Differential Attacks on Reduced RIPEMD-160

Florian Mendel[1], Tomislav Nad[2], Stefan Scherz, Martin Schläffer[2]

[1] Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Belgium
[2] Graz University of Technology, IAIK, Austria

Abstract. In this work, we provide the first security analysis of reduced RIPEMD-160 regarding its collision resistance with practical complexity. The ISO/IEC standard RIPEMD-160 was proposed 15 years ago and may be used as a drop-in replacement for SHA-1 due to their same hash output length. Only few results have been published for RIPEMD-160 so far and most attacks have a complexity very close to the generic bound. In this paper, we present the first application of the attacks of Wang et al. on MD5 and SHA-1 to RIPEMD-160. Due to the dual-stream structure of RIPEMD-160 the application of these attacks is nontrivial and almost impossible without the use of automated tools. We present practical examples of semi-free-start near-collisions for the middle 48 steps (out of 80) and semi-free-start collisions for 36 steps of RIPEMD-160. Furthermore, our results show that the differential characteristics get very dense in RIPEMD-160 such that a full-round attack seems unlikely in the near future.

Keywords: hash functions, cryptanalysis, semi-free-start collisions

1 Introduction

In the last decade, several significant advances have been made in the field of hash function research. Especially the collision attacks [16–18] on the MD4 family of hash functions, in particular on MD5 and SHA-1, have weakened the security assumptions of many commonly used hash functions. As a consequence, NIST is organizing the SHA-3 competition to evaluate alternative hash function designs and choose a new hash function standard in 2012 [11]. During this ongoing evaluation, not only the three classical security requirements (preimage resistance, second preimage resistance and collision resistance) are considered. Researchers also analyze (semi-) free-start collisions, near-collisions, and any other non-random behavior of a hash function or its building blocks. Commonly, also simplified or round-reduced variants are studied to get new insights in the design and strength of a cryptographic primitive.

RIPEMD-160 is a cryptographic hash function which was designed by Dobbertin et al. in 1996 [5] and standardized by ISO/IEC in 1997 [6]. As a part of the ISO/IEC 10118-3 standard on dedicated hash functions, RIPEMD-160 is used in many applications and is part of several standards, e.g. OpenSSL, OpenPGP. Furthermore, RIPEMD-160 is often recommended as a drop-in replacement for

D. Gollmann and F.C. Freiling (Eds.): ISC 2012, LNCS 7483, pp. 23–39, 2012.
The original publication is available at http://www.springerlink.com/content/j043317k151536t0/

SHA-1 due to their same output length. Even though RIPEMD-160 relies on the same design principles as MD5 and SHA-1, the dual-stream structure makes RIPEMD-160 more secure against recent attacks on other members of the MD4 family. For this reason, only few results on RIPEMD-160 have been published to date.

The only work regarding the collision resistance of RIPEMD-160 has been published by Mendel et al. [9]. In this work, the application of the differential attacks on RIPEMD by Dobbertin [4] and Wang et al. [16] has been studied. However, due to the increased number of steps and the two streams are more different than in RIPEMD, they concluded that RIPEMD-160 might be secure against these types of attacks. The best currently known attack on the hash function RIPEMD-160 is a preimage attack for 31 (out of 80) steps by Ohtahara et al. [12]. However, the complexity of the attack is very close to the generic complexity of 2^{160}. Recently, Sasaki and Wang [14] have shown non-random properties for up to 51 steps when starting from round 2. However, the complexity of the attack is very high 2^{158} and the attack setting is much weaker than in a collision attack.

In this paper, we provide the first analysis of unmodified RIPEMD-160 against collision attacks. We show how the collision attacks of Wang et al. can be applied on up to 3 rounds of the RIPEMD-160 compression function. We present semi-free-start collisions for 36 steps and semi-free-start near-collisions for 48 steps when starting at round 2. Contrary to all previous results, our results have a very low complexity and we are able to show practical examples in all cases. Although we were not able to attack the reduced round hash function, our results provide a significant improvement in the analysis of the collision resistance of RIPEMD-160 and gives new insights in its security.

The paper is structured as follows. In Sect. 2 we briefly describe the dual-stream hash function RIPEMD-160. In Sect. 3 we present different strategies to construct collisions for round-reduced RIPEMD-160 using local collisions in both streams. In Sect. 4, we show in detail how to find high-probability differential characteristics and confirming inputs using an automatic search tool. Finally, we conclude in Sect. 5.

