Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions

Yu Sasaki[1] and Lei Wang[2]

[1] NTT Secure Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585 Japan
sasaki.yu@lab.ntt.co.jp

[2] The University of Electro-Communications
1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan
lei.wang@uec.ac.jp

Abstract. This paper presents differential-based distinguishers against ISO standard hash functions RIPEMD-128 and RIPEMD-160. Second-order differential paths are constructed on reduced steps of their compression functions. These lead to 4-sum attacks on 47 steps (out of 64 steps) of RIPEMD-128 and 40 steps (out of 80 steps) of RIPEMD-160. Then new properties called a *(partial) 2-dimension sum* and *q-multi-second-order collision* are considered. The partial 2-dimension sum is generated on 48 steps of RIPEMD-128 and 42 steps of RIPEMD-160, with a complexity of 2^{35} and 2^{36}, respectively. Theoretically, 2-dimension sums are generated faster than the brute force attack up to 52 steps of RIPEMD-128 and 51 steps of RIPEMD-160, with a complexity of 2^{101} and 2^{158}, respectively. The attacks on RIPEMD-128 can also be regarded as q-multi-second-order collision attacks. The practical attacks are implemented and generated examples are presented. We stress that our results do not impact to the security of full RIPEMD-128 and RIPEMD-160 hash functions.

Keywords: RIPEMD-128, RIPEMD-160, double-branch structure, 2-dimension sum, q-multi-second-order collision.

1 Introduction

Hash functions are taking important roles in various aspects of the cryptography. Since the collision resistance of MD5 and SHA-1 were broken by Wang *et al.* [1,2], cryptographers have looked for stronger hash function designs. While various new designs are discussed in the SHA-3 competition [3], some of existing hash functions seem to have much higher security than the MD4-family. Evaluating such hash functions are useful especially if they are standardized internationally.

RIPEMD-128 and RIPEMD-160 [4] are hash functions standardized by ISO [5]. RIPEMD-160 is also included in the recommended ciphers list of the Cryptography Research and Evaluation Committees (CRYPTREC) set up by the

Japanese Government [6]. RIPEMD-160 is standardized in the SSL protocol [7], and is included in the OpenSSL Cryptography and SSL/TLS Toolkit [8]. RIPEMD-128 and RIPEMD-160 are implemented in various cryptographic libraries. For example, BouncyCastle [9], FlexiProvider [10], and GNU Crypto [11] for JAVA, and Crypto++ [12] for C++. The use of RIPEMD-128 and RIPEMD-160 in HMAC is explained in RFC [13,14].

RIPEMD-128 and -160 adopt the narrow-pipe Merkle-Damgård structure. Their compression functions adopt the double-branch structure, which takes a previous chaining variable H_{i-1} and a message block M_{i-1} as input and computes two compression functions $\mathrm{CF}^{\mathrm{L}}(H_{i-1}, M_{i-1})$ and $\mathrm{CF}^{\mathrm{R}}(H_{i-1}, M_{i-1})$. The output H_i is computed by merging H_{i-1}, $\mathrm{CF}^{\mathrm{L}}(H_{i-1}, M_{i-1})$, and $\mathrm{CF}^{\mathrm{R}}(H_{i-1}, M_{i-1})$. Due to the double size of the internal state and the difficulties of controlling the two functions simultaneously, only a few results were published before. Note that, in order to prevent the recent meet-in-the-middle preimage attacks [15,16,17], some hash functions adopt a structure, which applies the feed-forward function several times, e.g. ARIRANG [18]. Ohtahara *et al.* pointed out that such a structure can be viewed as the double-branch structure [19]. Hence, analyzing the double-branch structure is useful even for the future hash function design.

RIPEMD-128 produces 128-bit digests and its compression function consists of 4 rounds, 64 steps. RIPEMD-160 produces 160-bit digests and its compression function consists of 5 rounds, 80 steps. Mendel *et al.* investigated the differential property of the compression functions of RIPEMD-128 and -160 [20]. They applied the linear approximation and showed low Hamming weight differential paths up to 3 rounds (48 steps) for RIPEMD-128 and up to some intermediate step in the third round (steps 33 – 48) for RIPEMD-160. Although [20] is useful to obtain some intuition for collision attacks, a lot of work is necessary to complete the attacks. The complexity and even the possibility of the attacks are unclear. Other previous results are the ones by Ohtahara *et al.* [21] and Wang *et al.* [22], which investigated preimage attacks. [21] showed that the first 33 steps of RIPEMD-128 and the first 31 steps of RIPEMD-160 can be attacked while [22] showed that intermediate 36 steps of RIPEMD-128 can be attacked.

In this paper, boomerang type differential properties are discussed. The boomerang attack was first proposed by Wagner for analyzing block-ciphers [23]. It divides the function $E(\cdot)$ into two subparts E_0 and E_1 such that $E(\cdot) = E_1 \circ E_0(\cdot)$. Let the probabilities of differential paths for E_0 and E_1 be p and q, respectively. The boomerang attack exploits the fact that a second order differential property with a probability p^2q^2 exists for the entire function E. Aumasson *et al.* [24] applied the boomerang attack to the internal cipher of the hash function Skein. However, the goal of the attack is still recovering the secret key. After that Birukov *et al.* [25] and Lamberger and Mendel [26] independently applied this property on the compression function so as to mount distinguishers. Then, Sasaki [27] showed a straight-forward application of the framework of [25,26] to the MD4-family (using the single-branch structure) consisting of up to 5 rounds. With the straight-forward technique by [27], it is unclear how to attack the single-branch structure consisting of more than 5 rounds.

Our Contributions

This paper presents differential distinguishers against the compression functions of RIPEMD-128 and RIPEMD-160. The results are summarized in Table 1.

Our first target is the 4-sum property. Then, new differential property called a *2-dimension sum* and a *q-multi-second-order collision* are considered. Note that the partial 4-sum and partial 2-dimension sum can be introduced naturally.

Then, differential paths are constructed on reduced RIPEMD-128 and -160 compression functions. The differential path construction is based on the framework of the boomerang distinguisher by [25,26]. Our strategy is regarding CF^{L} as the first part of the boomerang attack (E_0) and CF^{R} as the second part (E_1), hence the entire function (E) is viewed as the single-branch structure with 8 and 10 rounds for RIPEMD-128 and -160, respectively. This simplifies the differential path construction because the differential paths for CF^{L} and CF^{R} can be analyzed almost independently. However, because the straight-forward application of the framework to the MD4-family [27] can only work up to 5 rounds, several improvements are necessary to extend the number of attacked rounds.

On RIPEMD-128, we use the local collision to construct differential paths. This leads to a long differential path satisfied with a high probability. As a result, 4-sums and partial 2-dimension sums are generated with a practical complexity up to 47 and 48 steps, respectively. In addition, 2-dimension sums are theoretically generated faster than the brute force attack up to 52 steps.

On RIPEMD-160, the local collision involves more message words than RIPEMD-128, and thus using the local collision is inefficient. Instead, we show an interesting non-linear differential property of RIPEMD-160 which can avoid the quick propagation of the difference. As a result, 4-sums and partial 2-dimension sums are generated with a practical complexity up to 38 and 40 steps, respectively. In addition, 2-dimension sums are theoretically generated faster than the brute force attack up to 43 steps. If the attack target starts from the second round, the numbers of attacked steps become 40, 42, and 51 for 4-sums, partial 2-dimension sums, and theoretical 2-dimension sums.

