IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2017, No. 2, pp. 177–202. DOI:10.13154/tosc.v2017.i2.177-202

Cryptanalysis of 48-step RIPEMD-160

Gaoli Wang[1], Yanzhao Shen[2] and Fukang Liu[1,3]

[1] Shanghai Key Laboratory of Trustworthy Computing, School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China
glwang@sei.ecnu.edu.cn

[2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China
63726578@qq.com

[3] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
1152049805@qq.com

Abstract. In this paper, we show how to theoretically compute the step differential probability of RIPEMD-160 under the condition that only one internal variable contains difference and the difference is a power of 2. Inspired by the way of computing the differential probability, we can do message modification such that a step differential hold with probability 1. Moreover, we propose a semi-free-start collision attack on 48-step RIPEMD-160, which improves the best semi-free start collision by 6 rounds. This is mainly due to that some bits of the chaining variable in the *i*-th step can be computed by adding some conditions in advance, even though some chaining variables before step *i* are unknown. Therefore, the uncontrolled probability of the differential path is increased and the number of the needed starting points is decreased. Then a semi-free-start collision attack on 48-step RIPEMD-160 can be obtained based on the differential path constructed by Mendel *et al.* at ASIACRYPT 2013. The experiments confirm our reasoning and complexity analysis.

Keywords: Hash functions · RIPEMD-160 · Semi-free-start collision · Generalized message modification

Introduction

Cryptographic hash functions play an important role in modern cryptography, which can be used in digital signature schemes, message authentication codes, password authentication schemes and so on. A cryptographic hash function H should fulfill some security requirements, among which collision resistance is one of the key security properties. Collision resistance means it is impossible to find two distinct messages M and M' such that $H(IV, M) = H(IV, M')$ in less than $2^{\frac{n}{2}}$ hash computations, where n is the size of the hash value and IV is the standard initial value of H. Semi-free-start collision resistance means it is impossible to find (IV', M) and (IV', M') such that $H(IV', M) = H(IV', M')$ in less than $2^{\frac{n}{2}}$ hash computations, where IV' is not always equal to IV. In Merkle-Damgård [Dam89, Mer89] construction, if the compression function is collision resistant, we can get that the corresponding hash function is also collision resistant. Thus, the semi-free-start collision attack on Merkle-Damgård hash functions is of great significance.

Because of the break-through progresses in MD-SHA hash function cryptanalysis [WLF+05, WY05, WYY05b, WYY05a, YWYP06, YWZW05], especially the analysis of MD5 [WY05] and SHA-1 [WYY05a], NIST started a four-year hash function competition to design a new hash standard SHA-3 [SHA]. Many techniques such as advanced message modification [WLF+05] and tunnels [Kli06] are proposed in the past decade. Recently, the free-start collision attack for the full SHA-1 [SKP16] speeds up the removal of SHA-1. The SHA-2 and SHA-3 hash functions are recommended to be deployed sooner in many products and services that used to

Received: 2017-03-01, Accepted: 2017-05-01, Published: 2017-06-19

rely on SHA-1. Among the MD-SHA family, only SHA-2 [DEM14, MNS11, MNS13] and RIPEMD-160 [MNSS12, MPS+13] compression functions are still unbroken.

RIPEMD-family is a subfamily of the MD-SHA-family. The first hash function of the RIPEMD-family is RIPEMD (denoted by RIPEMD-0) [BP95] which is devised in the framework of the EU project RIPE (RACE Integrity Primitives Evaluation, 1988-1992). The compression function of RIPEMD-0 consists of two parallel MD4-like functions with 48 steps. The early cryptanalysis of RIPEMD-0 was from Dobbertin [Dob97] and the practical collision attack was proposed by Wang *et al.* [WLF+05]. Due to the experience of evaluating MD4 and RIPEMD-0, RIPEMD-128 and RIPEMD-160 [DBP96] were proposed in 1996 by Dobbertin *et al.* as reinforced hash functions for RIPEMD-0. RIPEMD-128 and RIPEMD-160 were standardized by ISO/IEC [Int11], which have 128/160-bit output and 64/80 steps respectively. A series of collision attacks on reduced RIPEMD-128 are presented in [MPRR06, MNS12, WW08, Wan14, WY15] and a semi-free-start collision attack on the full RIPEMD-128 is proposed in [LP13] (the extended version is [LP16]). However, RIPEMD-160 is unbroken until now. Therefore, the continuous analysis of the security margin of RIPEMD-160 is of great significance.

Related Work

At ISC 2006, Mendel *et al.* [MPRR06] examined to what extent the attacks [Dob97, WLF+05] can apply to RIPEMD-128 and RIPEMD-160. At ISC 2012, Mendel *et al.* [MNSS12] presented a practical semi-free-start collision attack on 36-step RIPEMD-160 (not starting from the first step). Later at ASIACRYPT 2013, Mendel *et al.* [MPS+13] proposed semi-free-start collision attacks on 42-step RIPEMD-160 (not starting from the first step) and the first 36-step RIPEMD-160. As for the preimage attack, at INSCRYPT 2010, Ohtahara *et al.* [OSS10] gave a preimage attacks on the first 30-step and the last 31-step RIPEMD-160. At ISC 2014, Wang and Shen [WS14] presented a preimage attack on the first 34-step RIPEMD-160. Finally, at ACNS 2012, Sasaki and Wang [SW12] presented distinguishing attacks on up to 51-step RIPEMD-160 (not starting from the first step). The above are all the previous results that we are aware of on the analysis of RIPEMD-160.

In the semi-free-start collision attack on 42-step RIPEMD-160 [MPS+13], the authors firstly construct relatively sparse differential paths by choosing a proper message difference. The non-linear differential paths are constructed using very efficient automated search techniques [MNS11, MNS13]. The automated search algorithms are very effective to obtain the non-linear differential paths, and other non-linear differential path automated search algorithms include [DCR06] etc. After the differential paths in each branch of RIPEMD-160 are constructed, the authors leverage the method for using the freedom degrees proposed by Landelle and Peyrin [LP13] to get a semi-free-start collision. The method [LP13] uses some message words to ensure the nonlinear parts located in the middle of both branches hold, and use the remaining message words to merge both branches. Mendel *et al.* [MPS+13] point out that it is hard to calculate the differential probability for each step of a given differential path of RIPEMD-160, not as the case for RIPEMD-128. This is due to the step function in RIPEMD-160 is no longer a T-function (a function for which the i-th output bit depends only on the i first lower bits of all input words). The authors [MPS+13] left the problem of theoretically calculating the step differential probability of RIPEMD-160 as an open problem.