2 Description of RIPEMD-160

RIPEMD-160 was designed by Dobbertin, Bosselaers and Preneel in 1996 as a replacement for RIPEMD [5] and is part of the international standard ISO/IEC 10118-3:2004 on dedicated hash functions. It is an iterative hash functions based on the Merkle-Damgård design principle [2,10] and produces a 160-bit hash value by processing message blocks of 512 bits. Like its predecessor RIPEMD, the compression function of RIPEMD-160 consists of two parallel streams. The two streams of RIPEMD-160 are designed more differently than those of RIPEMD. In each stream the expanded message block is used to update the state variables. After the computations the results of both streams are combined with the

chaining input, which is illustrated in Fig. 1 and defined as follows:

$$h_0 = B_{-1} + B_{78} + (B'_{77} \lll 10) \qquad h_3 = B_{-4} + (B_{75} \lll 10) + B'_{79}$$
$$h_1 = B_{-2} + (B_{77} \lll 10) + (B'_{76} \lll 10) \qquad h_4 = B_{-5} + B_{79} + B'_{78}$$
$$h_2 = B_{-3} + (B_{76} \lll 10) + (B'_{75} \lll 10)$$

The final values of one iteration $h_0, \ldots, h_4$ are either the final hash value or the chaining input for the next message block.

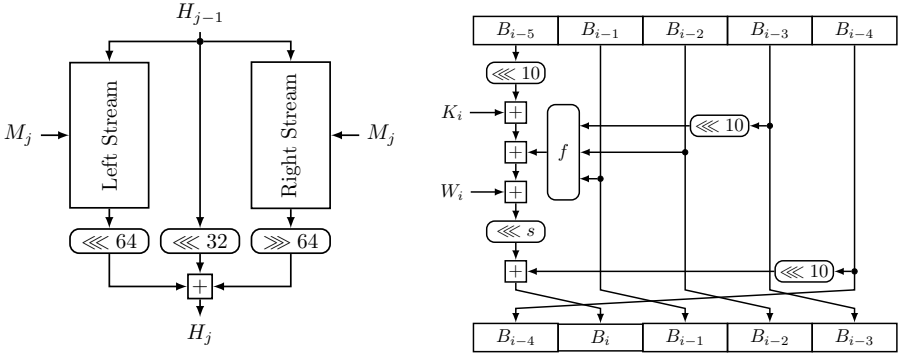

Fig. 1: The compression function (left) and state update transformation (right) of RIPEMD-160.

State Update Transformation. The five 32-bit input state variables of both streams $h_0 = B_{-5} = B'_{-5}$ and $h_i = B_{-i} = B'_{-i}$ with $1 \leq i \leq 4$ are initialized with the initial value (first block) or the previous chaining values. The state update transformation updates the five state variables in five rounds of 16 steps each using one expanded message word W_i in each step.

Note that he Boolean functions f and rotation values s are different in each stream and each round. f_r is used for the r-th round in the left stream and f_{6-r} for the r-th round in the right stream with $1 \leq r \leq 5$. For the rotation values s and the constants K_i we refer to [5].

$$f_1(X,Y,Z) = X \oplus Y \oplus Z \qquad f_4(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$f_2(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z) \qquad f_5(X,Y,Z) = X \oplus (Y \vee \neg Z)$$
$$f_3(X,Y,Z) = (X \vee \neg Y) \oplus Z$$

Message Expansion. The message expansion of RIPEMD-160 is a round-wise permutation of the 16 message block words. For the left and the right stream different permutations are used.

	Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Left Stream	Round 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Round 2	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
	Round 3	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
	Round 4	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
	Round 5	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13
Right Stream	Round 1	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
	Round 2	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
	Round 3	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
	Round 4	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
	Round 5	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

3 Constructing (Local) Collisions for RIPEMD-160

In this section, we provide a general outline how to construct collisions for RIPEMD-160. The idea is based on the recent differential attacks on the MD4 family of hash functions [16, 18] and its application to the dual stream hash function RIPEMD-128 in [8]. The high-level strategy is basically the same in all attacks and can be summarized as follows:

1. Find a characteristic for the hash function that holds with high probability after the first round of the hash function.
2. Find a characteristic (not necessary with high probability) for the first round of the hash function.
3. Use message modification techniques to fulfill conditions imposed by the characteristic in the first round to increase the probability of the characteristic.
4. Use random trials to find values for the remaining free message bits such that the message follows the characteristic.

The most difficult and important part of the attack is to find a good differential characteristic for both the first round and the remaining rounds of the hash function, since this defines the final attack complexity.