Paper Outline. In Sect. 2, differential properties are discussed. In Sect. 3, the specification of RIPEMD-128 and -160 are explained. In Sect. 4, attacks on RIPEMD-128 are explained. In Sect. 5, attacks on RIPEMD-160 are explained. Finally, the paper is concluded in Sect. 6.

2 Differential Properties to Be Distinguished

We summarize differential properties discussed in previous papers, which are 4-sums and second-order collisions, and introduce new properties called *2-dimension sums* and *q-multi-second-order collision*.

Table 1. Attacks on the compression functions. "2D-sum" and "q-multi-2nd" denote 2-dimension sum and q-multi-second-order collision respectively.

Target	#Steps	Property	Information Theoretic Bound	Generic Attack	Complexity	Reference
RIPEMD-128	33	preimage	2^{128}	2^{128}	2^{119}	[21]
(total 64 steps)	36†	preimage	2^{128}	2^{128}	2^{123}	[22]
	45	4-sum	2^{32}	2^{42}	2^{27}	Ours
	47	4-sum	2^{32}	2^{42}	2^{39}	Ours
	48	q-multi-2nd	2^{55} for $q = 17$	-	2^{53}	Ours
	52	q-multi-2nd	2^{108} for $q = 53$	-	2^{107}	Ours
RIPEMD-160	31	preimage	2^{160}	2^{160}	2^{148}	[21]
(total 80 steps)	38	4-sum	2^{40}	2^{53}	2^{42}	Ours
	40	partial 2D sum	2^{64}	2^{64}	2^{42}	Ours
	43	2D sum	2^{160}	2^{160}	2^{151}	Ours
	40‡	4-sum	2^{40}	2^{53}	2^{36}	Ours
	42‡	partial 2D sum	2^{64}	2^{64}	2^{36}	Ours
	51‡	2D sum	2^{160}	2^{160}	2^{158}	Ours

†: The attacked steps start from an intermediate step.
‡: The attacked steps start from the second round.

2.1 Previously Discussed Properties

4-sum is a set of 4 different inputs (I_0, I_1, I_2, I_3) where the sum of the corresponding outputs is 0, namely $\mathrm{CF}(I_0) \oplus \mathrm{CF}(I_1) \oplus \mathrm{CF}(I_2) \oplus \mathrm{CF}(I_3) = 0$. If the function is ideal, finding 4-sums requires at least $2^{n/4}$ queries for n-bit output. Therefore, if the 4-sum is obtained faster than $2^{n/4}$ computations, CF is regarded as non-ideal. Apart from the information theoretic bound ($2^{n/4}$), the current best generic attack to find 4-sums is a generalized birthday attack [28], which requires $2^{n/3}$ computations and $2^{n/3}$ memory. Hence, if 4-sums are generated with a complexity between $2^{n/4}$ and $2^{n/3}$, CF is said to be weak because the same property cannot be detected on other functions with the current knowledge.

The second-order collision [29,26] is a special form, in other words, a subset of the 4-sum. It can be viewed as limiting the form of input values on the 4-sum property. [29,26] defined the derivative at a point α for a function f as

$$\Delta_{(\alpha)} f(y) = f(y + \alpha) - f(y).$$

Then, the second-order derivative[1] at (α, β) is defined as

$$\begin{aligned}\Delta_{(\alpha,\beta)} f(y) &= \Delta_{(\beta)}(\Delta_{(\alpha)} f(y)) \\ &= f(y + \alpha + \beta) - f(y + \beta) - f(y + \alpha) + f(y).\end{aligned}$$

The second-order collision attack is finding (α, β, y) such that $\Delta_{(\alpha,\beta)} f(y) = 0$. Previous work [29,26] showed that the information theoretic bound is $3 \cdot 2^{n/3}$

[1] In [29,26], the n-th order derivative is discussed rather than the specific case $n = 2$.

because the problem is essentially finding three parameters α, β, y with an n-bit relation. On the other hand, the current best generic attack requires $2^{n/2}$.

2.2 2-Dimension Sums and Suitability for Double-Branch Structure

Similarly to the framework of the rebound attack [30], which limits the input and output differences before making any query, we introduce a new differential property, which we call *2-dimension sums*. The 2-dimension sum is a special form, in other words, a subset of the second-order collision. We further introduce limitations to the form of input values on the second-order collision. The problem is changed to find a value y such that $\Delta_{(\alpha,\beta)} f(y) = 0$ for two pre-specified values α and β. The 2-dimension sum is different from the second-order collision only in the sense that α and β are pre-specified. The information theoretic bound for the 2-dimension sum is 2^n because the problem is essentially finding an n-bit value satisfying an n-bit condition. A generic attack for this problem is also 2^n computations; choose the value of y and check that the corresponding $\mathrm{CF}(y) \oplus \mathrm{CF}(y \oplus \alpha) \oplus \mathrm{CF}(y \oplus \beta) \oplus \mathrm{CF}(y \oplus \alpha \oplus \beta)$ is 0.

The 2-dimension sum is particularly useful to attack the double-branch structure. The attacker can construct a pseudo-near-collision path for CF^{L} with setting input chaining variable difference to α. Then, a pseudo-near-collision path for CF^{R} is independently constructed with setting other difference β. If the product of the probability of each path (after the message modification) is higher than $2^{-n/2}$, the 2-dimension sum can be generated faster than 2^n by using the boomerang attack approach [25,26]. Different from the original RIPEMD, RIPEMD-128 and -160 adopt very different functions as CF^{L} and CF^{R}. Therefore, the independence of the path construction for CF^{L} and CF^{R} greatly helps the attacker. More detailed discussion is given in Sect. 4, and 5.

Note that the *partial 2-dimension sum* is naturally introduced, where $\mathrm{CF}(y) \oplus \mathrm{CF}(y \oplus \alpha) \oplus \mathrm{CF}(y \oplus \beta) \oplus \mathrm{CF}(y \oplus \alpha \oplus \beta)$ becomes 0 only for the specified partial bits, say d bits. In this case, the complexity of the generic attack is 2^d and thus a valid distinguisher must find it faster than 2^d computations.

2.3 *q*-Multi-second-order Collision

By following the framework of the q-multicollision [31], we introduce a notion of a q-multi-second-order collision on a function $f : \{0,1\}^X \rightarrow \{0,1\}^Y$, which is a set of two non-zero differences and q distinct inputs $\{\Delta, \nabla, x_1, x_2, \cdots, x_q\}$ satisfying

$$f(x_1) - f(x_1 + \Delta) + f(x_1 + \Delta + \nabla) - f(x_1 + \nabla) = 0,$$
$$\cdots$$
$$f(x_q) - f(x_q + \Delta) + f(x_q + \Delta + \nabla) - f(x_q + \nabla) = 0.$$

Information-Theoretic Bound. Let an adversary make k distinct queries $x_1, \ldots x_k$ to a random function. For any specified Δ and ∇, k distinct queries at most contribute to k quartets as below; $(x_1, x_1 + \varDelta, x_1 + \nabla, x_1 + \varDelta + \nabla), \ldots, (x_k, x_k + \varDelta, x_k + \nabla, x_k + \varDelta + \nabla)$. So the number of q-tuple quartets is $\binom{k}{q}$ for any specified $\varDelta$ and ∇. There are less than 2^{2X} values for $(\varDelta, \nabla)$. Thus in total the number of q-tuple quartets satisfying the form is at most $2^{2X} \cdot \binom{k}{q}$. One such q-tuple has a probability $2^{-q \times Y}$. We get the following inequality.

$$\begin{aligned} 2^{2X} \cdot \binom{k}{q} &\geq 2^{qY}, \\ \frac{k(k-1)\cdots(k-(q-1))}{q!} &\geq 2^{qY-2X}, \\ k &> \sqrt[q]{q!} \cdot 2^{Y - \frac{2X}{q}}. \end{aligned} \tag{1}$$

2.4 Remarks for the Motivation of Studying Weak Properties

Some may say that studying weak non-ideal properties such as the partial 2-dimension sum is meaningless. In fact, compared to the collision, the impact of finding weak differential properties are very limited. However, the security of symmetric primitives is usually evaluated and get trusted by demonstrating many cryptanalytic attempts. Therefore, we believe that not only investigating the standard properties but also extending the number of steps as much as possible with any non-ideal property is useful to understand the state-of-the-art about the security. Especially, such an activity is important for RIPEMD-128 and RIPEMD-160 because they are standardized and implemented in various environments but only a few cryptanalyses were presented so far. This paper is not claiming that weak distinguishers working for more steps are better than standard attack scenarios with a smaller number of attacked steps. Considering various approaches leads to better understanding and may be useful in future.

