Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128

Florian Mendel[1], Tomislav Nad[2], and Martin Schläffer[2]

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

Abstract. In this paper, we analyze the security of RIPEMD-128 against collision attacks. The ISO/IEC standard RIPEMD-128 was proposed 15 years ago and may be used as a drop-in replacement for 128-bit hash functions like MD5. Only few results have been published for RIPEMD-128, the best being a preimage attack for the first 33 steps of the hash function with complexity $2^{124.5}$. In this work, we provide a new assessment of the security margin of RIPEMD-128 by showing attacks on up to 48 (out of 64) steps of the hash function. We present a collision attack reduced to 38 steps and a near-collisions attack for 44 steps, both with practical complexity. Furthermore, we show non-random properties for 48 steps of the RIPEMD-128 hash function, and provide an example for a collision on the compression function for 48 steps.

For all attacks we use complex nonlinear differential characteristics. Due to the more complicated dual-stream structure of RIPEMD-128 compared to its predecessor, finding high-probability characteristics as well as conforming message pairs is nontrivial. Doing any of these steps by hand is almost impossible or at least, very time consuming. We present a general strategy to analyze dual-stream hash functions and use an automatic search tool for the two main steps of the attack. Our tool is able to find differential characteristics and perform advanced message modification simultaneously in the two streams.

Keywords: hash functions, RIPEMD-128, collisions, near-collisions, differential characteristic, message modification, automatic tool.

1 Introduction

In the last few years, the cryptanalysis of hash functions has become an important topic within the cryptographic community. Especially the collision attacks on the MD4 family of hash functions have weakened the security assumptions of many commonly used hash functions. Still, most of the existing cryptanalytic work has been published for this particular family of hash functions [17,19,20]. In fact, practical collisions have been shown for MD4, MD5, RIPEMD and SHA-0. For SHA-1, a collision attack has been proposed with a complexity of about 2^{63} [18]. However, some members of this family including the ISO/IEC standard RIPEMD-128 (the successor of RIPEMD) seems to be more resistant against these attacks. In this paper, we analyze the security of RIPEMD-128 against collision attacks and show that the security margin is less than expected.

A. Canteaut (Ed.): FSE 2012, LNCS 7549, pp. 226–243, 2012.

Related Work. Since its proposal 15 years ago only a few results have been published for RIPEMD-128. Most published results are concerning the preimage resistance of the hash function [13, 16]. The best currently known attack is a preimage attack for 33 steps and 36 intermediate steps of the hash function with a complexity only slightly faster than the generic complexity of 2^{128} [16]. The only work regarding the collision resistance of RIPEMD-128 has been published by Mendel et al. [11], where the application of the differential attacks on RIPEMD by Dobbertin [5] and Wang et al. [17] is studied. However, due to the increased number of steps and the fact that the two streams are more different than in RIPEMD, they concluded that RIPEMD-128 is secure against this type of attacks.

Our Contribution. In this paper, we first provide a general strategy to analyze dual-stream hash functions in Sect. 2. We analyze different methods to find high-probability differential characteristics which work for both streams. Similar as in the attack on RIPEMD [17], characteristics in two streams are impossible with a high probability. Therefore, in our attacks an automatic search tool is essential for finding valid differential characteristics [4, 10]. This is especially important in the first round of a hash function where characteristics are usually quite dense. In this first round, one usually assumes that conditions imposed by the characteristic can be fulfilled efficiently using message modification techniques. However, message modification is much more difficult in the dual-stream case since two state words are updated using a single message word. This reduced freedom could in general be compensated with hand-tuned advanced message modification techniques [8, 9, 15, 20]. However, another contribution of our work is to provide a fully automatic tool which can be used to find conforming message pairs in the first round of a dual-stream hash function.