Our Contributions

As a first contribution, we propose a method to theoretically calculate the real step differential probability when only one internal variable has difference and the difference is a power of 2. That means we partially answer the open problem raised in [MPS+13]. Furthermore, from the way of computing the differential probability, we can implement the message modification such that a step differential of RIPEMD-160 hold with probability 1. As a second and main contribution,

Table 1: Summary of the Attacks on RIPEMD-160

Attack Type	Steps	Generic	Complexity	Reference
semi-free-start collision	36*	2^{80}	practical	[MNSS12]
semi-free-start collision	42*	2^{80}	$2^{75.5}$	[MPS+13]
semi-free-start collision	36	2^{80}	$2^{70.4}$	[MPS+13]
semi-free-start collision	48*	2^{80}	$2^{76.4}$	Section 3.2
pseudo-preimage	30	2^{160}	2^{148}	[OSS10]
preimage	30	2^{160}	2^{155}	[OSS10]
pseudo-preimage	31*	2^{160}	2^{148}	[OSS10]
preimage	31*	2^{160}	2^{155}	[OSS10]
pseudo-preimage	34	2^{160}	$2^{155.81}$	[WS14]
preimage	34	2^{160}	$2^{158.91}$	[WS14]
distinguishing	48*		practical	[MNSS12]
distinguishing	51	2^{160}	2^{158}	[SW12]

* The attack starts from an intermediate step.

we improve the semi-free start collision attack on reduced RIPEMD-160 from 42 steps to 48 steps by using the 48-step differential path in [MPS+13]. The improvement is mainly due to the generalized message modification technique, using which we can calculate some bits of the chaining variables X_{37} and X_{38} (Y_{30} and Y_{32}) even though X_{36} (Y_{29} and Y_{31}) is unknown by adding some conditions in advance. This method was used by Landelle, Peyrin [LP13] and Mendel *et al.* [MPS+13] to improve the uncontrolled probability respectively. We extend it by incorporating more sophisticated techniques. Furthermore, some conditions of X_{37}, X_{38}, Y_{30} and Y_{32} can be satisfied using the message modification technique [WLF+05, WY05], which is very powerful to improve the probability of the differential path. Therefore, a semi-free start collision attack on 48-step RIPEMD-160 is obtained because the probability of the differential path is improved after the generalized message modification. The previous results and our results on RIPEMD-160 are summarized in Table 1.

Organization of the Paper

The rest of the paper is organized as follows: In Section 1, we describe the notations and the RIPEMD-160 algorithm. Section 2 proposes a method to theoretically calculate the differential probability of a given differential path of RIPEMD-160, and the message modification technique for RIPEMD-160. Section 3 shows the detailed description of the semi-free start collision attack on 48-step RIPEMD-160. Finally, we summarize the paper in Section 4.

1 Preliminaries

1.1 Notations

In order to describe our attack conveniently, we recall some notations [WLF+05] as follows.

1. $M = (m_0, m_1, ..., m_{15})$ and $M' = (m'_0, m'_1, ..., m'_{15})$ represent two 512-bit messages.
2. X_i denotes the output of the i-th step for compressing M in left branch, where $1 \leq i \leq 80$.
3. Y_i denotes the output of the i-th step for compressing M in right branch, where $1 \leq i \leq 80$.
4. X'_i denotes the output of the i-th step for compressing M' in left branch, where $1 \leq i \leq 80$.
5. Y'_i denotes the output of the i-th step for compressing M' in right branch, where $1 \leq i \leq 80$.

6. + denotes addition modulo 2^{32}.
7. $\lll s$ represents the circular shift s bit positions to the left.
8. $\Delta^+ x_i = x'_i - x_i$ denotes the modular difference of two words x_i and x'_i.
9. Δx_i denotes a bitwise signed difference of two words x_i and x'_i. It is noted that Δx_i expresses not only modular difference but also XOR difference.
10. $x_{i,j}$ represents the j-th bit of x_i, starting the counting from 0.
11. $\Delta x_i = [j]$ denotes $x_{i,j} = 0$, $x'_{i,j} = 1$ and $x_{i,k} = x'_{i,k}$ $(0 \leq k \leq 31, k \neq j)$.
12. $\Delta x_i = [-j]$ denotes $x_{i,j} = 1$, $x'_{i,j} = 0$ and $x_{i,k} = x'_{i,k}$ $(0 \leq k \leq 31, k \neq j)$.
13. $\Delta x_i = [\pm j_1, \pm j_2, ..., \pm j_l]$ denotes the j_1-th, j_2-th, ..., j_l-th bits of x_i and x'_i are different. The "+" sign means $x_{i,j} = 0$ and $x'_{i,j} = 1$, and the "-" sign means $x_{i,j} = 1$ and $x'_{i,j} = 0$.

1.2 Description of RIPEMD-160

The hash function RIPEMD-160 compresses any arbitrary message less than 2^{64} length into a hash value with length of 160 bits. The input message is padded, and then processed in 512-bit blocks in the Merkle-Damgård iterative structure. For each 512-bit message block, RIPEMD-160 compresses it into a 160-bit hash value by the compression function, which is composed of two parallel operations: left branch and right branch. Each branch contains 5 rounds, and each round contains 16 steps. The boolean functions in each round are as follows:

$$\begin{aligned}
F_1(X, Y, Z) &= X \oplus Y \oplus Z, \\
F_2(X, Y, Z) &= (X \wedge Y) \vee (\neg X \wedge Z), \\
F_3(X, Y, Z) &= (X \vee \neg Y) \oplus Z, \\
F_4(X, Y, Z) &= (X \wedge Z) \vee (Y \wedge \neg Z), \\
F_5(X, Y, Z) &= X \oplus (Y \vee \neg Z).
\end{aligned}$$

Here X, Y, Z are 32-bit words. The five boolean functions are all bitwise operations. $\neg$ represents the bitwise complement of X. $\wedge$, $\oplus$ and $\vee$ are bitwise AND, XOR and OR respectively.

The Message Expansion. The 512-bit input message block M is divided into 16 words m_i of 32 bits each. At step i, the expanded message word which will be used to update left branch and right branch are denoted by $m_{\pi^l(i)}$ and $m_{\pi^r(i)}$, where the permutations $\pi^l(i)$ and $\pi^r(i)$ can be seen in Table 2.

Initialization. The input chaining variable of the compression function is denoted by $cv = (cv_0, cv_1, cv_2, cv_3, cv_4)$, and the standard initial value can be found in [DBP96]. The initialization process of both branches are as follows:

$$X_{-4} = Y_{-4} = cv_0 \ggg 10, \qquad X_{-3} = Y_{-3} = cv_4 \ggg 10,$$

$$X_{-2} = Y_{-2} = cv_3 \ggg 10, \qquad X_{-1} = Y_{-1} = cv_2, \qquad X_0 = Y_0 = cv_1.$$

State Update Transformation. The state update transformation starts from a 160-bit input chaining variable $cv = (cv_0, cv_1, cv_2, cv_3, cv_4)$ and updates them in 80 steps (5 rounds, each round contains 16 steps). As depicted in Figure 1, in round j $(1 \leq j \leq 5)$, X_i and Y_i $(1 \leq i \leq 80)$ are updated as follows:

$$X_i = (X_{i-4} \lll 10) + \Big((X_{i-5} \lll 10) + F_j(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m_{\pi^l(i)} + k^l_j\Big) \lll s^l_i,$$