3 Specifications

RIPEMD-128/-160 were proposed by Dobbertin *et al.* [4] as stronger hash functions than RIPEMD [32]. They take a message of arbitrary length as input and produce 128-bit and 160-bit hash digests, respectively. Because our attack target is the compression function, we omit the description of the domain extension.

3.1 RIPEMD-128

The compression function of RIPEMD-128 takes a 128-bit chaining variable H_{i-1} and a 512-bit message block M_{i-1} as input and outputs a 128-bit chaining variable H_i. M_i is divided into sixteen 32-bit message words $m_0, m_1, \ldots, m_{15}$. Let p_j^L be a 128-bit chaining variable and a_j^L, b_j^L, c_j^L and d_j^L be 32-bit variables satisfying $p_j^L = a_j^L \| b_j^L \| c_j^L \| d_j^L$, where $0 \leq j \leq 64$. Similarly, p_j^R and $a_j^R, b_j^R, c_j^R, d_j^R$ are defined. The computation for $\mathrm{CF^L}$ is as follows.

$$p_0^L \leftarrow H_{i-1}, \qquad p_{j+1}^L \leftarrow \mathrm{SF}_j^L(p_j^L, m_{\pi^L(j)}) \text{ for } j = 0, 1, \ldots, 63,$$

Table 2. Boolean functions, message expansions, and rotation numbers

	$f_x(X,Y,Z)$
$x=0,\ldots,15$	$X \oplus Y \oplus Z$
$x=16,\ldots,31$	$(X \wedge Y) \vee (\neg X \wedge Z)$
$x=32,\ldots,47$	$(X \vee \neg Y) \oplus Z$
$x=48,\ldots,63$	$(X \wedge Z) \vee (Y \wedge \neg Z)$
$x=64,\ldots,79$	$X \oplus (Y \vee \neg Z)$

	$\pi^L(j)$																$\pi^R(j)$															
$j=0,\ldots,15$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
$j=16,\ldots,31$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
$j=32,\ldots,47$	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
$j=48,\ldots,63$	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
$j=64,\ldots,79$	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

Table 3. Computations for the Output of the Compression Function

	RIPEMD-128	RIPEMD-160
$H_i^{(a)}$	$H_{i-1}^{(b)} + c_{64}^L + d_{64}^R$	$H_{i-1}^{(b)} + c_{80}^L + d_{80}^R$
$H_i^{(b)}$	$H_{i-1}^{(c)} + d_{64}^L + a_{64}^R$	$H_{i-1}^{(c)} + d_{80}^L + e_{80}^R$
$H_i^{(c)}$	$H_{i-1}^{(d)} + a_{64}^L + b_{64}^R$	$H_{i-1}^{(d)} + e_{80}^L + a_{80}^R$
$H_i^{(d)}$	$H_{i-1}^{(a)} + b_{64}^L + c_{64}^R$	$H_{i-1}^{(e)} + a_{80}^L + b_{80}^R$
$H_i^{(e)}$	$-$	$H_{i-1}^{(a)} + b_{80}^L + c_{80}^R$

where SF_j^L is a step function for $\mathrm{CF^L}$ and performs the following computation.

$$\begin{aligned} a_{j+1}^L &\leftarrow d_j^L, & b_{j+1}^L &\leftarrow (a_j^L + f_j(b_j^L, c_j^L, d_j^L) + m_{\pi^L(j)} + k_j^L) \lll s_j^L, \\ c_{j+1}^L &\leftarrow b_j^L, & d_{j+1}^L &\leftarrow c_j^L, \end{aligned}$$

where '+' represents the addition on modulo 2^{32}, '$\lll s$' represents left cyclic shift by s bits, f_x is a Boolean function, $\pi^L(j)$ is the message expansion, and k_j^L is a constant. $\mathrm{CF^R}$ is similarly described. The values of $s_j^R, \pi^R(j), k_j^R$ are different and f_{63-j} is used in step j. Details of $f_x, \pi^L(j), \pi^R(j)$ are in Table 2. Finally, the output $H_i = H_i^{(a)} \| H_i^{(b)} \| H_i^{(c)} \| H_i^{(d)}$ is computed as shown in Table 3.

3.2 RIPEMD-160

The compression function of RIPEMD-160 is almost the same as the one for RIPEMD-128. The chaining variable size is 160 bits, and thus 160-bit intermediate states are represented by five 32-bit variables, e.g. $p_j^L = a_j^L \| b_j^L \| c_j^L \| d_j^L \| e_j^L$. The step functions SF^L and SF^R are iteratively computed 80 times ($0 \leq j \leq 79$). The details of the computation of SF^L are as follows.

$$\begin{aligned} a_{j+1}^L &\leftarrow e_j^L, & b_{j+1}^L &\leftarrow ((a_j^L + f_j(b_j^L, c_j^L, d_j^L) + m_{\pi^L(j)} + k_j^L) \lll s_j^L) + e_j^L, \\ c_{j+1}^L &\leftarrow b_j^L, & d_{j+1}^L &\leftarrow c_j^L \lll 10, \quad e_{j+1}^L \leftarrow d_j^L. \end{aligned}$$

Most of the parameters are shared with RIPEMD-128. The details are described in Table 2. In the computations of SF^R, the Boolean function in step j is $f_{79-j}(b_j^R, c_j^R, d_j^R)$. The other computations are similarly specified as SF^L. Finally, the output chaining variable H_i is computed as shown in Table 3.

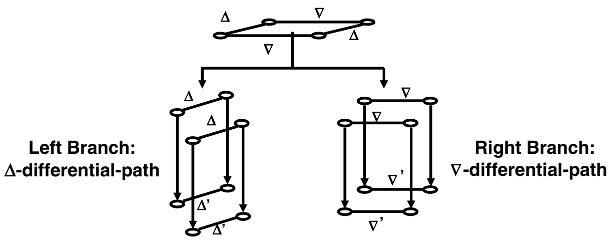

Fig. 1. Our Attack Strategy on RIPEMD-128 and -160

4 Attacks on RIPEMD-128

We construct differential distinguishers against compression function of RIPEMD-128. Hereafter, we compute the difference in modular subtraction because the main operation of RIPEMD-128/-160 is the modular addition. The message differences, differential path, and sufficient conditions against RIPEMD-128 are shown in Tables 4, 5 and 6 respectively.