Table 1. Summary of our new and previous results on RIPEMD-128

component	attack	steps	complexity	generic	reference
hash	collision	38	example, 2^{14}	2^{64}	Sect. 4
hash	near-collision	44	example, 2^{32}	$2^{47.8}$	Sect. 5.1
hash	non-randomness	48	2^{70}	2^{76}	Sect. 5.2
compression	collision	48	example, 2^{40}	2^{64}	Sect. 5.3
hash	preimage	33	$2^{124.5}$	2^{128}	[13]
hash	preimage	interm. 35	2^{121}	2^{128}	[13]
hash	preimage	interm. 36	$2^{126.5}$	2^{128}	[16]

We apply our attack strategy and tools to the ISO/IEC standard RIPEMD-128 which we describe in Sect. 3. Using our automatic tools, we are able to construct the first practical collisions for up to 38 steps of RIPEMD-128 with a complexity of 2^{14}. We describe the collision attack in details in Sect. 4. The attack can be extended (Sect. 5) to practical near-collisions on 44 steps with complexity 2^{32}. Furthermore, we provide a theoretical distinguisher of the hash function for 48

steps (3 out of 4 rounds) and show that 3 rounds of the RIPEMD-128 compression function are not collision free. Our results are summarized in Table 1, together with all known previous results. Finally, we conclude in Sect. 6 and discuss directions of future work on hash functions with parallel state update transformation.

2 Cryptanalysis of Dual-Stream Hash Functions

In this section, we describe our attack strategy for the cryptanalysis of dual-stream hash functions. The general attack strategy is based on the recent results in cryptanalysis of the MD4-family of hash functions [17, 20]. However, the application of this strategy is nontrivial in the case of dual stream hash functions. Since in each step, one message word is used to update two state words, the freedom of an attacker in finding valid differential characteristics and performing message modification is limited. Hence, a more careful analysis is required to overcome this problem.

2.1 Collision Attacks on Hash Functions

In the following, we first give a brief overview of the attack strategy used in the recent collision attacks on the MD4-family of hash functions [17, 20]. All attacks basically use the same strategy which we adopt for dual-stream hash functions. The high-level strategy 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. This increases 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. There are several methods to find good differential characteristics. The second important part of the attack is to find conforming inputs for the complex nonlinear differential characteristic in the first round of the hash function using message modification techniques.

2.2 Collision Attacks on Dual-Stream Hash Functions

In the following, we will describe our approach to construct good differential characteristics and find colliding message pairs for dual-stream hash functions. We focus on hash functions like RIPEMD-128, but the general idea is applicable to any hash function with two or more streams.

Finding Suitable Differential Characteristics. If the two streams of the hash function are the same except for constant additions, the same differential characteristic can be used in both streams. For instance, in the case of RIPEMD, the permutation and rotation values are indeed equal for both streams. Hence, it is sufficient to find a collision-producing characteristic for only one stream (3 rounds) and apply it simultaneously to both streams [17]. Nevertheless, the number of necessary conditions increases for two streams. Hence, it is more likely to have contradicting conditions. In fact, Wang et al. reported that among 30 selected collision-producing characteristics only one can produce a real collision.

If the two streams are more different, we first need to find a differential characteristic for the hash function after round 1, which holds with a high probability in both streams. One approach is to find such characteristics is to use a linearized model of the hash function and algorithms from coding theory [2, 7, 14]. This works quite well for hash functions with a regular message expansion and step update transformation (like SHA-1), and can be applied to dual-stream hash functions in a straight-forward way.

However, the linearization approach does not work well for hash functions with a permutation of words in the message expansion and different rotation values in the state update transformation (RIPEMD-128 and RIPEMD-160). One usually gets linear differential characteristics with high Hamming weight and hence, a high complexity. However, for such hash functions, we can still make use of the approach of Wang et al. in the attacks on MD4, RIPEMD and MD5 [17, 20]. The idea is to use differences in one or more message words to find local (or inner) collisions within a few steps in the last round(s) of the hash function. Then a suitable characteristic for the remaining steps, preferably also using short local collisions, has to be constructed. Although this is obviously more difficult for dual-stream hash functions, we were able to construct such high-probability differential characteristics for reduced RIPEMD-128 (see Sect. 4.1).

Once, the characteristic after round 1 is fixed we need to find a characteristic (not necessary with high probability) for the first round of the hash function for both streams. Note that in the previous part of the attack it might still be possible to construct inner collisions with hand by choosing the differences carefully. However, to construct a valid nonlinear differential characteristic for both streams in the first round, an automatic search tool is needed. While one can use complex differential characteristics in both streams, we aim for differential characteristics that are sparse in at least one of the two streams, since such sparse characteristics will then also reduce the complexity of the message modification step.