Table 2: Order of the Message Words and Rotation Values in RIPEMD-160

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi^l(i)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi^r(i)$	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
s^l_i	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
s^r_i	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$\pi^l(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
$\pi^r(i)$	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
s^l_i	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
s^r_i	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
i	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
$\pi^l(i)$	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
$\pi^r(i)$	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
s^l_i	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
s^r_i	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
i	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
$\pi^l(i)$	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
$\pi^r(i)$	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
s^l_i	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
s^r_i	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8
i	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
$\pi^l(i)$	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13
$\pi^r(i)$	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11
s^l_i	9	15	5	11	6	8	13	12	5	12	13	14	11	8	5	6
s^r_i	8	5	12	9	12	5	14	6	8	13	6	5	15	13	11	11

$$Y_i = (Y_{i-4} \lll 10) + \Big((Y_{i-5} \lll 10) + F_{6-j}\,(Y_{i-1}, Y_{i-2}, (Y_{i-3} \lll 10)) + m_{\pi^r(i)} + k^r_j\Big) \lll s^r_i,$$

where the boolean function F_j, the constants k^l_j and k^r_j depend on round j ($j = \lfloor \frac{i+16}{16} \rfloor$) and left/right branch. The order of message words $\pi^l(i)$, $\pi^r(i)$ and the details of the shift positions s^l_i, s^r_i can be seen in Table 2. For the details of RIPEMD-160, we refer to [DBP96].

The Finalization. The output of compressing the block M is obtained by combining the initial value cv with the outputs of both branch operations. The five 32-bit words cv'_i composing the output chaining variable are calculated by:

$cv'_0 = cv_1 + X_{79} + (Y_{78} \lll 10)$, $\quad cv'_1 = cv_2 + (X_{78} \lll 10) + (Y_{77} \lll 10)$,
$cv'_2 = cv_3 + (X_{77} \lll 10) + (Y_{76} \lll 10)$, $\quad cv'_3 = cv_4 + (X_{76} \lll 10) + Y_{80}$,
$cv'_4 = cv_0 + X_{80} + Y_{79}$.

2 Some Properties and the Message Modification

For RIPEMD-160, Mendel *et al.* [MPS+13] pointed out that it is difficult to theoretically calculate the differential probability for each step of a given differential path and left it as an open problem. Daum [Dau05] proposed a method to calculate the probability that X satisfies the equation $(X + C_0) \lll S = (X \lll S) + C_1$ (i.e., $Pr[X \mid (X + C_0) \lll S = (X \lll S) + C_1]$), where C_0 and C_1 are random constants. We find that from the calculation of such a probability, the probability of the modular difference of the differential probability can be calculated. Moreover, from Theorem 2 in [FLN], we can also compute the differential probability for each step under some particular

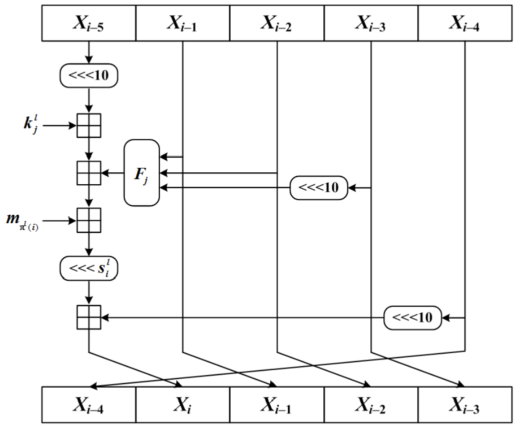

Figure 1: The left branch of step update transformation of RIPEMD-160

circumstances. In this section, we present some properties to compute the differential probability accurately under the circumstance that only one internal state word contains difference and the difference is a power of 2. Moreover, the way of computing the differential probability sheds light on the message modification, so we propose a method to implement the message modification such that the sufficient conditions of a step differential hold with probability 1 after message modification.

2.1 Calculating the Differential Probability

In order to express the following propositions easily, we simplify the i-th step function

$$X_i = (X_{i-4} \lll 10) + \Big((X_{i-5} \lll 10) + F_j\,(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m_{\pi^l(i)} + k_j^l\Big) \lll s_i^l$$

as $X_i = (X_{i-4} \lll 10) + \Big(\,(X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)\,) + m + k\Big) \lll s$, where m, k and s depend on the step i. If ΔX_{i-1}, ΔX_{i-2}, ΔX_{i-3}, ΔX_{i-5} and Δm are all zero, for any given value α, the probability of $\Delta X_i = \alpha$ can be calculated easily as in RIPEMD-128. However, if ΔX_{i-5}, $\Delta f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10))$ or Δm is not zero, then the probability of $\Delta X_i = \alpha$ can not be computed as the case for RIPEMD-128. When there is only one internal state word containing difference among X_{i-1}, X_{i-2}, X_{i-3}, X_{i-5} and m, and the difference is a power of 2, the following propositions can be used to compute the step differential probability.

Proposition 1. *Let $y = x_1 + (x_2 + x_3) \lll s$, $y' = x_1 + (x_2 + x_3') \lll s$, here x_1, x_2, x_3 and x_3' are 32-bit words. If $\Delta x_3 = [i]$ (i.e., $x_{3,i} = 0$, $x'_{3,i} = 1$, $x_{3,k} = x'_{3,k}$ $(0 \le k \le 31, k \ne i)$), then the probability that $\Delta^+ y$ equals to $2^{i+s(mod32)}$ can be computed as follows:*

$$Pr[\Delta^+ y = 2^{i+s(mod32)}] = \begin{cases} 1 - \dfrac{1}{2^{32-s-i}}, & i+s \le 31, \\ \;1 - \dfrac{1}{2^{32-i}}, & i+s > 31. \end{cases}$$

Proof. Let $r_1 = x_2 + x_3$, $r_2 = r_1 \lll s$, $r_1' = x_2 + x_3'$, $r_2' = r_1' \lll s$, then it is obvious that

$$\Delta^+ y = \Delta^+ r_2.$$

Therefore,

$$Pr[\Delta^+ y = 2^{i+s(mod32)}] = Pr[\Delta^+ r_2 = 2^{i+s(mod32)}].$$

Case 1: When $i + s \leq 31$, then

$$\begin{aligned} \Delta^+ r_2 = 2^{i+s} \quad \Longleftrightarrow \quad & \Delta r_1 = [i] \\ or \quad & \Delta r_1 = [-i, i+1] \\ or \quad & \Delta r_1 = [-i, -(i+1), i+2] \\ & ... \\ or \quad & \Delta r_1 = [-i, -(i+1), ..., -(30-s), 31-s]. \end{aligned} \tag{1}$$

From $\Delta x_3 = [i]$ we can get $\Delta^+ r_1 = 2^i$. Thus $\Delta r_1 = [i]$ (i.e. $\Delta^+ r_1 = 2^i$ and $r_{1,i} = 0$) holds with probability $\frac{1}{2}$. In a similarly way, $\Delta r_1 = [-i, i+1]$ (i.e. $\Delta^+ r_1 = 2^i$, $r_{1,i} = 1$ and $r_{1,i+1} = 0$) holds with probability $\frac{1}{2^2}$. Therefore, the probability of event (1) is $\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{32-s-i}} = 1 - (\frac{1}{2})^{32-s-i}$.
Case 2: When $i + s > 31$, then

$$\begin{aligned} \Delta^+ r_2 = 2^{i+s-32} \quad \Longleftrightarrow \quad & \Delta r_1 = [i] \\ or \quad & \Delta r_1 = [-i, i+1] \\ or \quad & \Delta r_1 = [-i, -(i+1), i+2] \\ & ... \\ or \quad & \Delta r_1 = [-i, -(i+1), ..., -30, 31]. \end{aligned} \tag{2}$$