4.1 Overall Strategy

A graphical description of our strategy is given in Fig. 1. First, we construct a $\varDelta$-differential-path $\varDelta \xrightarrow{\varDelta^M} \varDelta'$ in the left branch, and a ∇-differential-path $\nabla \xrightarrow{\nabla^M} \nabla'$ in the right branch. Then we try to search for an input of the compression function (H, M) such that (H, M), $(H + \varDelta, M + \varDelta^M)$, $(H + \nabla, M + \nabla^M)$, and $(M + \varDelta + \nabla, H + \varDelta^M + \nabla^M)$ satisfy the following conditions.

- The difference propagations between (H, M) and $(H + \varDelta, M + \varDelta^M)$ and between $(H + \nabla, M + \nabla^M)$ and $(H + \nabla + \varDelta, M + \nabla^M + \varDelta^M)$ follow $\varDelta$-differential-path in the left branch.
- The difference propagations between (H, M) and $(H + \nabla, M + \nabla^M)$ and between $(H + \varDelta, M + \varDelta^M)$ and $(H + \varDelta + \nabla, M + \varDelta^M + \nabla^M)$ follow ∇-differential-path in the right branch.

For such a (H, M), we obtain the relationship; $\mathrm{CF}(H, M) + \mathrm{CF}(H + \varDelta + \nabla, M + \varDelta^M + \nabla^M) - \mathrm{CF}(H + \varDelta, M + \varDelta^M) - \mathrm{CF}(H + \nabla, M + \nabla^M) = 0$, where CF is the compression function of RIPEMD-128.

Note that the terminology "4-sum" is somehow strange to discuss the above modular subtraction. However, the terminology "second-order collision" usually considers the limitation of the inputs, and discusses essentially different properties. Hence, to avoid the confusion, we use "4-sum" for the above property.

4.2 Constructing $\varDelta$-Differential-Path

We should keep the differential path as simple as possible in order to maximize its probability. One natural approach is to restrict the difference propagations. Particularly, we expect that f functions do not produce new differences.

Table 4. Differential path construction for 3-round RIPEMD-128

round	$\pi^L(j)$	$\pi^R(j)$
1	⓪ 1 2 3 4 5 ⑥ 7 8 9 10 11 12 13 14 15	5 14 7 0 9 2 11 4 13 ⑥ 15 8 1 10 3 12
	$\varDelta$ MM $\leftarrow$ $\varDelta$ | constant	MM $\leftarrow$
2	7 4 13 1 10 ⑥ 15 3 12 ⓪ 9 5 2 14 11 8	⑥ 11 3 7 0 13 5 10 14 15 8 12 4 9 1 2
	constant | $\leftarrow$ LC $\rightarrow$ | constant	∇ | constant
3	3 10 14 4 9 15 8 1 2 7 ⓪ ⑥ 13 11 5 12	15 5 1 3 7 14 ⑥ 9 11 8 12 2 10 0 4 13
	constant | $\varDelta$ $\varDelta$ $\rightarrow$	constant | ∇ $\rightarrow$

The f functions of the first and the third rounds in the left branch have weak absorption property. By weak absorption property, we mean that a bit difference is produced by f with the probability 1 if a (particular) input variable has a difference at that bit. So we must make the $\varDelta$-differential-path short in these two rounds. In order to achieve it, we generate a *local collision* in the second round of the left branch, and pick a message difference $\varDelta^M$ which appears at a very beginning step in the first round and at a very late step in the third round. Such a strategy maximizes the probability of the whole differential path. Finally we choose the message difference $\varDelta^M$ as below

$$\varDelta m_0 = -2^{10}; \text{ and } \varDelta m_6 = 2^{31}.$$

$\varDelta$ is determined backwards according to the differential path in the first round.

$$\varDelta a_0 = 2^8; \ \varDelta b_0 = 0; \ \varDelta c_0 = 2^{31} + 2^{16}; \text{ and } \varDelta d_0 = 2^{31} + 2^{16}.$$

$\varDelta'$ changes with the number of the attacked steps. Moreover, we do not specify $\varDelta'$ according to *amplified probability* using multiple outside differential paths.

Remarks on Multiple Differential Path. At step 44 of $\varDelta$-differential-path, a difference of $*2^5$ is produced by the f function. For this difference, we do not limit the sign for each pair, but we need the condition that the signs are identical between two pairs. Hence, the probability to satisfy this condition is 2^{-1} rather than 2^{-2}. We also set 2 similar conditions at steps 46 and 47.

4.3 Constructing ∇-Differential-Path

The f function of the second round in the right branch has weak absorption property. So we must make ∇-differential-path short in the second round of the right branch. At the same time, the f function in the first round of the right branch has strong absorption property. By strong absorption property, we mean that no bit difference is produced by f by setting conditions if only one input variant has a difference on that bit. So we decide to generate a relatively long but simple sub-path in the first round, which should be ended by the message difference in the second round. We should pick a message difference which appears at a very beginning step in the second round and at a very late step in the third round. The whole differential path consists of 2 sub-paths: a long path going through the whole first round and ending at a beginning step in the second round; and a short path at the late steps in the third round. Finally we choose the message difference ∇^M and corresponding ∇ as below

$$\nabla m_6 = -2^{30};\ \nabla a_0 = 2^{20};\ \nabla b_0 = 0;\ \nabla c_0 = 0;\ \text{and } \nabla d_0 = 2^6.$$

4.4 Searching for (H, M)

Firstly, we search for (H, M)s which satisfy the differential path for the first 17 steps in both branches. The complexity of finding such (H, M)s, i.e. the complexity for satisfying the first 17 steps can be ignored by applying the message modification technique (e.g. [1,2]) and optimizing the computation order. More precisely, the message modification is a technique to efficiently satisfy all conditions in the first round. It exploits the property that each step in the first round is computed with a message word which is not fixed yet. For example, to satisfy the conditions of the variable b_{j+1} in the first round, you can iterate the computation in step j many times by only changing the value of $m_{\pi(j)}$ without influencing the previous steps. Hence, by satisfying the conditions step by step, the complexity is greatly reduced. Moreover, modifying the message word m_{12} never impacts to the sufficient conditions in the first round. This is because m_{12} is used in late steps of the first round in both branches (See Table. 4). Thus, once we obtain an (H, M) satisfying the differential path up to step 17, we can generate up to 2^{32} valid (H, M) by modifying m_{12}. As a summary, the complexity for satisfying the differential path for the first 17 steps is (approximately) 2^{-32} times of the complexity of satisfying the whole differential path, and thus can be ignored.

For the remaining steps (after step 17), we simply satisfy the path in the brute-force manner. Hence, the entire attack complexity only depends on the number of conditions after step 17.

4.5 Complexity Evaluation and Experiments

Besides counting the number of conditions in Δ- and ∇-differential-paths after step 17, we also experimentally verify the amplified probability for outside paths.

Attack on 45 steps. There are 9 and 7 conditions in the Δ- and ∇-differential-paths, respectively. Each condition must be satisfied in two pairs and thus its probability is 2^{-2}. However, 1 condition in the Δ-path is the one discussed in the remarks in Sect. 4.2, which satisfied with probability 2^{-1}. Overall, the complexity is $2^{31(=2(8+7)+1)}$. We then experimentally check the amplified probability, and the final complexity is 2^{27}. This implies that 45 steps of the compression function is non-ideal with respect to the 4-sum property.

Attack on 46 and 47 steps. We experimentally checked the amplified probability, and the final complexities on 46 and 47 steps are 2^{34} and 2^{39} respectively. With respect to the 4-sum property, our attack on 47 steps is faster than the current best generic attack [28].