3.1 Constructing High-Probability Characteristics for RIPEMD-160

In a differential attack on hash functions, we first need to construct differential characteristics that hold with high probability. In general, a characteristic has a high probability if the number of differences and conditions imposed by the differential characteristic is small. We refer to such a characteristic to be sparse. For single-stream hash functions, a characteristic does not need to be sparse in the first round, since we can use basic message modification [18] to deterministically construct conforming message pairs. However, in the case of dual-stream hash functions, a single message word is used to update two streams in each round which complicates message modification. Therefore, we try to construct sparse characteristics also in at least one stream of the first round such that the message modification part can be carried out more efficiently.

For MD5 a very sparse characteristic with only differences in the MSB of the chaining variable exists, which can be used to construct long high-probability characteristics [1]. RIPEMD-160 consists of two MD5-like step update transformations. However, due to the additional rotation of the state variable B_{i-4} this high-probability characteristic does not exist in RIPEMD-160. Moreover, because of the additional rotation also differences spread rather quickly such that differential characteristics get dense easily. Therefore, also using a linearized approximation of the hash function and algorithms from coding theory (as it is done for instance in the attacks on SHA-1 [13]) does not result in sparse characteristics [9]. The best choice is to use local collisions, which result in large areas which do not contain any differences at all. This strategy is usually advantageous for hash functions using permuted message words in the message expansion.

We start by constructing a very sparse (high probability) characteristic for the hash function after the first round. Then a suitable characteristic for the first round needs to be constructed. The goal is to use one or more message words to construct short local collisions within only a few steps in both streams of the later rounds. This is obviously more difficult for dual-stream hash functions, since more constraints have to be fulfilled. In particular, the different message word permutations and rotation values in each stream of RIPEMD-160 make the construction of many short local collisions difficult.

3.2 Local Collisions

In MD4-like hash functions, a local collision has to start and end by a difference in a message word. We basically have three options to construct local collisions for two rounds which are shown in Fig. 2. First, we can use differences in two message words, to construct local collisions within each round. Second, we can use differences in a single message word, to construct local collisions spanning over two rounds. Thirdly, we can combine the two approaches or use even more message word differences.

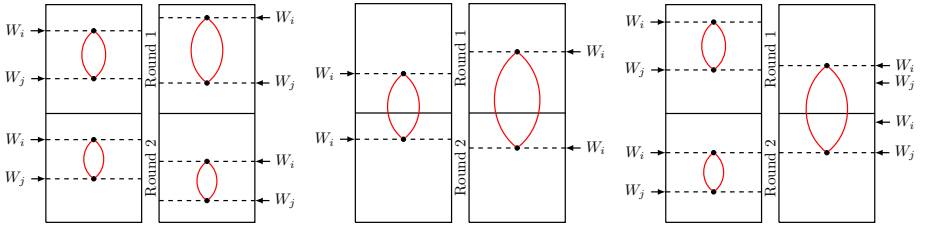

Fig. 2: Three options to construct local collisions for two rounds

Remember that we aim for a high probability differential characteristics after the first round in both streams. This can be achieved by using short local collisions in the second round. Another possibility is to use local collisions spanning

over two rounds and cancel the differences very early in the second round in each stream.

To get high-probability differential characteristics, we aim for local collisions over as few steps as possible. The minimum number of steps for a local collision in RIPEMD-160 is five since a difference has to pass through all 5 state words. A single 5-step local collision can easily be constructed using differences in at least three message words depending on the Boolean function f_r. Assuming that the differences at the input of the Boolean function can always be absorbed, one needs only differences in the message words that are used in step one, four and five of the local collision. Note that if the differences can not be absorbed also differences in the other message words might be needed.

However, even if we have differences in three message words, we need to construct up to four (short) local collisions. This places several constraints on the message words. Possible candidates for message words which contain differences are given in Sect. 3.3.

If differences in a single message word are used, the local collisions have to be constructed over two rounds. The advantage is that we only need two local collisions instead of four. In general, this places less constraints on the message words and may also lead to sparser characteristics in the third round. However, this approach has consequences on the minimum number of steps of a local collision:

Observation 1 *The shortest local collision, which uses difference(s) in a single message word, has to be constructed over six steps.*

The reason for this is the update process of RIPEMD-160. A 5-step local collision only allows non-zero differences in a single state variable, more precisely if a message word W_j is introducing a difference in step i, only the state variable B_i contains a difference. This difference can be canceled five steps later using the same message word. However, at the input of step $i+4$ of the 5-step local collision we get the following setting:

$$\Delta B_{i-1} = 0, \Delta B_{i+3} = 0, \Delta B_{i+2} = 0, \Delta B_{i+1} = 0 \text{ and } \Delta B_i \neq 0 \text{ and } \Delta W_k = 0$$