It is easy to see that the probability of event (2) is $\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{32-i}} = 1 - (\frac{1}{2})^{32-i}$. □

Corollary 1 *Let* $y = x_1 + (x_2 + x_3) \lll s$, $y' = x_1 + (x_2 + x'_3) \lll s$, *here* x_1, x_2, x_3 *and* x'_3 *are 32-bit words. If* $\Delta x_3 = [-i]$ $(i.e., x_{3,i} = 1, x'_{3,i} = 0, x_{3,k} = x'_{3,k}\ (0 \leq k \leq 31, k \neq i))$, *then we can get*

$$Pr[\Delta^+ y = -2^{i+s(mod 32)}] = \begin{cases} 1 - \dfrac{1}{2^{32-s-i}}, & i + s \leq 31, \\ 1 - \dfrac{1}{2^{32-i}}, & i + s > 31. \end{cases}$$

2.2 Message Modification

In order to find a collision of a Merkle-Damgård hash function (esp. RIPEMD-160), a high probability (after the accelerating process) differential path must be constructed on the basis of a proper differences of the message and/or the initial value. Then a set of sufficient conditions that ensure the differential path hold is deduced. In order to find a pair of messages which satisfy the sufficient conditions, many accelerating techniques are proposed. Message modification [WY05, WYY05a] is one of the powerful techniques to accelerate the process of finding a pair of messages following the differential path.

For the sufficient conditions of each step of a given differential path of RIPEMD-128, it is easy to use the message modification techniques to fulfill the conditions. However, it is not the case for RIPEMD-160 due to the step function of RIPEMD-160 is not a T-function (i.e., the i-th output bit depends only on the i first lower bits of all input words). The derivation process of the sufficient conditions of the differential path in RIPEMD-160 is different from that in RIPEMD-128, which will be illustrated through the following example. The differential characteristic in step 37 of the left branch is

$$(\Delta X_{32} = [-10, -16], \Delta X_{33} = [-21, -24, 30], \Delta X_{34} = [-7], \Delta X_{35} = [21, 24], \Delta X_{36} = [-7])$$
$$\longrightarrow \Delta X_{37} = [21],$$

and

$$X_{37} = (X_{33} \lll 10) + \Big((X_{32} \lll 10) + F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9 + k^l_3\Big) \lll 14.$$

According to the structure of the nonlinear function F_3, the conditions $X_{35,7} = 1$, $X_{34,29} = 1$, $X_{36,21} = 0$, $X_{34,11} = 1$ and $X_{36,24} = 1$ ensure that $(\Delta X_{36} = [-7], \Delta X_{35} = [21, 24], \Delta X_{34} = [-7])$ result in $\Delta F_3(X_{36}, X_{35}, (X_{34} \lll 10)) = [7, 21, \pm 17]$. Denote

$$T_{37} = (X_{32} \lll 10) + F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9 + k_3^l, \qquad P_{37} = T_{37} \lll 14,$$

obviously, we anticipate that the differences [±17] in ΔF_3 cancels out the difference [-21] in ΔX_{33}. However, if the differences [±17] in ΔF_3 spreads to the 18-th bit, then the corresponding difference of P_{37} not only locates in bit 31, but also in bit 0. Thus, $\Delta^+ X_{37} = 2^{21}$ can not be hold. Therefore, the set of conditions $X_{35,7} = 1$, $X_{34,29} = 1$, $X_{36,21} = 0$, $X_{34,11} = 1$, $X_{36,24} = 1$ and $X_{37,21} = 0$ is not a set of sufficient conditions of the 37-th step differential path. Thus, in the message modification, we must make sure $\Delta^+ X_{37} = 2^{21}$ and $X_{37,21} = 0$ hold simultaneously. So the message modification for RIPEMD-160 is very complex. We present two examples (the second example is illustrated in Appendix B) to illustrate part of the process of the message modification.

Example 1. For the i-th step function $X_i = (X_{i-4} \lll 10) + \Big((X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m + k\Big) \lll s$, where m, k and s depend on the step i. Denote

$$r_1 = (X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m + k,$$

$$r_1' = (X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m' + k,$$

$$r_2 = r_1 \lll s, \qquad r_2' = r_1' \lll s.$$

Let $\Delta^+ m = 2^{30}$ and $s = 3$, in order to ensure $\Delta X_i = [1]$, the message modification can be processed as follows.

Step 1. If $\Delta^+ X_i \neq 2$, from the proof of Proposition 1, we know that $\Delta r_1 = [-30, -31]$, i.e. $r_{1,30} = r_{1,31} = 1$ and $r'_{1,30} = r'_{1,31} = 0$. The message word m can be modified in the following three different ways:
(1)

$$m \longleftarrow m \pm 2^{31},$$

then after this modification, the most two significant bits of r_1 and r_1' are $r_{1,30} = 1$, $r_{1,31} = 0$, $r'_{1,30} = 0$ and $r'_{1,31} = 1$, which means $\Delta r_1 = [-30, 31]$, thus $\Delta r_2 = [-1, 2]$. Therefore, $\Delta^+ X_i = -2 + 2^2 = 2$.
(2)

$$m \longleftarrow m - 2^{30},$$

then after this modification, the most two significant bits of r_1 and r_1' are $r_{1,30} = 0$, $r_{1,31} = 1$, $r'_{1,30} = 1$ and $r'_{1,31} = 1$, which means $\Delta r_1 = [30]$, thus $\Delta r_2 = [1]$. Therefore, $\Delta^+ X_i = 2$.
(3)

$$m \longleftarrow m + 2^{30},$$

then after this modification, the most two significant bits of r_1 and r_1' are $r_{1,30} = 0$, $r_{1,31} = 0$, $r'_{1,30} = 1$ and $r'_{1,31} = 0$, which means $\Delta r_1 = [30]$, thus $\Delta r_2 = [1]$. Therefore, $\Delta^+ X_i = 2$.

Step 2. If $X_{i,1} \neq 0$, we modify the message word m according to the following three circumstances.
(1) If $r_{1,30} = 0$ and $r_{1,31} = 0$ (i.e. $r'_{1,30} = 1$ and $r'_{1,31} = 0$), m can be modified as: $m \longleftarrow m + 2^{30}$. After the modification, the most two significant bits of r_1 and r_1' are $r_{1,30} = 1$, $r_{1,31} = 0$, $r'_{1,30} = 0$ and $r'_{1,31} = 1$. On the one hand, $r_{1,30}$ is flipped, so $r_{2,1}$ is flipped, thus $X_{i,1}$ is flipped. On the other, after the modification, there is $\Delta^+ r_2 = -2 + 2^2 = 2$. Therefore, $\Delta^+ X_i = 2$ still holds.