Table 5. Differential Paths for 2-Dimension Sums on 3-Round RIPEMD-128

$*$ means that the $+/-$sign of difference is not specified. Δb_j^L is 0 for all index j which are not listed below. Similarly Δb_j^R is 0 for all index j which are not listed below.

Path for $\mathrm{CF^L}$		Path for $\mathrm{CF^R}$	
$\Delta a_0^L = 2^8$; $\Delta b_0^L = 0$; $\Delta c_0^L = *2^{31} + 2^{16}$; $\Delta d_0^L = *2^{31} + 2^{16}$;		$\Delta a_0^R = 2^{20}$; $\Delta b_0^R = 0$; $\Delta c_0^R = 0$; $\Delta d_0^R = 2^6$;	
Step j	Δb_j^L	Step j	Δb_j^R
2	$*2^{31}$	1	2^{28}
3	$*2^{31}$	2	2^{15}
22	2^8	5	2^9
43	-2^{21}	6	2^{30}
44	$*2^5$	9	2^{16}
46	$*2^{26}$	13	2^{30}
47	$-2^{28} * 2^{12}$	39	-2^4
48	$*2^{10}$	43	-2^9
		47	-2^{16}

Attack on 48 steps. The complexity to satisfy the path is 2^{48}. We then introduce q-multi-second-order collisions. Generating them by following the path requires $q \cdot 2^{48}$. When $q = 17$, the complexity is less than 2^{53}. On the other hand, from Eq.(1), the generic case requires more than $\sqrt[17]{17!} \cdot 2^{128-\frac{2(512+128)}{17}} \approx 2^{55.55}$. Hence, our attack is faster than the generic case.

Attack on 52 steps. ∇-differential-path in the fourth round of the right branch becomes very complicated because the f function does not have the absorption property. Thus we only verified the amplified probability in the fourth round. As a result, the complexity to obtain a 4-sum is 2^{101}. We then consider the q-multi-second-order collision for $q = 53$. Our attack requires $53 \cdot 2^{101} < 2^{107}$, while the generic case in Eq.(1) requires $2^{108.21}$.

The attack was implemented on single PC. The generated 46-step 2-dimension sum, which is 3-round (48-step) partial 2-dimension sum, is shown in Table. 8.

Table 6. Sufficient Conditions of Attacks on 3-Round RIPEMD-128

Left Branch	Right Branch
$c_{0,16}^L = 0$; $d_{0,16}^L = 0$; $b_{1,16}^L = b_{0,16}^L$;	$b_{0,6}^R = c_{0,6}^R$; $b_{0,28}^R = 1$; $b_{0,15}^R = 0$; $c_{0,28}^R = 0$;
no carry in b_2^L; no carry in b_3^L;	$d_{0,6}^R = 0$; $b_{1,28}^R = 0$, $b_{1,15}^R = 1$; $b_{2,15}^R = 0$
$b_{22,8}^L = 0$; $c_{22,8}^L = d_{22,8}^L$; $b_{24,8}^L = 0$; $b_{25,8}^L = 1$	$b_{3,9}^R = 0$; $b_{4,15}^R = b_{3,15}^R$; $b_{4,30}^R = 0$; $b_{4,9}^R = 1$;
$b_{43,21}^L = 1$; $b_{43,5}^L = 0$; $d_{43,21}^L = 0$; $b_{44,21}^L = 1$;	$b_{5,9}^R = 0$; $b_{5,30}^R = 1$; $b_{6,30} = 0$; $b_{7,9}^R = b_{6,9}^R$;
$b_{45,26}^L = 0$; $b_{45,5}^L = 1$; $b_{46,28}^L = 0$, $b_{46,12}^L = 0$;	$b_{7,16}^R = 0$; $b_{8,30}^R = b_{7,30}^R$; $b_{8,16}^R = 1$; $b_{9,16}^R = 0$;
no carry in b_{46}^L; $b_{47,28}^L = 1$; $b_{47,26}^L = 1$;	$b_{11,16}^R = b_{10,16}^R$; $b_{12,30}^R = 0$; $b_{13,30}^R = 0$;
no carry in b_{47}^L; no carry in b_{48}^L;	$b_{15,30}^R = b_{14,30}^R$; $b_{39,4}^R = 1$; $b_{40,4}^R = 0$;
(H, M) and $(H + \nabla, M + \nabla^M)$:	$c_{39,4}^R = d_{39,4}^R$; $b_{41,4}^R = 1$; $b_{42,9}^R = b_{41,9}^R$; $b_{43,9}^R = 1$;
share same bit values on $b_{44,5}^L$, $b_{46,26}^L$, $b_{47,12}^L$;	$b_{44,9}^R = 0$; $b_{45,9}^R = 0$; $b_{46,16}^R = b_{45,16}^R$; $b_{47,16}^R = 1$;

5 Attacks on RIPEMD-160

Two attacks are presented on RIPEMD-160; in the first scenario, the attack target is starting from the first round and in the second scenario, the attack target is starting from the second round.

In the first scenario, the f functions of both branches do not have the absorption property in the first and third rounds. This makes the efficient differential path construction hard. On the other hand, in the second scenario, the absorption property is available in both branches in the first and third rounds. The differential path is more efficient than the first scenario, and the number of attacked steps is beyond 3 rounds (up to 52 steps).

5.1 Overall Strategy and Relatively Slow Differential Propagation

Different from RIPEMD-128, using the local-collision to construct the path is not efficient in RIPEMD-160. For RIPEMD-128, the local-collision is formed only with differences in 2 message words. However, in RIPEMD-160, we need the difference in 3 message-words due to the direct addition from chaining variable e_j. Hence, we stop using the local-collision. Instead, we insert the difference only into 1 message word that appears in a late step of the second round, and just propagate it to the third round as much as possible. The problem is that the differential propagation in RIPEMD-160 seems much quicker than RIPEMD-128 due to the direct addition from chaining variable e_j, and thus not so many steps can be attacked. However, we explain an useful property of RIPEMD-160 where we can limit the impact of the differential propagation. In fact, this is the main reason why we can attack more than 3 rounds in the second scenario.

Cancelling Differences between e_j and f_{j+1}. Assume that, in some step, there is no difference in chaining variables and a message difference is inserted. If the difference is not propagated through f in the following 3 steps, only chaining variable e has the difference. This situation is illustrated in Fig. 2. Let j be the step index of this chaining variable and $(\Delta a_j, \Delta b_j, \Delta c_j, \Delta d_j, \Delta e_j) = (0, 0, 0, 0, +2^n)$. In step j, e_j is directly added to compute b_{j+1}, thus the difference $+2^n$ is always propagated to b_{j+1}. As shown in Fig. 2, Δe_j and Δb_{j+1} can cancel each other in step $j+1$ through f_{j+1}.

Assume that the difference 2^n in b_{j+1} does not cause the carry, and thus only n-th bit has the difference. In step $j+1$, if the difference in the n-th bit of b_{j+1} is output from f_{j+1}, moreover if its sign is opposite (-2^n), the cancellation occurs.

In the attack starting from the first round, we utilize this property in the third round where $f_x(X, Y, Z)$ is $(X \vee \neg Y) \oplus Z$ in both branches. $Y = 1$ and $Z = 1$ are the conditions for this event. On the other hand, in the attack starting from the second round, $f_x(X, Y, Z)$ in the left branch is $(X \wedge Z) \vee (Y \wedge \neg Z)$. Then, the sign of Δf_{j+1} cannot be opposite of Δe_j, and we need a different strategy. In step j, we make a carry in b_{j+1} as shown in Fig. 3. Therefore, n-th bit position changes in the opposite direction as Δe_j. Finally, in step $j+1$, by propagating the difference in the n-th bit and by absorbing the difference in the $(n+1)$-th bit, the cancellation occurs.