Then, the following step update transformation has to lead to a zero difference in $\Delta B_{i+4} = 0$ in order to produce a 5-step local collision:

$$\overbrace{B_{i+4}}^{\Delta=0} = \overbrace{(((B_{i-1} \lll 10) + f(B_{i+3}, B_{i+2}, (B_{i+1} \lll 10)) + W_k + K_{i+4}) \lll s)}^{\Delta=0} + \underbrace{(B_i \lll 10)}_{\Delta \neq 0}$$

However, this equation leads to a contradiction since only one term contains a non-zero difference. Hence, a 5-step local collision cannot be constructed using a difference in a single message word. Note that this restriction also applies to local collisions constructed within one round using only differences in two distinct message words.

Since RIPEMD-160 consists of two streams with different permutations of message words, both streams need to be considered concurrently. The first step in the attack is to determine those message words, which may contain differences in order to lead to local collisions. We have several constraints regarding those local collisions such that the whole attack can be carried out efficiently.

3.3 Choosing Message Word Differences

In this section, we describe the different options to construct sparse local collision in RIPEMD-160. The best result and the semi-free-start near-collision on 48 steps (see Sect. 4) was obtained using a difference in a single message word. However, we have also analyzed the use of differences in two or three message words and present local collisions for these cases as well.

Using a single message word difference, we need to construct local collisions between round 1 and 2 of RIPEMD-160. If we consider only a difference in a single message word, we get a sparse differential characteristic in the second round if the local collisions end as early as possible in both streams. Using W_7, we can construct local collisions which end in the first and fourth step of the second round in the left and right stream, respectively. The candidates, which can construct local collisions that end in the first half of round 2 in both streams are given in Table 1.

Table 1: Local collision candidates (single message word)

Message Word	Local Collision Lengths	
	Left Stream	Right Stream
W_7	9 steps (step 7 to 16)	17 steps (step 2 to 19)
W_6	15 steps (step 6 to 21)	7 steps (step 9 to 16)
W_{10}	10 steps (step 10 to 20)	10 steps (step 13 to 23)

Unfortunately, it is very hard to find a corresponding differential characteristic for the first round (also see Section 4). Due to the XOR-function used in round 1 of the left stream and the rather short local collision (9 steps), we did not succeed in finding a corresponding differential characteristic. However, due to the repeating pattern in the message expansion of RIPEMD-160, we can use a local collision of the same length and position between round 2 and 3 using message word W_3 by skipping the first round. Note that this setting is used for the main attack of this paper and is also shown in Fig. 3.

Note that a single 5-step local collision can be constructed easily using differences in multiple message words. Since the Boolean functions in the second round of both streams can absorb the differences, three message words are sufficient to construct 5-step local collisions. However, due to the message permutation, it is not possible to construct 5-step local collisions in the second round of both streams concurrently. We list the shortest local collision for the second round

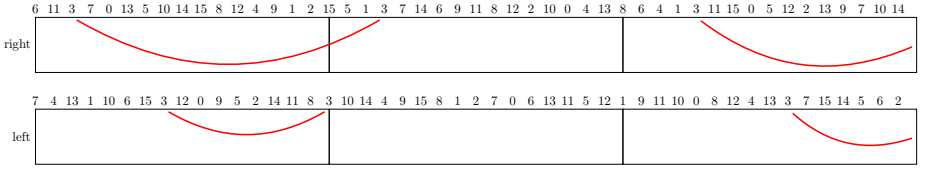

Fig. 3: Using message word W_3 and round 2-4.

and the resulting local collisions for the first round in Table 2. Unfortunately, we did not find high-probability characteristics using differences in three message words.

Table 2: Local collision candidates (triples of message words)

Triples of Message Words	Local Collision Lengths			
	Left Stream		Right Stream	
	Round 1	Round 2	Round 1	Round 2
(W_4, W_{10}, W_{12})	8 steps (4 - 12)	7 steps (17 - 24)	8 steps (7 - 15)	5 steps (23 - 28)
(W_5, W_9, W_{15})	10 steps (5 - 15)	5 steps (22 - 27)	10 steps (0 - 10)	7 steps (22 - 29)
(W_4, W_{10}, W_{12})	20 steps (4 - 24)		8 steps (7 - 15)	5 steps (23 - 28)

Another possibility is to use differences in two distinct message words. Only three message pairs can construct 6-step local collision in round 2 of both streams concurrently and suitable local collisions in round 1. Those pairs of message words are presented in Table 3 as well as some candidates for a combined approach. In such 6-step local collisions, two state variables have to contain differences. Due to different rotation values and the additional modular addition in the state update process, it is not possible to use a single bit difference in both state variables concurrently. Moreover, we need a long carry expansion to cancel differences in the modular addition which results in not so sparse characteristics.