Using Message Modification Techniques. Once we have fixed the differential characteristic for both streams we start with the message search. In the first round, the freedom of the whole message block can be used to get a conforming message pair for the first 16 steps. For single-stream hash function, basic message modification techniques simply choose conforming state words and invert each step update transformation to get the message word [20]. However, as already noted by Wang et al. [17], message modification is more complicated for

two streams since the conditions on two state words need to be fulfilled using a single message word. While in RIPEMD the same message word is used in the same step of the left and right stream, this is not the case in RIPEMD-128, which significantly increases the complexity of message modification.

In the attack on RIPEMD, two techniques have been proposed exploiting the freedom of other message bits using carry effects, the Boolean function and previous message words. The same rotation values in RIPEMD allow an easier application of this idea since it is still possible to fulfill conditions from LSB to MSB. However, for streams with different rotation values, previously corrected conditions may become invalid again. In general, conditions on two state words using a single message word can be fulfilled using advanced message modification techniques. Many dedicated techniques have been proposed in recent years [8, 9, 15, 20], which could also be used to fulfill conditions in the first round of dual-stream hash functions.

To simplify the message modification we use a more general approach. Instead of complicated, dedicated techniques, we use an automated tool for the message modification in the first round. To be more precise, we use the same tool as for the differential path search in the first round. Instead of searching for valid differential characteristics in both streams, we search for valid bit-wise assignments of 0's and 1's to the message and state bits in the first round. Since we solve for conforming message words bit-wise, a different message word permutation, different rotation values and carry effects are handled automatically, similar as in the search for differential characteristics. Moreover, this approach can be generalized to any ARX based design.

The disadvantage of our automated bit-wise approach is a slightly higher complexity, compared to a hand-tuned word-wise approach. However, this increased costs can be amortized by randomizing message words at the end of round 1 to find solutions efficiently for the high-probability characteristic of the remaining rounds.

2.3 Automatic Search Tool

The application of the above strategies is far from being trivial and requires an advanced set of techniques and tools to be successful. Due to the increased complexity of dual-stream hash functions with different streams, finding good differential characteristics by hand is almost impossible. Therefore, we have developed 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 free message bits, i.e. to find confirming message pairs. Our tool is based on the approach of Mendel et al. [10] to find both complex nonlinear differential characteristics and conforming message pairs for SHA-2.

The basic idea is to consider differential characteristics which impose arbitrary conditions on pairs of bits using generalized conditions [4]. Generalized conditions are inspired by signed-bit differences and take all 16 possible conditions on a pair of bits into account. Table 2 lists all these possible conditions and introduces the notation for the various cases.

Table 2. Notation for possible generalized conditions on a pair of bits [4]

(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	-	✓	✓	✓

Using these generalized conditions and propagating them in a bitsliced manner, we can construct complex differential characteristics in an efficient way. 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 [4, 10]):

1. Define a set of unrestricted bits (`?`) and unsigned 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 unsigned 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.

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 for word-wise defined ARX-based hash functions. Using this approach, we can instantly (milliseconds) find solutions for the first round of dual-stream hash functions without the need for hand-tuned advanced message modification techniques.

3 Description of RIPEMD-128

RIPEMD-128 was designed by Dobbertin, Bosselaers and Preneel in [6] as a replacement for RIPEMD. It is an iterative hash functions based on the Merkle-Damgård design principle [3,12] and processes 512-bit input message blocks and produces a 128-bit hash value. To guarantee that the message length is a multiple of 512 bits, an unambiguous padding method is applied. For the description of the padding method we refer to [6].

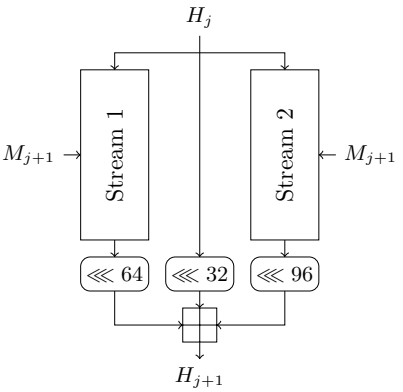

Fig. 1. Structure of the RIPEMD-128 compression function

Like its predecessor, the function of RIPEMD-128 consists of two parallel streams. In each stream the state variables are updated corresponding to the message block and combined with the previous chaining value after the last step, depicted in Figure 1. While RIPEMD consists of two parallel streams of MD4, the two streams are designed differently in the case of RIPEMD-128. In the following, we describe the compression function in detail.