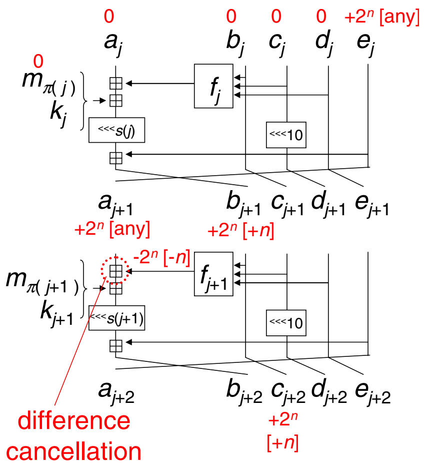

Fig. 2. Difference cancellation between e_j and f_{j+1}. The sign of Δf_{j+1} must be opposite of Δe_j. Information in '[ ]' is the bitwise difference. '[any]' represents that the bitwise difference is irrelevant.

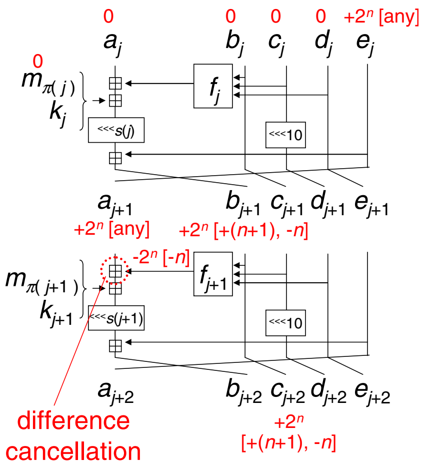

Fig. 3. Difference cancellation with considering the carry effect for the case that the sign of Δf_{j+1} is always the same as Δe_j

Table 7. Differential path construction for the first 3-rounds of RIPEMD-160

round	$\pi^L(j)$	$\pi^R(j)$
1	0 1 ② 3 4 5 6 7 8 9 10 11 12 13 14 15	5 14 7 0 9 ② 11 4 13 6 15 8 1 10 3 12
	$\leftarrow$ Δ | constant	MM $\leftarrow$ ∇ | constant
2	7 4 13 1 10 6 15 3 12 0 9 5 ② 14 11 8	6 11 3 7 0 13 5 10 14 15 8 12 4 9 1 ②
	constant | Δ $\rightarrow$	constant | ∇
3	3 10 14 4 9 15 8 1 ② 7 0 6 13 11 5 12	15 5 1 3 7 14 6 9 11 8 12 ② 10 0 4 13
	$\rightarrow$ Δ	$\rightarrow$ ∇

5.2 Scenario 1: Attack from the First Round

The message differences for the attack from the first round is shown in Table 7. We need to insert both of the Δ-difference and ∇-difference in the 10th bit of m_2. To avoid the contradiction of two paths, the differences and the values of m_2 must be carefully chosen. We choose the following message differences;

$$\Delta m_2 = m_2^2 - m_2^1 = m_2^4 - m_2^3 = +2^{10}, \nabla m_2 = m_2^3 - m_2^1 = m_2^4 - m_2^2 = -2^{10}. \quad (2)$$

To achieve this, we first choose m_2^1 and then compute $m_2^2 \leftarrow m_2^1 + 2^{10}$ and $m_2^3 \leftarrow m_2^1 - 2^{10}$. m_2^4 should be $m_2^1 + 2^{10} - 2^{10}$ and thus identical with m_2^1. Hence, the attack only requires 3 messages m_2^1, m_2^2, m_2^3 rather than the standard message quartet. The differential paths and sufficient conditions are shown in Tables 10 and 11.

For the differential path in Table 10, the first round of $\mathrm{CF^L}$ and $\mathrm{CF^R}$ can be guaranteed with the message modification in negligible time. Hence, the attack cost only depends on the differential path from the second round. As a result, we can generate 4-sums up to 38 steps with 2^{42} computations. Because $2^{42} < 2^{160/3}$, the attack runs faster than the generic 4-sum attack. This can be regarded as partial 2-dimension sums up to 40 steps because the newly computed values in the last 2 steps are not used to compute two output chaining variables $H_{i-1}^{(b)}$ and $H_{i-1}^{(c)}$. The attack was implemented on a PC. The generated 38-step 4-sum (or 40-step partial 2-dimension sum) is shown in Table 9 in Appendix. Theoretically, 2-dimension sums can be generated up to 43 steps with 2^{151} computations.

5.3 Scenario 2: Attack from the Second Round

The overall strategy is the same as the first scenario. The details of the attack such as the message difference, differential path, and sufficient conditions are optimized for this scenario. Different from the first scenario, the f functions in the third round (round 4) have the absorption property. Hence, the differential propagation can be controlled more efficiently and this enables us to attack more rounds. Due to the limited space, we only show the message differences and how to propagate them in Table 12. The complexity to generate 4-sums up to 40 steps is 2^{36} computations, which is faster than the generic 4-sum attack using the generalized birthday attack. A 40-step 2-dimension sum, which can also be regarded as 42-step partial 2-dimension sum, was generated in our experiment. Due to the limited space, the generated data is omitted.

6 Concluding Remarks

We presented distinguishers against compression functions of RIPEMD-128 and -160. Differential paths were constructed by regarding CF^{L} as the first part and CF^{R} as the second part. This enabled us to analyze CF^{L} and CF^{R} independently. On RIPEMD-128, the local collision was applied to construct differential paths. Partial 2-dimension sums were generated for 48 steps. Theoretically, the attack works up to 52 steps. On RIPEMD-160, the difference cancelation between Δe_j and Δf_{j+1} was exploited. Partial 2-dimension sums were generated up to 40 steps. Theoretically, the attack works up to 43 steps. If the attack starts from the second round, more rounds can be attacked. We stress that our results do not impact to the security of full RIPEMD-128 and RIPEMD-160 hash functions.

Acknowledgements. Lei Wang was supported by Grant-in-Aid for JSPS Fellows (23001043).

References

1. 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)

2. 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)
3. U.S. Department of Commerce, National Institute of Standards and Technology: Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices (2007), `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`.
4. 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)
5. International Organization for Standardization: ISO/IEC 10118-3:2004, Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions (2004)
6. Cryptography Research and Evaluation Committees (CRYPTREC): e-Government recommended ciphers list (2003), `http://www.cryptrec.go.jp/english/images/cryptrec_01en.pdf`
7. Freier, A., Karlton, P., Kocher, P.: The Secure Sockets Layer (SSL) Protocol Version 3.0. Internet Engineering Task Force (IETF), RFC 6101 (2001), `http://www.ietf.org/rfc/rfc6101.txt`
8. Project, T.O. (crypto - OpenSSL cryptographic library), `http://www.openssl.org/docs/crypto/ripemd.html`
9. The Legion of the Bouncy Castle (Bouncy Castle Crypto APIs), `http://www.bouncycastle.org/`
10. Technische Universität Darmstadt (FlexiProvider), `http://www.flexiprovider.de/`
11. The GNU Crypto project: (GNU Crypto), `http://www.gnu.org/software/gnu-crypto/`
12. Crypto++: (Crypto++ Library 5.6.1 API Reference), `http://www.cryptopp.com/`
13. Kap, J.: Test Cases for HMAC-RIPEMD160 and HMAC-RIPEMD128. Internet Engineering Task Force (IETF), RFC 2286 (1998), `http://www.ietf.org/rfc/rfc2286.txt`
14. Keromyti, A., Provos, N.: The Use of HMAC-RIPEMD-160-96 within ESP and AH. Internet Engineering Task Force (IETF), RFC 2857 (2001), `http://www.ietf.org/rfc/rfc2857.txt`
15. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
16. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
17. Sasaki, Y., Aoki, K.: Meet-in-the-Middle Preimage Attacks on Double-Branch Hash Functions: Application to RIPEMD and Others. In: Boyd, C., González Nieto, J. (eds.) ACISP 2009. LNCS, vol. 5594, pp. 214–231. Springer, Heidelberg (2009)
18. Chang, D., Hong, S., Kang, C., Kang, J., Kim, J., Lee, C., Lee, J., Lee, J., Lee, S., Lee, Y., Lim, J., Sung, J. (ARIRANG), Available at NIST home page: `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html`
19. Ohtahara, C., Okada, K., Sasaki, Y., Shimoyama, T.: Preimage Attacks on Full-ARIRANG: Analysis of DM-Mode with Middle Feed-Forward. In: Jung, S., Yung, M. (eds.) WISA 2011. LNCS, vol. 7115, pp. 40–54. Springer, Heidelberg (2012)