Table 3: Local collision candidates (pairs of message words)

Pairs of Message Words	Local Collision Lengths			
	Left Stream		Right Stream	
	Round 1	Round 2	Round 1	Round 2
(W_0, W_8)	8 steps (0 - 8)	6 steps (25 - 31)	8 steps (3 - 11)	6 steps (20 - 26)
(W_7, W_{15})	8 steps (7 - 15)	6 steps (16 - 22)	8 steps (2 - 10)	6 steps (19 - 25)
(W_3, W_{14})	11 steps (3 - 14)	6 steps (23 - 29)	13 steps (1 - 14)	6 steps (18 - 24)
(W_{12}, W_{13})	12 steps (12 - 24)		7 steps (8 - 15)	6 steps (21 - 27)
(W_7, W_{15})	15 steps (7 - 22)		8 steps (2 - 10)	6 steps (19 - 25)
(W_9, W_{10})	17 steps (9 - 26)		9 steps (4 - 13)	6 steps (23 - 29)

4 Collision Attacks on RIPEMD-160

In this section, we show how to find semi-free-start near-collisions for 48 steps and semi-free-start collisions for 36 steps of RIPEMD-160 using an automated search tool. The results are obtained using the middle 3 rounds of RIPEMD-160 and a single bit difference in message word W_3.

4.1 Automatic Search Tool

To find a differential characteristic and confirming inputs after fixing the message words which contain differences requires an advanced set of techniques and tools. Due to the increased complexity of RIPEMD-160 compared to RIPEMD, finding good (high probability) differential characteristics by hand is almost impossible. Hence, we have used an automatic tool which can be used for finding complex nonlinear differential characteristics as well as for solving nonlinear equations involving conditions on state words and message words. The tool is based on the approach of Mendel et al. to find both complex nonlinear differential characteristics and conforming message pairs for RIPEMD-128 [8] and SHA-256 [7].

The basic idea is to consider differential characteristics which impose arbitrary conditions on pairs of bits using generalized conditions [3]. Generalized conditions are inspired by signed-bit differences and take all 16 possible conditions on a pair of bits into account. Table 4 lists all these possible conditions and introduces the notation for the various cases.

Table 4: Notation for possible generalized conditions on a pair of bits [3].

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

By considering the propagation of these generalized conditions in a bit sliced way we can construct differential characteristics efficiently. The basic idea of the search algorithm is to randomly pick a bit from a set of bit positions with predefined conditions, impose a more restricted condition and compute how this new condition propagates. This is repeated until an inconsistency is found or all unrestricted bits from the set are eliminated. Note that this general approach can be used for both, finding differential characteristics and conforming message pairs.

For example, the search strategy for finding nonlinear characteristics works as follows (for a more detailed description of the search algorithm or how the conditions are propagated we refer to [7]):

1. Define a set of unrestricted bits (`?`) and differences (`x`).
2. Pick a random bit from the set.
3. Impose a zero-difference (`-`) on unrestricted bits (`?`), or randomly choose a sign (`u` or `n`) for differences (`x`).
4. Check how the new conditions propagate.
5. If an inconsistency occurs, remember the last bit and jump back until this bit can be restricted without leading to a contradiction.
6. Repeat from step 2 until all bits from the set have been restricted.

Note that in RIPEMD-160 we have two modular additions (separated by a rotation operation) within one state update. Therefore, two different carry expansions may occur and by only picking random bits of B_i respectively B_i' in the search, the conditions propagate very slowly. Hence, we also consider output bits of the first modular addition and impose more restrictions on these conditions. This way contradictions are detected much earlier, which improves the search significantly.

We use the same strategy to find conforming input pairs for a given differential characteristic. Instead of picking an unrestricted bit (`?`) we pick an undetermined bit without difference (`-`) and assign randomly a value (`0` or `1`) until a solution is found:

1. Define a set of undetermined bits without difference (`-`).
2. Pick a random bit from the set.
3. Randomly choose the value of the bit (`0` or `1`).
4. Check how the new conditions propagate.
5. If an inconsistency occurs, remember the last bit and jump back until this bit can be restricted without leading to a contradiction.
6. Repeat from step 2 until all bits from the set have been restricted.