(2) If $r_{1,30} = 0$ and $r_{1,31} = 1$, (i.e. $r'_{1,30} = 1$ and $r'_{1,31} = 1$), m can be modified as: $m \longleftarrow m - 2^{30}$. After the modification, the most two significant bits of r_1 and r_1' are $r_{1,30} = 1$, $r_{1,31} = 0$, $r'_{1,30} = 0$

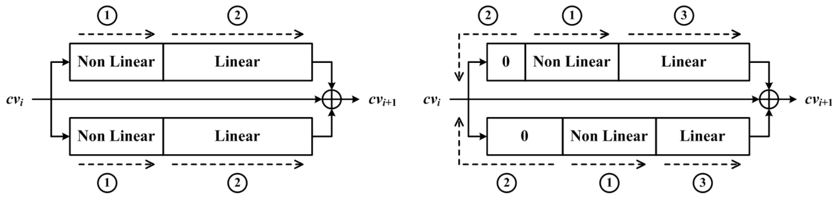

Figure 2: The traditional (left-hand side) and new (right-hand side) approach for collision search on double-branch compression functions [LP13]

and $r'_{1,31} = 1$. Therefore, $X_{i,1}$ is flipped and $\Delta^+ X_i = 2$ still holds.

(3) If $r_{1,30} = 1$ and $r_{1,31} = 0$, (i.e. $r'_{1,30} = 0$ and $r'_{1,31} = 1$), m can be modified as: $m \longleftarrow m \pm 2^{30}$. After the modification, $X_{i,1}$ is flipped and $\Delta^+ X_i = 2$ still holds.

3 Improved Semi-free-start Collision Attack on Reduced RIPEMD-160

The key point of collision attacks on RIPEMD family hash functions is searching a low probability non-linear part and a high probability linear part. In the traditional collision attacks [Dob97, MNS12, MNSS12, Wan14, WLF+05], the non-linear parts are located in the first steps and linear parts in the remaining ones. This restricts the space of the possible differential paths and usually fail to implement a collision attack. In this circumstance, Landelle and Peyrin [LP13] propose a new method to utilize both the freedom of the message words and the freedom of the initial value, which gives a semi-free-start collision attack on the full RIPEMD-128. The non-linear part is fixed in the intermediate steps. Some message words are used to ensure the non-linear part hold and the remaining ones are used to merge both branches. Figure 2 shows the differences between the previous strategy and the strategy in [LP13].

Mendel *et al.* [MPS+13] constructs a differential path for 48-step RIPEMD-160, which is located between steps 17-64. Using the differential path, they obtain a semi-free-start collision attack on 42-step RIPEMD-160 with complexity $2^{75.5}$. Meanwhile, they measure that the probability of the differential path from step 58 to 64 is about $2^{-11.3}$. So the complexity of 48-step attack is about $2^{86.8}$. Obviously, if the differential probability between steps 17-58 is improved, the collision attack on 48-step RIPEMD-160 may be obtained.

In this section, we provide a method to compute the values of $X_{37,i}$ ($i = 2, 7, 17, 21$), $X_{38,i}$ ($i = 17, 21$), $Y_{30,i}$ ($i = 9, 15, 21, 27, 30, 31$) and $Y_{32,20}$ by adding some conditions on the chaining variables in advance, even though X_{36}, Y_{29} and Y_{31} are unknown. Thus, the complexity of the collision attack on 42-step RIPEMD-160 is decreased and a semi-free-start collision attack on 48-step RIPEMD-160 is constructed.

3.1 Review the General Strategy of the Attack on 42-step RIPEMD-160

At a high level, the process of getting a semi-free-start collision attack on 42-step RIPEMD-160 [MPS+13] consists of the following three phases:

Phase 1: A 48-step differential path (between steps 17-64) is constructed by inserting differences in the message word m_7.

Phase 2: The message words m_i ($0 \le i \le 15, i \ne 1, 4, 7, 13$) are modified such that the conditions between steps 17-35 and $X_{37,i}$ ($i = 2, 7$) in the left branch and the conditions between steps 17-28 in the right branch are fulfilled. The candidate (the internal state variables and the corresponding message words) at the end of this phase is called a starting point.

Phase 3: Using the remaining free message words m_1, m_4, m_7 and m_{13} to merge the initial states of both branches to the same value. If the rest conditions of the differential path of 42-step RIPEMD-160 are fulfilled, a semi-free-start collision is obtained.

In phase 2 of searching a starting point, firstly, the internal states words $(X_{26}, X_{27}, X_{28}, X_{29}, X_{30})$ in the left branch and $(Y_{21}, Y_{22}, Y_{23}, Y_{24}, Y_{25})$ in the right branch are fixed. Then the message words are modified in the following order m_{11}, m_{15}, m_8, m_3, m_{12}, m_{14}, m_{10}, m_2, m_5, m_9, m_0 and m_6. That means the conditions in the internal states in both branches are satisfied in the following order: X_{31}, Y_{26}, (X_{32}, Y_{27}), X_{33}, Y_{28}, (X_{25}, Y_{20}), (X_{34}, X_{35}, Y_{19}), X_{24}, (X_{23}, Y_{18}), $(X_{22}, X_{37,2}, X_{37,7})$, $(X_{21}, X_{20}, X_{19}, X_{18})$ and X_{17}. After finding a starting point, the remaining free message words m_1, m_4, m_7 and m_{13} are used to make sure that the five initial words of both branches have a match, i.e., $X_i = Y_i$ $(i = 12, 13, 14, 15, 16)$, which holds with a probability of 2^{-32}.

After Phase 2 and Phase 3, the sufficient conditions in steps 36-58 of the left branch (except $X_{37,2} = 0$ and $X_{37,7} = 1$) and the sufficient conditions in steps 29-58 of the right branch are not modified. [MPS+13] measures the probability of both branches by experiment. The probability of the differential path in steps 36-58 of the left branch (except $X_{37,2} = 0$ and $X_{37,7} = 1$) is $2^{-8.8}$. The probability of the differential path in steps 29-58 of the right branch is $2^{-36.6}$. Moreover, the probability of merging the initial states of both branches to the same value in Phase 3 is 2^{-32}. Therefore, the uncontrolled probability of the differential path of 42-step RIPEMD-160 is $2^{-32} \times 2^{-8.8} \times 2^{-36.6} = 2^{-77.4}$. Thus, in Phase 2, it needs to generate $2^{77.4}$ starting points, which requires 2^{73} 42-step RIPEMD-160. Moreover, the merging costs about $2^{-1.9}$ calls of the 42-step compression function. Therefore, the complexity of the semi-free start collision attack on 42-step RIPEMD-160 is $2^{73} + 2^{77.4-1.9} \approx 2^{75.5}$.

3.2 Semi-free-start Collision Attack on 48-step RIPEMD-160

Mendel *et al.* [MPS+13] proposes a differential path for 48-step RIPEMD-160 (in Table 7) and gets a collision attack on 42-step RIPEMD-160. The conditions on X_i $(i = 36, 37, 38, 57, ..., 64)$ and Y_i $(i = 29, ..., 33, 60, ..., 64)$ of the differential path [MPS+13] are presented in Table 8 and Table 9. In this section, by using the generalized message modification technique, we can obtain a semi-free-start collision attack on 48-step RIPEMD-160 by decreasing the number of the starting points needed in Phase 3.