20. 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)
21. 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)
22. 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)
23. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
24. Aumasson, J.-P., Çalık, Ç., Meier, W., Özen, O., Phan, R.C.-W., Varıcı, K.: Improved Cryptanalysis of Skein. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 542–559. Springer, Heidelberg (2009)
25. Biryukov, A., Nikolić, I., Roy, A.: Boomerang Attacks on BLAKE-32. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 218–237. Springer, Heidelberg (2011)
26. Lamberger, M., Mendel, F.: Higher-order differential attack on reduced SHA-256. Cryptology ePrint Archive, Report 2011/037 (2011), `http://eprint.iacr.org/2011/037`
27. Sasaki, Y.: Boomerang Distinguishers on MD4-Family: First Practical Results on Full 5-Pass HAVAL. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 1–18. Springer, Heidelberg (2012)
28. Wagner, D.: A Generalized Birthday Problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)
29. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-Order Differential Collisions for Reduced SHA-256. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 270–287. Springer, Heidelberg (2011)
30. Mendel, F., Rechberger, C., Schläffer, M., Thomsen, S.S.: The Rebound Attack: Cryptanalysis of Reduced Whirlpool and Grøstl. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 260–276. Springer, Heidelberg (2009)
31. 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)
32. RIPE Integrity Primitives Berlin, Heidelberg, New York: Integrity Primitives for Secure Information Systems, Final RIPE Report of RACE Integrity Primitives Evaluation, RIPE-RACE 1040 (1995)

A Data for Experiments and Attacks on RIPEMD-160

Table 8. 4-sum on 46-steps and partial 2-dimension sum on 48-steps of RIPEMD-128

H_i^1	0x400268ec; 0x159b2e00 0x 6a66026; 0x268c3594;
M_i^1	0x7b69e00f; 0x7a1da89e; 0xb6760e61; 0x222860d3; 0x20d0343c; 0x36333ccb; 0x44a67388; 0x9034d3f9;
	0x19810c83; 0x9cba7f1e; 0x43aa3459; 0xe8987de1; 0xf48cfeb0; 0x36b56404; 0xca71b589; 0x4e4956b3;
46-Step H_{i+1}^1	0x8e492929; 0x34c37860; 0x085981da; 0x3a28780d;
3-Round H_{i+1}^1	0xf57d1452; 0x00cc6f47; 0x9c7fe2e0; 0x5cdae22d;
H_i^2	0x400269ec; 0x159b2e00; 0x86a76026; 0xa68d3594;
M_i^2	0x7b69df0f; 0x7a1da89e; 0xb6760e61; 0x222860d3; 0x20d0343c; 0x36333ccb; 0xc4a67388; 0x9034d3f9;
	0x19810c83; 0x9cba7f1e; 0x43aa3459; 0xe8987de1; 0xf48cfeb0; 0x36b56404; 0xca71b589; 0x4e4956b3;
46-Step H_{i+1}^2	0x4b37a7fb; 0xe0a9ebf0; 0x09e98a18; 0x17b730cd;
3-Round H_{i+1}^2	0x672c3692; 0x5e5c2707; 0xe9e5bbda; 0xc6e4b82a;
H_i^3	0x401268ec; 0x159b2e00; 0x06a66026; 0x268c35d4;
M_i^3	0x7b69e00f; 0x7a1da89e; 0xb6760e61; 0x222860d3; 0x20d0343c; 0x36333ccb; 0x04a67388; 0x9034d3f9;
	0x19810c83; 0x9cba7f1e; 0x43aa3459; 0xe8987de1; 0xf48cfeb0; 0x36b56404; 0xca71b589; 0x4e4956b3;
46-Step H_{i+1}^3	0xd2ae4d5e; 0x5b495822; 0x13ac118a; 0x9c22aa6a;
3-Round H_{i+1}^3	0x5955db12; 0x62b6a1a4; 0xe0a50755; 0xa0eb49c4;
H_i^4	0x401269ec; 0x159b2e00; 0x86a76026; 0xa68d35d4;
M_i^4	0x7b69df0f; 0x7a1da89e; 0xb6760e61; 0x222860d3; 0x20d0343c; 0x36333ccb; 0x84a67388; 0x9034d3f9;
	0x19810c83; 0x9cba7f1e; 0x43aa3459; 0xe8987de1; 0xf48cfeb0; 0x36b56404; 0xca71b589; 0x4e4956b3;
46-Step H_{i+1}^3	0x8f9ccc30; 0x072fcbb2; 0x153c19c8; 0x79b1632a;
3-Round H_{i+1}^3	0xcb04f456; 0xc0465964; 0x2e0ae04f; 0x0afb77c2;
46-Step 4-sum	0x00000000; 0x00000000; 0x00000000; 0x00000000;
3-Round 4-Sum	0xfffff704; 0x00000000; 0x00000000; 0x00065801;