Note that the efficiency of finding a conforming message pair can be increased if the undetermined bits without difference (`-`) are picked in a specific order. The order strongly depends on the specific hash function. In general, fully determining word after word turns out to be a good approach. It can be used to find solutions without the need for hand-tuned advanced message modification techniques.

4.2 Finding a Differential Characteristic

The starting point for the search tool to find a semi-free-start near-collision on 48 steps is given in Table 6. Note that we do not fix the message difference prior to the search to allow the tool to find a good solution. In order to get a differential characteristics resulting in a low attack complexity, we aim for a low Hamming weight difference in the message words, and hence, also in the state

words after the first 16 steps, i.e. after step 32. Hence, we first search for a good characteristic in this area.

We start by searching for a sparse differential characteristic only in state words B'_{24} and B'_{30} in the right stream. This way the search space gets reduced significantly on one hand, but also the resulting differential characteristics get sparser after step 24 in the right stream. This in turn simplifies finding conforming inputs later. We are able to find a differential characteristics with a single bit differences in W_3 (see Table 7) and only few conditions after step 32, which results in an overall low attack complexity.

We continue the search for the remaining parts of the differential characteristic. Using our search tool, we are able find many differential characteristic for the left and right stream. The differential characteristic for round 2-4 (48 steps) of RIPEMD-160 is given in Table 8.

4.3 Finding a Confirming Message Pair

To fulfill all conditions imposed by the differential characteristic in the first 16 steps (steps 16-31), we need to apply message modification techniques. Since we have many conditions in the first steps of the left and right stream this may not be an easy task. However, using our tool and generalized conditions, we can do message modification for the first 16 steps quite efficiently. Of course, by hand-tuning basic message modification the complexity might be significantly improved, but using our tool this phase of the message search can be automated and still be done quite efficiently. It can be summarized as follows.

- Since the first steps in the right stream are very dense, we start with guessing the remaining free bits in the state words B'_{16} to B'_{22}. This determines the message words W_5 and W_{13}.
- Next we guess the state words B_{26} to B_{16} in the left stream. This determines B_{-1}, B_{-2}, B_{-3} and most of the message words, except W_4, W_{11} and large parts of W_2, W_8.
- Finally, we guess all free bits in the remaining message words to determine the remaining state variables and to find a confirming message pair.

The resulting semi-free-start near-collision for three rounds (steps 16-63) of RIPEMD-160 is given in Table 5. Note that the same input also leads to a collision for 36 steps of RIPEMD-160. Furthermore, one can combine two semi-free-start near-collision as given in Table 5 to construct a second-order differential collision (4-sum) with practical complexity for 48 steps of RIPEMD-160, which also improves the result of Sasaki and Wang in [15] by 8 steps.

5 Conclusions and Future Work

In this work, we have presented new results on the dual-stream hash function RIPEMD-160 standardized by ISO/IEC. To be more precise, we show how the collision attacks of Wang et al. on MD5 and SHA-1, and the recent attack on

Table 5: Semi-free-start near-collision for round 2-4 (48 steps) of RIPEMD-160.

H_0	b23f78a3 7775d378 20806ef8 8d6b662d 4f669598
M_1	2e3d54df e568a9cd d5e45e10 52f4e41a bb1bcda9 ffd073a6 ffe9b7f6 bfe436a9 1273b786 b4ce0002 254a969e 359b7260 817f9eda ef3fff6d bc5068f5 2c3fc390
M_1^*	2e3d54df e568a9cd d5e45e10 52f4f41a bb1bcda9 ffd073a6 ffe9b7f6 bfe436a9 1273b786 b4ce0002 254a969e 359b7260 817f9eda ef3fff6d bc5068f5 2c3fc390
ΔM_1	00000000 00000000 00000000 00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_1	d98051ed e2a12c89 a16b3753 e8764785 1cb36d97
H_1^*	d98053ed e3a16c89 a16b3753 e8764785 1cb36d97
ΔH_1	00000200 01004000 00000000 00000000 00000000

RIPEMD-128 by Mendel et al. can be extended to RIPEMD-160. We have presented practical semi-free-start near-collisions for 48 out of 80 steps and semi-free-start collisions for 36 steps by skipping the first round. Our results improve upon previous results in a number of ways. First, we have increased the number of steps for which (near-) collisions for the compression function of RIPEMD-160 can be found. Second, our attacks have a very low complexity and we are even able to show practical examples.