Each stream of the compression function of RIPEMD-128 basically consists of two parts: the state update transformation and the message expansion. Furthermore, RIPEMD-128 consists of a feed-forward where the input and output state words are added in a different order. For a detailed description we refer to [6].

State Update Transformation. The state update transformation of each stream starts from a (fixed) initial value IV of four 32-bit words B_{-4}, B_{-3}, B_{-2}, B_{-1}. and updates them in 4 rounds of 16 steps each. In each step one message word is used to update the four state variables. Figure 2 shows one step of the state update transformation of each stream of RIPEMD-128.

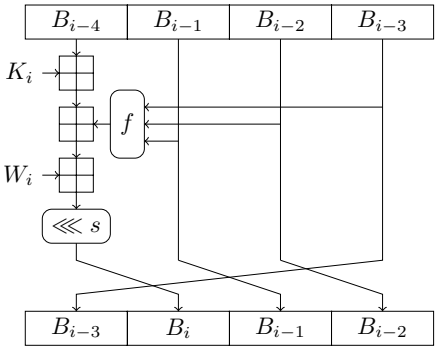

Fig. 2. The step update transformation of RIPEMD-128

The function f is different in each round. f_r is used for the r-th round in the left stream, and f_{5-r} is used for the r-th round in the right stream ($r = 1, \ldots, 4$):

$$f_1(x, y, z) = x \oplus y \oplus z,$$
$$f_2(x, y, z) = (x \wedge y) \vee (\neg x \wedge z),$$
$$f_3(x, y, z) = (x \vee \neg y) \oplus z,$$
$$f_4(x, y, z) = (x \wedge z) \vee (y \wedge \neg z).$$

A step constant K_r is added in every step; the constant is different for each round and for each stream. For the actual values of the constants we refer to [6], since we do not need them in the analysis. For both streams the following rotation values s given in Table 3 are used.

Table 3. The rotation values s for each step and each stream of RIPEMD-128

	Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
left stream	Round 1	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
	Round 2	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
	Round 3	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
	Round 4	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
right stream	Round 1	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
	Round 2	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
	Round 3	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
	Round 4	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8

Message Expansion. The message expansion of RIPEMD-128 is a permutation of the 16 message words in each round. Different permutations are used for the left and the right stream. For both streams the message words are permuted according to Table 4.

Table 4. The index of the message words m_i which are used as the expanded message words W_i in each step and each stream of RIPEMD-128

	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
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

Feed-Forward. After the last step of the state update transformation, the initial values $B_{-4}, \ldots, B_{-1}$ and the output values of the last step of the left stream $B_{63}, \ldots, B_{60}$ and the last step of the right stream $B'_{63}, \ldots, B'_{60}$ are combined, resulting in the final value of one iteration (feed-forward). The result is the final hash value or the initial value for the next message block:

$$B_{-1} \boxplus B_{62} \boxplus B'_{61}$$
$$B_{-4} \boxplus B_{63} \boxplus B'_{62}$$
$$B_{-3} \boxplus B_{60} \boxplus B'_{63}$$
$$B_{-2} \boxplus B_{61} \boxplus B'_{60}$$

4 Collision Attacks on RIPEMD-128

To find collisions in reduced RIPEMD-128 we use the strategy proposed in Sect. 2.2. The attack consists of 3 major parts given as follows:

1. **Starting Point:** Find a good start setting, i.e. differences in only a few specific message words that may lead in a differential characteristic with high probability after step 15.
2. **Differential Characteristic:** Search for a high-probability differential characteristic for the whole hash function where at most one stream has a low probability in step 0-15.
3. **Message Pair:** Find a colliding message pair using automated message modification and random trials.

4.1 Finding a Starting Point

In MD4-like hash functions, differences are introduced and canceled using differences in the expanded message words. Since RIPEMD-128 has two streams with different permutation of message words, the first step in the attack is to determine those message words which may contain differences. We have several constraints such that the whole attack can be carried out efficiently.