In Phase 2 of searching a starting point, firstly, similar to [MPS+13], we fix the internal state variables $(X_{26}, X_{27}, X_{28}, X_{29}, X_{30})$ in the left branch and $(Y_{21}, Y_{22}, Y_{23}, Y_{24}, Y_{25})$ in the right branch. Then the message modification is used to make sure the conditions on the chaining variables X_i $(i = 17, ..., 25, 31, ..., 35)$ and Y_i $(i = 18, 19, 20, 26, 27, 28)$ hold. Moreover, the message words m_1, m_4, m_7 and m_{13} are used to merge the initial values of both branches, i.e., $X_i = Y_i$ $(i = 12, 13, 14, 15, 16)$ in Phase 3, thus X_{36}, Y_{29} and Y_{31} (depend on m_4, m_4 and m_1 respectively) are unknown.

However, by adding some other conditions on the chaining variables in both branches in advance, we can calculate the values of $X_{37,i}$ $(i = 2, 7, 17, 21)$, $X_{38,i}$ $(i = 17, 21)$ of the left branch and $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$, $Y_{32,20}$ of the right branch correctly. The computation of the value of these bits is verified by the experiment. Furthermore, the conditions of $X_{37,i}$ $(i = 2, 7, 17, 21)$, $X_{38,i}$ $(i = 17, 21)$, $Y_{30,i}$ $(i = 15, 21, 27, 30, 31)$ and $Y_{32,20}$ can be fulfilled by modifying or searching the message words. The above 12 conditions are marked with blue in Table 7. That means we can ensure 10 more conditions hold in the starting point compared with [MPS+13]. Therefore, the probability of the differential path can be improved and the needed starting points in phase 3 is decreased after the generalized message modification. Thus, a semi-free-start-collision attack on 48-step RIPEMD-160 can be obtained. The comparison of the starting points in [MPS+13] and in this paper are shown in Table 4. The procedure of the above calculation is illustrated in Table 3.

The process of computing $X_{37,i}$ $(i = 2, 7, 17, 21)$, $X_{38,i}$ $(i = 17, 21)$, $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ and $Y_{32,20}$ is independent of X_{36}, Y_{29} and Y_{31} which follow the differential paths. The calculation is described in the following. Furthermore, experiments are conducted, which confirm our reasoning and calculation process. Our C implementation can be found in [ver16a, ver16b].

Table 3: The Process of Finding a Starting Point

Step	1	2	3	4	5	6
Message word	m_{11}	m_{15}	m_8	m_3	m_{14}	m_{10}
MCV	X_{31}	Y_{26}	X_{32}, Y_{27}	X_{33}	X_{25}, Y_{20}	X_{34}
Step	7	8	9	10	11	12
Message word	m_2	m_5	m_{12}	m_9	m_0	m_6
MCV	X_{24}, X_{35}, Y_{19}	X_{23}, Y_{18}	Y_{28}	$X_{22}, X_{37}, X_{38}, Y_{30}, Y_{32}$	$X_{21}, X_{20}, X_{19}, X_{18}$	X_{17}

MCV denotes Modified Chaining Variable.

Table 4: The Comparison of the Components of the Starting Points

The Components of the Starting Points	Reference
X_i $(i = 17, ..., 35)$, Y_i $(i = 18, ..., 28)$, $X_{37,i}$ $(i = 2, 7)$	[MPS+13]
X_i $(i = 17, ..., 35)$, Y_i $(i = 18, ..., 28)$, $X_{37,i}$ $(i = 2, 7, 17, 21)$, $X_{38,i}$ $(i = 17, 21)$, $\Delta X_{37} =$ `0x200000`, $\Delta X_{38} = 0$ $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$, $Y_{32,20}$, $\Delta Y_{30} =$ `0x8200000`	Section 4.2

Calculate $X_{37,i}$ $(i = 2, 7, 17, 21)$

The message word m_4 is used in the process of merging both branches, so in the process of finding a starting point, X_{36} is unknown. From $F_3(X_{36}, X_{35}, (X_{34} \lll 10)) = (X_{36} \vee \neg X_{35}) \oplus (X_{34} \lll 10)$, we know that X_{36} will have no influence on the output of the boolean function F_3 if $X_{35} = 0$ is satisfied. Therefore,

$$X_{37} = (X_{33} \lll 10) + \left((X_{32} \lll 10) + F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9 + k_3^l\right) \lll 14$$

can be calculated under the condition that $X_{35} = 0$.

1. In order to calculate $X_{37,i}$ $(i = 17, 21)$, the conditions $X_{35,i} = 0$ $(i = 0, ..., 3, 5, 6)$ are added (there is the condition $X_{35,7} = 1$ in the differential path). Then bits 0-3, 5, 6 of $F_3(X_{36}, X_{35}, (X_{34} \lll 10))$ can be calculated. Let

$$T_1 = (X_{32} \lll 10) + ((\neg X_{34} \lll 10) \wedge \texttt{0xef}) + (m_9 \wedge \texttt{0xef}) + k_3^l,$$

then T_1 can be deduced. We can choose m_9 such that $T_{1,i} = 0$ $(i = 2, 3, 5)$. Meanwhile, we add the extra condition $X_{33,i} = 0$ $(i = 6, 8, 9)$ which is ensured to hold by modifying m_3. Thus, the carry coming from bit 16 (19) to bit 17 (20) in the process of computing X_{37} is stopped. Thanks to $X_{36,7} = 1$ being hold in the conditions of the differential path, we can compute bit 7 of $F_3(X_{36}, X_{35}, (X_{34} \lll 10))$, then $X_{37,21}$ can be calculated. Therefore, bits 17 and 21 of X_{37} can be calculated correctly if the differential path hold.

   The above procedure will definitely compute $X_{37,i}$ $(i = 17, 21)$ correctly under the condition that X_{36} is unknown, which is confirmed by the experiment. For the randomly chosen X_i $(i = 32, ..., 36)$ and m_9 such that the above conditions are satisfied. On one hand, X_{37} is calculated as

$$X_{37} = (X_{33} \lll 10) + \left((X_{32} \lll 10) + F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9 + k_3^l\right) \lll 14.$$

   On the other hand, X_{37} is computed as

$$X_{37} = (X_{33} \lll 10) + \left((X_{32} \lll 10) + ((\neg X_{34} \lll 10) \wedge \texttt{0xef}) + (m_9 \wedge \texttt{0xef}) + k_3^l\right) \lll 14.$$

The experiment shows that the two methods of calculation obtain the same value of $X_{37,17}$ ($X_{37,21}$) with probability 1, for randomly chosen X_{36} which satisfies the conditions of the differential path. This means $X_{37,17}$ and $X_{37,21}$ can be computed correctly independent of X_{36}. The additional conditions of computing $X_{37,i}$ ($i = 2, 7, 17, 21$) and $X_{38,i}$ ($i = 17, 21$) are shown in Table 10. It is worth noting that these extra conditions are not contradictory to the conditions of the differential path.