Table 9. 38-step 4-sum and 40-step partial 2D sum on RIPEMD-160 from 1st round

H_i^1	0x4144c3a7; 0x8a965cea; 0x647e4d03; 0x04e7a03c; 0x18814c3e;
M_i^1	0x15f04e2b; 0xb2c328cd; 0x8eea7e12; 0xa1d55a25; 0xbff66c59; 0x399570f1; 0x997d1fd4; 0xea8f403e;
	0xab607923; 0x712bc2a1; 0x32c11766; 0xafaf4bfe; 0x7297f9d4; 0xe2c4573f; 0x96dc27dc; 0x566d9d73;
38 Steps H_{i+1}^1	0x33d3fb92; 0x753e7a17; 0x50fb34b1; 0x1874be98; 0x48de951c;
H_i^2	0x4146bfa7; 0x8a965cea; 0x647e4d03; 0x04e7a43c; 0x08814c3e;
M_i^2	0x15f04e2b; 0xb2c328cd; 0x8eea7a12; 0xa1d55a25; 0xbff66c59; 0x399570f1; 0x997d1fd4; 0xea8f403e;
	0xab607923; 0x712bc2a1; 0x32c11766; 0xafaf4bfe; 0x7297f9d4; 0xe2c4573f; 0x96dc27dc; 0x566d9d73;
38 Steps H_{i+1}^2	0x07e43a48; 0x1482c47c; 0x473df79e; 0xefed372c; 0x55037e85;
H_i^3	0x404cc5a7; 0x8a9e5cea; 0x647e4c03; 0x04e7a23c; 0x18814c3e;
M_i^3	0x15f04e2b; 0xb2c328cd; 0x8eea8212; 0xa1d55a25; 0xbff66c59; 0x399570f1; 0x997d1fd4; 0xea8f403e;
	0xab607923; 0x712bc2a1; 0x32c11766; 0xafaf4bfe; 0x7297f9d4; 0xe2c4573f; 0x96dc27dc; 0x566d9d73;
38 Steps H_{i+1}^3	0xad046562; 0x61c2166a; 0x9d7dc2b3; 0x901e2f34; 0x12d8aa5c;
H_i^4	0x404ec1a7; 0x8a9e5cea; 0x647e4c03; 0x04e7a63c; 0x08814c3e;
M_i^4	0x15f04e2b; 0xb2c328cd; 0x8eea7e12; 0xa1d55a25; 0xbff66c59; 0x399570f1; 0x997d1fd4; 0xea8f403e;
	0xab607923; 0x712bc2a1; 0x32c11766; 0xafaf4bfe; 0x7297f9d4; 0xe2c4573f; 0x96dc27dc; 0x566d9d73;
38 Steps H_{i+1}^3	0x8114a418; 0x010660cf; 0x93c085a0; 0x6796a7c8; 0x1efd93c5;
38 Steps 4-sum	0x00000000; 0x00000000; 0x00000000; 0x00000000; 0x00000000;
40 Steps 4-Sum	noisy data noisy data 0x00000000; 0x00000000; noisy data

Table 10. Differential paths for 2D sums on RIPEMD-160 in the first scenario

Path for CF^{L}		Path for CF^{R}	
$\Delta a_0^L = 2^{17} - 2^{10}$; $\Delta b_0^L = 0$; $\Delta c_0^L = 0$; $\Delta d_0^L = 2^{10}$; $\Delta e_0^L = -2^{28}$;		$\Delta a_0^R = -2^{24} + 2^{19} + 2^9$; $\Delta b_0^R = 2^{19}$; $\Delta c_0^R = -2^8$; $\Delta d_0^R = 2^9$; $\Delta e_0^R = 0$;	
Step j	Δb_j^L	Step j	Δb_j^R
29	-2^{21}	1	-2^0
33	-2^{31}	32	2^{21}
36	-2^{16}	35	-2^{14}
39	2^7	36	$+2^{31}$
40	$-2^{26} - 2^2$	38	2^{30}
41	$-2^{26} + 2^{19}$	39	$-2^{24} - 2^{15}$
42	-2^{25}	40	2^9
43	2^{25}	41	-2^{20}
		42	$2^{15} + 2^8 + 2^6$
		43	$-2^{25} - 2^{24} - 2^2$

Table 11. Sufficient conditions on RIPEMD-160 in the first scenario

Left Branch	Right Branch
$b^L_{0,10} = c^L_{0,10}$; no carry in d^L_0 and e^L_0;	$b^R_{0,9} = 0$; $c^R_{0,19} = 1$; $d^R_{0,8} = 0$;
no carry in b^L_{29}; $b^L_{28,21} = b^L_{27,21}$; $b^L_{30,21} = 0$;	no carry in b^R_0, c^R_0 and d^R_0;
$b^L_{31,31} = 1$; no carry in b^L_{33}; $b^L_{32,31} = 1$;	$b^R_{0,0} = 0$; $b^R_{0,18} = 1$; $c^R_{0,9} = 1$; $c^R_{0,22} = 1$;
$b^L_{34,31} = 1$; $b^L_{35,31} \vee \neg b^L_{34,31} = 1$; $b^L_{35,16} = 0$;	$b^R_{0,22} = 0$; $b^R_{1,29} = 1$; $b^R_{2,10} = 1$;
no carry in b^L_{36} and b^L_{37};	no carry in b^R_{32}; $b^R_{31,21} = 0$; $b^R_{33,21} = 1$;
$b^L_{37,16} = 1$; $b^L_{36,9} = 1$; $b^L_{35,9} = 1$;	$b^R_{34,31} \vee \neg b^R_{33,31} = 1$; no carry in b^R_{35};
$b^L_{38,9} = 1$; $b^L_{38,26} \vee \neg b^L_{37,26} = 1$; $b^L_{38,7} = 0$;	$b^R_{34,14} = 0$; no carry in b^R_{36}; $b^R_{34,14} = 0$;
no carry in b^L_{39}; $b^L_{39,19} \vee \neg b^L_{38,19} = 1$; $b^L_{40,7} = 1$;	no carry in b^R_{36}; $b^R_{36,14} = 1$; $b^R_{35,31} = 0$;
no carry in b^L_{40}; $b^L_{39,26} = 1$; $b^L_{39,2} = 0$;	$b^R_{37,31} = 1$; $b^R_{37,24} \vee \neg b^R_{36,24} = 1$; no carry in b^R_{38};
$b^L_{38,26} = 1$; no carry in b^L_{41}; $b^L_{41,26} = 1$;	$b^R_{37,30} = 0, b^R_{38,9} \vee \neg b^R_{37,9} = 1$; no carry in b^R_{39}; $b^R_{39,30} = 1$;
$b^L_{41,2} = 1$; $b^L_{40,26} = 0$; $b^L_{40,24} = 1$;	$b^R_{38,24} = 1$; $b^R_{38,15} = 0$; $b^R_{37,24} = 1$; $b^R_{40,24} = 1$
$b^L_{39,24} = 1$; $b^L_{41,17} \vee \neg b^L_{40,17} = 1$; no carry in b^L_{42};	no carry in b^R_{40}; $b^R_{40,15} = 1$; $b^R_{39,9} = 1$; $b^R_{38,9} = 1$;
$b^L_{42,24} = 1$; $b^L_{42,19} = 1$; $b^L_{42,4} = 0$;	$b^R_{40,8} \vee \neg b^R_{39,8} = 1$; no carry in b^R_{41}; $b^R_{41,9} = 1$;
$b^L_{41,25} = 0$; $b^L_{41,4} = 1$; $b^L_{42,12} \vee \neg b^L_{41,12} = 1$;	$b^R_{40,20} = 0$; $b^R_{41,25} \vee \neg b^R_{40,25} = 1$; $b^R_{41,2} \vee \neg b^R_{40,2} = 1$;
	no carry in b^R_{42}; $b^R_{42,20} = 1$; $b^R_{41,15} = 0$; $b^R_{41,8} = 1$;
	$b^R_{42,19} \vee \neg b^R_{41,19} = 1$;

Table 12. Differential path construction for the intermediate 3-rounds of RIPEMD-160

round	$\pi^L(j)$	$\pi^R(j)$
2	7 4 13 1 10 6 15 3 $\textcircled{12}$ 0 9 5 2 14 11 8	6 11 3 7 0 $\textcircled{13}$ 5 10 14 15 8 12 4 9 1 2
	MM $\leftarrow \Delta$ | constant	MM $\leftarrow \nabla$ | constant
3	3 10 14 4 9 15 8 1 2 7 0 6 13 11 5 $\textcircled{12}$	15 5 1 3 7 14 6 9 11 8 12 2 10 0 4 $\textcircled{13}$
	constant | Δ	constant | ∇
4	1 9 11 10 0 8 $\textcircled{12}$ 4 13 3 7 15 14 5 6 2	8 6 4 1 3 11 15 0 5 12 2 $\textcircled{13}$ 9 7 10 14
	$\rightarrow$ Δ	$\rightarrow$ ∇