First of all, we aim for a high probability differential characteristics after step 15 in both streams. Such high probability differential characteristics can be constructed if the differences introduced by the message words are canceled immediately using local collisions spanning over only a few steps. The shortest local collision in the MD4 step update goes over 4 steps. However, due to the different message permutation used in each stream, it is difficult to achieve short local collisions in both streams simultaneously.

Another possibility is to cancel all differences in each stream as early as possible in round 2 and find message words, such that new differences are introduced late in round 3. A further constraint is to have a short local collision and hence sparse differential characteristic in one stream between step 0-15 such that the message modification part can be carried out more efficiently (see Sect. 2.2).

A single message word which seems to be a good choice is m_{13}. In this case, we get one short local collision between round 1 and round 2 in the left stream and one slightly longer local collision between round 1 and round 2 in the right stream. Both local collisions end in the first few steps of round 2. Furthermore, the message word m_{13} introduces differences very late in the last few steps of round 3 (see Fig. 3). Note that a similar approach was used by Dobbertin in the attack on RIPEMD [5]. Unfortunately, no local collision spanning over 5 steps in the left stream between round 1 and 2 can be constructed which renders the attack impossible.

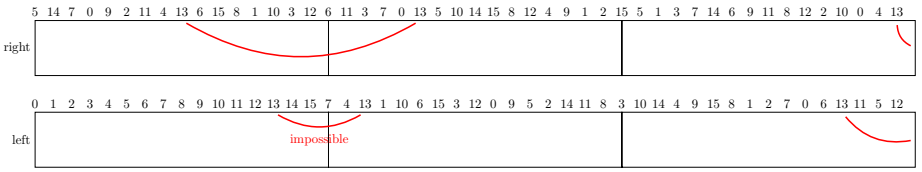

Fig. 3. Using only message word m_{13}

A better choice is to use differences in two message words, like it was done by Wang et al. in the attack on RIPEMD [17]. If we choose differences in m_0 and m_6 then we get for the left stream one local collision over 6 steps in round 1, and another local collision over 4 steps in round 2. Note that in the right stream a short local collision over 4 steps (step 16-20) is actually impossible. This is

due to the fact that for f_3 (ONX-function), a local collision over 4 steps with differences in only two message does not exist. Hence, we combine in the right stream the two local collisions resulting in one long local collision between step 3 and 20. In round 3, the first difference is added in step 38. Hence, using this starting point we can get a collision for 38 steps of RIPEMD-128.

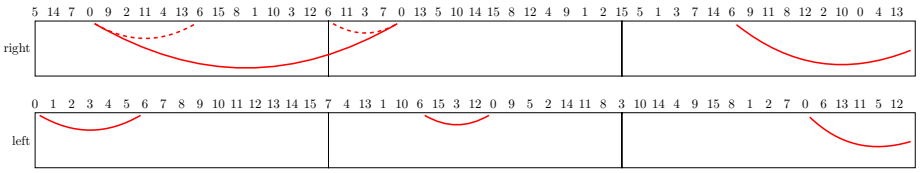

Fig. 4. Using message words m_0 and m_6

4.2 Finding a Differential Characteristic

Once we have fixed the starting point, i.e. the message words which may contain differences, we use an automated tool to find high-probability differential characteristics. Note that we do not fix the message difference prior to the search to allow the tool to find an optimal solution.

In order to get a differential characteristics resulting in a low attack complexity, we aim for a low Hamming weight difference in state word B_{21}. The best we could find is a differential characteristic with 2 differences in B_{21} (see Table 8). Furthermore, the Boolean function XOR in the first round of the left stream provides less freedom in constructing local collisions than the non-linear functions. Hence, we first search for a differential characteristic in the left stream.

Once the characteristic in the left stream is fixed, we use an arbitrary first message block to fulfill the conditions on the chaining value. Since we have 14 conditions on the chaining value (see Table 8), finding the 1st block has a complexity of about 2^{14}.

Next, we search for a differential characteristic in the right stream. To get a low complexity for the message search in round 2, we search for characteristics with only a few differences in state words B'_{14} and B'_{15}. Using our search tool, we can find many differential characteristic for the left and right stream within only a few minutes on an ordinary PC. A colliding differential characteristic for 38 steps of RIPEMD-128 is given in Table 8.