2. $X_{37,i}$ ($i = 2, 7$) can be computed as follows. Let $X_{35,i} = 0$ ($i = 17, ..., 25$), then bits 17-25 of $F_3(X_{36}, X_{35}, (X_{34} \lll 10))$ can be computed. Let

$$T_2 = (X_{32} \lll 10) + ((\neg X_{34} \lll 10) \wedge \texttt{0x3fe0000}) + (m_9 \wedge \texttt{0x3fe0000}) + k_3^l,$$

and force $T_{2,17} = 0$. Let $S_1 = (X_{32} \lll 10) + k_3^l$, and add the conditions $S_{1,i} = 0$ ($i = 16, 17$). Let $S_2 = F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9$, and force $S_{2,16} = 0$. Then there will be no carry coming from bit 17 to bit 18 in the process of computing T_2. Therefore, the 8 lowest bits of X_{37} especially $X_{37,i}$ ($i = 2, 7$) can be calculated correctly by

$$(X_{33} \lll 10) + \Big((X_{32} \lll 10) + ((\neg X_{34} \lll 10) \wedge \texttt{0x3fe0000}) + m_9 + k_3^l\Big) \lll 14.$$

In [MPS+13], in order to compute $X_{37,i}$ ($i = 2, 7$), the conditions that the 16-th bit of $(X_{32} \lll 10) + \Big(F_3(X_{36}, X_{35}, (X_{34} \lll 10)) \wedge \texttt{0x3ff0000}\Big) + (m_9 \wedge \texttt{0x3ff0000}) + k_3^l$ equals to zero and $X_{35,i} = 0$ ($i = 17, ..., 25$) are added. The experiment shows that the probability of $X_{37,i}$ ($i = 2, 7$) being computed correctly is less than 1. It is due to the carry from bit 17 to bit 18 can not always be stopped when computing $(X_{32} \lll 10) + F_3(X_{36}, X_{35}, (X_{34} \lll 10)) + m_9 + k_3^l$.

Calculate $X_{38,i}$ ($i = 17, 21$)

Since $F_3(X_{37}, X_{36}, (X_{35} \lll 10)) = (X_{37} \vee \neg X_{36}) \oplus (X_{35} \lll 10)$ and X_{36} is unknown, then the output of F_3 can be computed if $X_{37} = \texttt{0xffffffff}$. Taking into consideration the fact that

$$X_{38} = (X_{34} \lll 10) + \Big((X_{33} \lll 10) + F_3(X_{37}, X_{36}, (X_{35} \lll 10)) + m_{15} + k_3^l\Big) \lll 9,$$

we had better obtain bits 0-12 of $F_3(X_{37}, X_{36}, (X_{35} \lll 10))$ in order to compute $X_{38,i}$ ($i = 17, 21$). Thus, we need to force $X_{37,i} = 1$ ($i = 0, ..., 12$). We have computed $X_{37,i}$ ($i = 0, ..., 7$) above. Furthermore, $X_{37,i}$ ($i = 8, ..., 12$) can be calculated in the same way as the above method by forcing $X_{35,i} = 0$ ($i = 26, ..., 30$). However, there is $X_{37,2} = 0$ in the differential path. Thanks to $X_{36,2} = 1$ in the conditions of differential path, so we can get bit 2 of $F_3(X_{37}, X_{36}, (X_{35} \lll 10))$. Thus, bits 0-12 of $F_3(X_{37}, X_{36}, (X_{35} \lll 10))$ can be obtained. Let

$$T_3 = (X_{33} \lll 10) + \Big((\texttt{0x1ffb} \oplus (X_{35} \lll 10)) \wedge \texttt{0x1fff}\Big) + (m_{15} \wedge \texttt{0x1fff}) + k_3^l,$$

and force $T_{3,7} = 0$. Moreover, we add the extra condition $X_{34,6} = 0$ by modifying m_{10}. Therefore, the carry will be stopped at bit 16 in the process of computing X_{38}. Thus, $X_{38,i}$ ($i = 17, ..., 21$) can be calculated correctly by

$$(X_{34} \lll 10) + (T_3 \lll 9).$$

We verify the above computations by experiments, which show that $X_{37,i}$ ($i = 2, 7, 17, 21$) and $X_{38,i}$ ($i = 17, 21$) can be calculated correctly with probability 1 independent of X_{36}. Our C implementation verifying the computation of $X_{37,i}$ ($i = 2, 7, 17, 21$) and $X_{38,i}$ ($i = 17, 21$) can be found in [ver16a].

Moreover, we give an example in Table 5, where X_i and X'_i ($i = 17, 18, 19, 20, 21$) satisfy the conditions of the differential path in the left branch in Table 7. Then for randomly chosen X_{36} and X'_{36} (which satisfy the conditions of the differential path), we can get that $X_{37,i} = 0$ ($i = 2, 21$), $X_{37,i} = 1$ ($i = 0, 1, 3, ..., 12, 17$), $X_{38,i} = 0$ ($i = 17, 21$), $\Delta X_{37} = \texttt{0x200000}$ and $\Delta X_{38} = 0$ hold, which means X_{37}, X'_{37}, X_{38} and X'_{38} follow the differential path in Table 7. Therefore, the conditions

Table 5: An Example of the 17-38 Steps Differential Path in the Left Branch

X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{36} [c]
e17af3a8	cc966a05	b7573bfe	9336419e	f6920500	RV
X'_{17}	X'_{18}	X'_{19}	X'_{20}	X'_{21}	X'_{36}
e17af3e8	cc966a05	b7573bfe	9336419e	f6930500	RV′
m_0	m_2	m_3	m_5	m_6	m_8
7ed471ba	7e3d71d1	1edb7709	a3d9b37f	70477536	bdb9fa58
m_9	m_{10}	m_{11}	m_{12}	m_{14}	m_{15}
2e63c462	b0477764	67740efc	f9e7b23c	50321425	61b8f56a
(X_{37}, X'_{37}) $X_{37,i} = 0$ $(i = 2, 21)$, $X_{37,i} = 1$ $(i = 0, 1, 3, ..., 12, 17)$ $\Delta X_{37} =$ 200000				(X_{38}, X'_{38}) $X_{38,i} = 0$ $(i = 17, 21)$ $\Delta X_{38} = 0$	

[c] X_{36} and X'_{36} can be random values (RV) which satisfy the conditions of the differential path.
The values in the table are in hexadecimal notation, and "0x" is omitted because of limited space.

of X_{37}, X'_{37}, X_{38} and X'_{38} in Table 5 can be guaranteed to be hold even though X_{36} is unknown. Our C implementation of verifying the instance in Table 5 can be found in [ver16c].

