0
24	............................1..	1	....	0
25	..............a................	1	....	0
26	..............n................	1	....	0
27	..............0................	1	....	0
28	..............1................	1	....	0
29	....	0	....	0
30	....	0	....	0
31	....	0	............................a...	1
32	....	0	............................n...	1
33	....	0	............................0...	1
34	....	0	............................1...	1
35	....	0	..............a................	1
36	............................0...	1	..............n................	1
37	............................n...	1	..............0................	1
38	............................1...	1	..............1................	1
39	..............0.............0...	2	....	0
40	..............n................	1	....	0
41	..............n................	1	....	0
42	..............0................	1	....	0
43	....	0	....	0
44	....	0	....	0
45	....	0	....	0
46	....	0	....	0
47	....	0	....	0
48	0..............................	1	....	0
49	1..............................	1	....	0
50	u..............................	1	....	0
51	....	0	....	0
52	a..............................	1	....	0
53	....	0	u..............................	1