4.3 Finding a Confirming Message Pair

To fulfill all conditions imposed by the differential characteristic in the first round, we need to apply message modification techniques. Since we have many conditions in the first 6 steps of the left stream and the first 15 steps of the 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 efficiently

and immediately within milliseconds on a PC. Of course, by hand-tuning basic message modification the complexity might be improved, but using our tool this phase of the message search can be fully automated. Furthermore, the cost of message modification is fully amortized by randomizing e.g. message word m_{12} to find a solution also for the high-probability characteristic in round 2 (and 3). Using the approximately 2^{30} possible value for m_{12}, we can find a solution for the differential characteristic (complexity 2^{14} after round 1) including message modification in less than a second on our PC. The resulting message pair for a collision on 38 steps of RIPEMD-128 is given in Table 5.

Table 5. Collision for 38 steps of RIPEMD-128

M_1	9431bddf 7b9827d6 f54a64a9 df41a58a fd707a50 dad10eb6 48b0cc76 be66cb8c ab3b7afa 084ba98e ab0a4798 2a4b0d06 a79bf8b7 3fd6008a 4da2112d 849c5b9c
M_2	952bc70f d0840848 eafffa57 0ca3c38a 45383ffb ddc6a9a1 796f1e20 0b9ff55f ddb80113 f0ffe1b5 b7d75dc0 82c7298f f2c442f4 96cbf293 c441d662 06e9eec2
M_2^*	952bc50e d0840848 eafffa57 0ca3c38a 45383ffb ddc6a9a1 79ef1e21 0b9ff55f ddb80113 f0ffe1b5 b7d75dc0 82c7298f f2c442f4 96cbf293 c441d662 06e9eec2
ΔM_2	00000201 00000000 00000000 00000000 00000000 00000000 00800001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_2	a0a00507 fd4c7274 ba230d53 87a0d10a
H_2^*	a0a00507 fd4c7274 ba230d53 87a0d10a
ΔH_2	00000000 00000000 00000000 00000000

5 Extending the Attack to More Steps

In this section, we will show how the collision attack on 38 steps can be extended to more steps of the hash function by using a weaker attack setting, i.e. near-collisions and subspace distinguisher. Furthermore, we present a free-start collision for 48 steps of RIPEMD-128 compression function.

5.1 Near-Collisions for the Hash Function

It is easy to see that by appending 6 steps to the characteristic for 38 steps one gets a near-collision for 44 steps of the hash function with only 6 differences in the hash value. However, note that while in the collision attack one can always append a message block with the correct padding this can not be done for a near-collision. Hence, in order to construct a near-collision for the hash function the padding has to be fixed on beforehand. Luckily, we have such a high amount of freedom in our attack the we can easily fix m_{15}, m_{14} and parts of m_{13} in the attack to guarantee that the padding is correct. The result is a practical near-collision (see Table 6) for 44 steps of RIPEMD-128 with complexity of 2^{32}. Note that the generic attack to find a near-collision with only 6 differences in the hash value has a complexity of about $2^{47.8}$.

Table 6. Near-collision for 44 steps of RIPEMD-128

M_1	2ca95052 425a8f73 08be4537 c790e019 0dcc7d4e 29075123 75327262 8d0d4803 1e57a6a4 73550688 59263eb1 98c6f6ce f03b8b4b 62d3fdf7 638db196 68c0b7b3
M_2	aa1437ef f3646663 c339343a 52c43a1a 779995d5 7b6bd784 e927bb74 5e7cb217 7af2ac15 93392ccf 07e847cf 86318b70 d9d33105 809693dd 000003b8 00000000
M_2^*	aa1435ee f3646663 c339343a 52c43a1a 779995d5 7b6bd784 e9a7bb75 5e7cb217 7af2ac15 93392ccf 07e847cf 86318b70 d9d33105 809693dd 000003b8 00000000
ΔM_2	00000201 00000000 00000000 00000000 00000000 00000000 00800001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_2	92dd7ef7 b1f15ee4 b3e6a250 9db2131b
H_2^*	929d5ef7 b1f15ee4 b3e6a250 bdb21b5f
ΔH_2	00402000 00000000 00000000 20000844

5.2 Non-randomness for the Hash Function

In this section, we show non-random properties for 48 steps (3 rounds) of the hash function. It is based on the differential q-multicollision distinguisher and the differential characteristic for 44 steps which is extended to 48 steps.