Unfortunately, it is very hard to find a similar attack including the first round of RIPEMD-160. Due to the XOR-function used in round 1 of the left stream we did not succeed in finding a corresponding differential characteristic. It is part of future work to apply the attack also to round 1-3 which will probably require many improvements in the automated tool. Furthermore, to find (near-) collisions also for the reduced hash function of RIPEMD-160 where the chaining value is fixed, we need to construct sparser local collisions with more freedom to perform message modification.

The ideas and techniques in this paper may also be used in attacks on other hash functions, which update more state variables using a single message word, like SHA-2 or the SHA-3 candidates Blake and Skein.

Acknowledgments

This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II. In addition, this work was supported by the Research Fund KU Leuven, OT/08/027 and by the Austrian Science Fund (FWF, project P21936).

References

1. den Boer, B., Bosselaers, A.: Collisions for the Compressin Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT. LNCS, vol. 765, pp. 293–304. Springer (1993)

2. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO. LNCS, vol. 435, pp. 416–427. Springer (1989)
3. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT. LNCS, vol. 4284, pp. 1–20. Springer (2006)
4. Dobbertin, H.: RIPEMD with Two-Round Compress Function is Not Collision-Free. J. Cryptology 10(1), 51–70 (1997)
5. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE. LNCS, vol. 1039, pp. 71–82. Springer (1996)
6. International Organization for Standardization: ISO/IEC 10118-3:2004. Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions (2004), http://www.iso.org/
7. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT. LNCS, vol. 7073, pp. 288–307. Springer (2011)
8. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128 . In: Canteaut, A. (ed.) FSE. LNCS, Springer (2012), to appear
9. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC. LNCS, vol. 4176, pp. 101–116. Springer (2006)
10. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO. LNCS, vol. 435, pp. 428–446. Springer (1989)
11. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), available online: http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
12. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt. LNCS, vol. 6584, pp. 169–186. Springer (2010)
13. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) IMA Int. Conf. LNCS, vol. 3796, pp. 78–95. Springer (2005)
14. Sasaki, Y., Wang, L.: 2-Dimension Sums: Distinguishers Beyond Three Rounds of RIPEMD-128 and RIPEMD-160. Cryptology ePrint Archive, Report 2012/049 (2012), http://eprint.iacr.org/
15. Sasaki, Y., Wang, L.: Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS. LNCS, vol. 7341, pp. 275–292. Springer (2012)
16. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 1–18. Springer (2005)
17. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO. LNCS, vol. 3621, pp. 17–36. Springer (2005)
18. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT. LNCS, vol. 3494, pp. 19–35. Springer (2005)

A Differential Characteristics

Table 6: Starting characteristic for the attack on round 2-4 of RIPEMD-160.

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	----------			
-4	----------			
-3	----------			
-2	----------			
-1	----------			
16	----------	----------	0	----------
17	----------	----------	1	----------
18	----------	??????????????????????????????????	2	----------
19	----------	??????????????????????????????????	3	??????????????????????????????????
20	----------	??????????????????????????????????	4	----------
21	----------	??????????????????????????????????	5	----------
22	----------	??????????????????????????????????	6	----------
23	??????????????????????????????????	??????????????????????????????????	7	----------
24	??????????????????????????????????	??????????????????????????????????	8	----------
25	??????????????????????????????????	??????????????????????????????????	9	----------
26	??????????????????????????????????	??????????????????????????????????	10	----------
27	??????????????????????????????????	??????????????????????????????????	11	----------
28	----------	??????????????????????????????????	12	----------
29	----------	??????????????????????????????????	13	----------
30	----------	??????????????????????????????????	14	----------
31	----------		15	----------
32	----------	----------		
33	----------	----------		
34	----------	----------		
35	----------	----------		
36	----------	----------		
37	----------	----------		
38	----------	----------		
39	----------	----------		
40	----------	----------		
41	----------	----------		
42	----------	----------		
43	----------	----------		
44	----------	----------		
45	----------	----------		
46	----------	----------		
47	----------	----------		
48	----------	----------		
49	----------	----------		
50	----------	----------		
51	----------	----------		
52	----------	??????????????????????????????????		
53	----------	??????????????????????????????????		
54	----------	??????????????????????????????????		
55	----------	??????????????????????????????????		
56	----------	??????????????????????????????????		
57	??????????????????????????????????	??????????????????????????????????		
58	??????????????????????????????????	??????????????????????????????????		
59	??????????????????????????????????	??????????????????????????????????		
60	??????????????????????????????????	??????????????????????????????????		
61	??????????????????????????????????	??????????????????????????????????		
62	??????????????????????????????????	??????????????????????????????????		
63	??????????????????????????????????	??????????????????????????????????		
	??????????????????????????????????			
	??????????????????????????????????			
	??????????????????????????????????			
	??????????????????????????????????			
	??????????????????????????????????			