Differential q-multicollisions were introduced by Biryukov et al. in the cryptanalysis of the block cipher AES-256 [1]. Note that in [1] the attack is described for a block cipher. However, it can be easily adapted for a hash function. Below we repeat the basic definition and lemma, we need for the attack on RIPEMD-128.

Definition 1. *A set of one difference and q inputs*

$$\{\Delta M; (M^1), (M^2), \cdots, (M^q)\}$$

is called a differential q-multicollision for $h(\cdot)$ if

$$\begin{aligned} h(M^1) \boxminus h(M^1 \boxminus \Delta M) &= h(M^2) \boxminus h(M^2 \boxminus \Delta M) \\ &= \cdots = h(M^q) \boxminus h(M^q \boxminus \Delta M). \end{aligned}$$

The complexity of the generic attack is measured in the number of queries.

Lemma 1. *To construct a differential q-multicollision for an ideal has function with an n-bit output an adversary needs at least*

$$O(q \cdot 2^{\frac{q-1}{q+1} \cdot n})$$

queries on the average for small q.

The proof for Lemma 1 works similar as in [1] for an ideal cipher. Finally, we construct a differential q-multicollision to show non-random properties for RIPEMD-128 reduced to 48 steps. The attack has a complexity of about $4 \cdot 2^{68}$ while the generic attack has a complexity of about 2^{76}.

5.3 Collisions for the Compression Function

When attacking the compression function an adversary has additional the possibility to inject difference in the chaining input. Using this additional freedom

and the same techniques as for the collision attack on the RIPEMD-128 hash function (see Section 4), we can construct a collision for the compression function of RIPEMD-128 reduced to 48 steps. In Table 9 the differential characteristic is shown, resulting in a practical collision for 48 steps of the compression function with a complexity of 2^{40}. The example is given in Table 7.

Table 7. Free-start collision for 48 steps of RIPEMD-128

H_0	5a1d2fbd cd6d40c7 128dd546 900e0e65
H_0^*	5a1927bd edad5cc7 128dd542 900e0e65
ΔH_0	00040800 20c01c00 00000004 00000000
M_1	06083719 9ae0b19b 7ffae1ec 637041ad 28d722d7 fa0082c3 5e78f84e 416ee5e7 faf2b4fc 56738a9f 363c6155 cc7d7ae3 0cb5fc95 b362a16f 6cac81a9 cc11fedd
M_1^*	06083719 9ae0b19b 7ffae1ec 637041ad 28d722d7 fa0082c3 5e78f84e 416ee5e7 faf2b4fc 56738a9f 363c6155 cc7d7ae3 0cb5fc95 b362a16f 6cac81a9 cc11fedd
ΔM_1	00000200 00000000 00000000 00000000 00000000 00000000 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_1	e6428c57 a9f1f589 fc045baf a9cdbc1f
H_1^*	e6428c57 a9f1f589 fc045baf a9cdbc1f
ΔH_1	00000000 00000000 00000000 00000000

6 Conclusions and Future Work

In this work, we have presented new results on the ISO/IEC standard RIPEMD-128, a dual-stream hash function where the message permutation and rotation values are different in the two streams. More specifically, we have presented a collision attack on reduced RIPEMD-128 and get practical collisions for 38 steps of the hash function with a complexity of about 2^{14}. Furthermore, our attack can be extended to near-collisions on 44 steps with complexity 2^{32} and a theoretical distinguisher on the hash function for 48 steps (3 out of 4 rounds) with complexity 2^{70}. Furthermore, we present practical collisions for the RIPEMD-128 compression function, also reduced to 48 steps with complexity 2^{40}.

Apart from these new results, we have outlined a strategy to analyze ARX-based dual-stream hash functions more efficiently. More precisely, we have shown how to automate the most difficult parts of an attack involving more than one stream: finding a differential characteristic and performing message modification in the first round. In particular, message modification had to be hand-tuned or was omitted in previous attacks on ARX-based hash functions. What remains for an attacker is to determine a good starting point (possibly using tools from coding theory) and to assist the tools in the order of guessing words or parts of the state, to improve the overall complexity.

Ideally, these tools can immediately be applied to more complicated hash functions. However, the obtained results depend mainly on the choice of the starting point for the nonlinear tool. If no good starting point can be found or the search space is too large, no attack can be obtained. Future work is to analyze also other, stronger dual-stream hash functions like RIPEMD-160. Furthermore,

the tools and techniques used in this paper can also be applied to other ARX-based hash functions, where more than one state word is updated using a single message word. Examples are 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. Biryukov, A., Khovratovich, D., Nikolić, I.: Distinguisher and Related-Key Attack on the Full AES-256. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 231–249. Springer, Heidelberg (2009)
2. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Linearization Framework for Collision Attacks: Application to CubeHash and MD6. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 560–577. Springer, Heidelberg (2009)
3. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
4. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
5. Dobbertin, H.: RIPEMD with Two-Round Compress Function is Not Collision-Free. J. Cryptology 10(1), 51–70 (1997)
6. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
7. Indesteege, S., Preneel, B.: Practical Collisions for EnRUPT. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 246–259. Springer, Heidelberg (2009)
8. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
9. Klíma, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. IACR Cryptology ePrint Archive 2006, 105 (2006)
10. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
11. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 101–116. Springer, Heidelberg (2006)
12. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
13. 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 2010. LNCS, vol. 6584, pp. 169–186. Springer, Heidelberg (2011)

14. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 78–95. Springer, Heidelberg (2005)
15. Sugita, M., Kawazoe, M., Imai, H.: Gröbner Basis Based Cryptanalysis of SHA-1. IACR Cryptology ePrint Archive 2006, 98 (2006)
16. Wang, L., Sasaki, Y., Komatsubara, W., Ohta, K., Sakiyama, K.: (Second) Preimage Attacks on Step-Reduced RIPEMD/RIPEMD-128 with a New Local-Collision Approach. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 197–212. Springer, Heidelberg (2011)
17. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
18. Wang, X., Yao, A., Yao, F.: New Collision Search for SHA-1. Presented at rump session of CRYPTO (2005)
19. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
20. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Differential Characteristics and Conditions

Table 8. Characteristic for a collision on 38 steps of RIPEMD-128. Bits with gray background have one additional conditions.

i	∇B_i	$\nabla B'_i$	∇m_i
-4	----------	----------	----------
-3	----------	----------	----------
-2	----------	----------	----------
-1	----------	----------	----------
0	-unnnnunnnnnnnn----------	----------	----------u--------u
1	-n----------nuuuunnnnnn	----------0--------0----------	--0----------
2	-unnunnnnnnnnnnnnnnnnnnnnn	----------0--------0----------	----------
3	----------	--0100----u--------u---0110---	----------
4	----------	--1101----1-1-------1---1111---	--0----------
5	----------	--unnn00--1-1-------1---unnn-00	----------
6	----------	--000010--n-u---00--n---0111-10	--------n----------n
7	----------	001nuuuu--0----11111---1001-nu	----------
8	----------	110100----1-----un11n------u---	----------
9	----------	un1n00----------1-unn---1---1---	----------
10	----------	--n0u1----------0-10000-----1---	----------
11	----------	--0nuu----------01n11-----n---	----------
12	----------	--110----------nuuu------------	----------
13	----------	---01-----------11-1-------1---	----------
14	----------	----------00-1--------0	----------
15	----------	----------n--------n	----------
16	----------	----------n--------n	
17	----------	----------0--------0	
18	----------	----------	
19	----------	----------	
20	-----------------------	----------	
21	----------------------n-------n	----------	
22	----------------------0-------0	----------	
23	----------------------1-------1	----------	
24	----------	----------	
25	----------	----------	
26	----------	----------	
27	----------	----------	
28	----------	----------	
29	----------	----------	
30	----------	----------	
31	----------	----------	
32	----------	----------	
33	----------	----------	
34	----------	----------	
35	----------	----------	
36	----------	----------	
37	----------	----------	

Table 9. Characteristic for a free-start collision for 48 steps of RIPEMD-128 compression function. Bits with gray background have one additional conditions.