Table 7: Characteristic for the attack on round 2-4 of RIPEMD-160 once the differential characteristic after step 24 in the right stream has been fixed.

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	----------			
-4	----------			
-3	----------			
-2	----------			
-1	----------			
16	----------	----------	0	----------
17	----------	----------	1	----------
18	----------	??????????????????????????????????	2	----------
19	----------	??????????????????????????????????	3	----------n----------
20	----------	??????????????????????????????????	4	----------
21	----------	??????????????????????????????????	5	----------
22	----------	??????????????????????????????????	6	----------
23	??????????????????????????????????	??????????????????????????????????	7	----------
24	??????????????????????????????????	-1--nu-u---n-----n----------	8	----------
25	??????????????????????????????????	-1-un-0----------0---0-----u--	9	----------
26	?????????x??????????????????????	------0u---------1---1n-0--00--	10	----------
27	----------??????????????????x--	------n---u--------0--1--n0--	11	----------
28	----------	------0----------un--1-----u--	12	----------
29	----------	------00--n-----1---------1--	13	----------
30	----------	--1----1--0----------u--	14	----------
31	----------	--0----------0--	15	----------
32	----------	----------		
33	----------	----------		
34	----------	----------		
35	----------	----------		
36	----------	----------		
37	----------	----------		
38	----------	----------		
39	----------	----------		
40	----------	----------		
41	----------	----------		
42	----------	----------		
43	----------	----------		
44	----------	----------		
45	----------	----------		
46	----------	----------		
47	----------	----------		
48	----------	----------		
49	----------	----------		
50	----------	----------		
51	----------	----------		
52	----------	-----x----------		
53	----------	----------		
54	----------	----------		
55	----------	----------		
56	----------	----------x----		
57	-----x----------	----------		
58	----------	----------		
59	----------	----------		
60	----------	----------x----------		
61	----------x----	----------		
62	----------x----------	----------		
63	----------	----------		
	----------x----------			

	-------x--------x----------			

Table 8: Differential Characteristic for round 2-4 of RIPEMD-160 resulting in a semi-free-start near-collision.

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	----------			
-4	----------			
-3	----------			
-2	--------1000000----------			
-1	----------			
16	----------	00000000000000000000000000000000	0	----------
17	----------	00001010111111-11111000100011111	1	----------
18	----------	uuuu1uuuuuuuuuuuuuuuuuuuuuuuuu	2	----------
19	----------	01-01101001110101001100011-100--	3	----------n----------
20	---000----------	nu-nu01000uuuu0u0u01101011-unn--	4	----------
21	----------1----------	11-000u0uu10--u1-111-11--10-00nn	5	-----11111--000----------01--
22	---0----------	uu-0nu01------11-11n110--n1-10uu	6	----------
23	---nu----------	----1-01---u-0-1-0n01111-1uuuu10	7	----------
24	---00------nuuuuuuuuuuuu000u---	-1--nu0u0---n011-00n0000--01-----	8	----------1----------
25	000-----10--111111111111111-0000	-10un-0-00-------011000---101u-0	9	----------
26	--101101nu111111--1111111--------	--1---0u0111-----100--1n-00100--	10	----0101----------11--
27	--------00----unnnnnnnnnnnnnnn--	-------n11--u--1---1--0--10-n0--	11	----------
28	----------1000100000000000011	-------0----------un--1------u--	12	----------
29	----0000000001111111----------	-------00---n----11---------1--	13	----11110----------
30	----------0	--1-----1----------u--	14	----------1---10-
31	----------	--0----------0--	15	----------00-----
32	----------	----------		
33	----------	----------		
34	----------	----------		
35	----------	----------		
36	----------	----------		
37	----------	----------		
38	----------	----------		
39	----------	----------		
40	----------	----------		
41	----------	----------		
42	----------	----------		
43	----------	----------		
44	----------	----------		
45	----------	----------		
46	----------	----------		
47	----------	----------		
48	----------	----------		
49	----------	----------		
50	----------	----------		
51	----------	----------		
52	----------	-----x----------		
53	----------	----------		
54	----------	----------		
55	----------	----------		
56	----------	----------x----		
57	-----x----------	----------		
58	----------	----------		
59	----------	----------		
60	----------	----------x----------		
61	----------x----	----------		
62	----------x---------	----------		
63	----------	----------		
	----------x---------			

	-------x--------x----------			