Calculate $Y_{30,9}$

We need to know $Y_{30,9}$ in the process of computing $Y_{32,20}$, so we calculate $Y_{30,9}$ in advance. Because $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) = (Y_{29} \wedge (Y_{27} \lll 10)) \vee (Y_{28} \wedge \neg(Y_{27} \lll 10))$ and Y_{29} is unknown, if the condition $Y_{27} = 0$ is added, then

$$Y_{30} = (Y_{26} \lll 10) + \Big((Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r\Big) \lll 15$$

can be calculated. It is obviously that bits 23, 24, 26 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ can be calculated by adding the extra conditions $Y_{27,i} = 0$ $(i = 13, 14, 16)$ ($Y_{27,15} = 1$ is a condition of the differential path). Let

$$T_4 = (Y_{25} \lll 10) + (Y_{28} \wedge \texttt{0x5800000}) + m_9 + k_2^r,$$

$$R_1 = (Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r, \quad Q_1 = R_1 \lll 15,$$

$$R_2 = (Y_{25} \lll 10) + m_9 + k_2^r, \quad R_3 = F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)),$$

and force $T_{4,i} = 0$ $(i = 23, 24)$, $R_{2,i} = 0$ $(i = 23, 24, 25)$, and $R_{3,23} = 0$ (which is equivalent to $Y_{28,23} = 0$ and $Y_{27,13} = 0$). Therefore, $R_{1,i} = T_{4,i}$ $(i = 24, 26)$ can be calculated correctly because the carry is stopped at bit 23 in the process of computing R_1. Thus, $Q_{1,i}$ $(i = 7, 9)$ can be computed correctly and $Q_{1,7} = 0$. Furthermore, we add the extra condition $Y_{26,i} = 0$ $(i = 29, 30)$, which will stop the carry coming from the lower bits when computing $Y_{30,9}$ combined with the condition $Q_{1,7} = 0$. Therefore, $Y_{30,9}$ can be calculated correctly by

$$(Y_{26} \lll 10) + ((Y_{25} \lll 10) + (Y_{28} \wedge \texttt{0x5800000}) + m_9 + k_2^r) \lll 15.$$

The process of computing $Y_{30,i}$ $(i = 15, 21, 27, 30, 31)$ and $Y_{32,20}$ can be seen in Appendix. The additional conditions of computing $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ and $Y_{32,20}$ are shown in Table 11, which are not contradictory to the conditions of the differential path.

We verify the above computations by experiment, which show that even though Y_{29} is unknown, $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ can be calculated correctly with probability 1, and $Y_{32,20}$ can be obtained correctly with a probability of more than 0.9. Our C implementation verifying the computation of $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ and $Y_{32,20}$ can be found in [ver16b].

Moreover, we give an example in Table 6, where Y_i and Y'_i $(i = 17, 18, 19, 20, 21)$ satisfy the conditions of the differential path in the right branch in Table 7. Then for randomly chosen

Table 6: An Example of the 17-32 Steps Differential Path in the Right Branch

Y_{17} f2c0a544	Y_{18} 9453c4fc	Y_{19} b6052b22	Y_{20} ef1c7569	Y_{21} cb2693e7	Y_{29}, Y_{31} [d] RV
Y'_{17} f2c0a544	Y'_{18} 9453c4fc	Y'_{19} b6052b22	Y'_{20} ef1c7529	Y'_{21} cb2293e7	Y'_{29}, Y'_{31} RV′
m_2 74f1f030	m_5 f3bf86c1	m_8 87ffd902	m_9 47c76e30	m_{10} 8f04abf1	m_{12} 9df619c1
m_{13} b5eb9e8d	m_{14} bc9e3a19	m_{15} 10630dee			
(Y_{30}, Y'_{30}) $Y_{30,i} = 0$ $(i = 21, 27, 30, 31)$, $Y_{30,15} = 1$ $Y'_{30,i} = 0$ $(i = 30, 31)$, $Y'_{30,i} = 1$ $(i = 15, 21, 27)$					(Y_{32}, Y'_{32}) $Y_{32,20} = 0$ $Y'_{32,20} = 0$

[d] Y_{29}, Y'_{29}, Y_{31} and Y'_{31} can be random values (RV) which satisfy the conditions of the differential path. The values in the table are in hexadecimal notation, and "0x" is omitted because of limited space.

Y_{29}, Y'_{29}, Y_{31} and Y'_{31} (which satisfy the conditions of the differential path), we can get that $Y_{30,i} = 0$ $(i = 21, 27, 30, 31)$, $Y_{30,15} = 1$, $Y'_{30,i} = 0$ $(i = 30, 31)$, $Y'_{30,i} = 1$ $(i = 15, 21, 27)$ and $Y_{32,20} = Y'_{32,20} = 0$ hold with a probability of more than 0.9. Therefore, the conditions of Y_{30}, Y'_{30}, Y_{32} and Y'_{32} in Table 6 can be guaranteed to be hold with a probability of more than 0.9 even though Y_{29} and Y_{31} are unknown. Our C implementation of verifying the instance in Table 6 can be found in [ver16d].

A starting point example is presented in Table 7, which follows the differential path in [MPS+13]. Furthermore, for randomly chosen X_{36} and X'_{36} which satisfy the conditions of the differential path, the starting point makes sure that $X_{37,i} = 0$ $(i = 2, 21)$, $X_{37,i} = 1$ $(i = 7, 17)$, ΔX_{37} =0x200000, $X_{38,17} = 0$ and $\Delta X_{38} = 0$ hold with probability 1. Meanwhile, for randomly chosen Y_{29}, Y'_{29}, Y_{31} and Y'_{31} which satisfy the conditions of the differential path, the starting point ensures that $Y_{30,i} = 0$ $(i = 21, 27, 30, 31)$, $Y_{30,15} = 1$ and ΔY_{30} =0x8200000 hold with probability 1. Moreover, $Y_{32,20} = 0$ holds with a probability of more than 0.9 even though Y_{29} and Y_{31} are unknown. Our C implementation of verifying the starting point in Table 7 can be found in [ver16e].

In this paper, one of the main works focuses on providing a method to compute the values of $X_{37,i}$ $(i = 2, 7, 17, 21)$, $X_{38,i}$ $(i = 17, 21)$, $Y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ and $Y_{32,20}$ under the condition that X_{36}, Y_{29} and Y_{31} are unknown (these bits are listed in Table 4). Then the conditions on these bits are satisfied by message modification or exhaustive search. $Y_{30,9}$ is computed correctly in the starting point example shown in Table 7, and the purpose of computing $Y_{30,9}$ is to compute $Y_{32,20}$. It is noted that there is no condition on $Y_{30,9}$ in the differential path. There is a condition $X_{38,21} = 0$ in the differential path. We had no high performance computer to make the condition $X_{38,21} = 0$ hold. However, the value of $X_{38,21}$ is computed correctly (which is the most important part) in the starting point example shown in Table 7.

Uncontrolled probability.

1. After a starting point is discovered, the remaining free message words m_1, m_4, m_7 and m_{13} are used to make sure that there is a perfect match on the values of the five initial words of both branches, i.e. $X_i = Y_i$ $(i = 12, 13, 14, 15, 16)$. Furthermore, the success probability of the match phase is 2^{-32}.

2. In the left branch, after finding a starting point with the generalized message modification, there are 5 conditions $X_{36,i} = 0$ $(i = 11, 21)$, $X_{36,i} = 1$ $(i = 2, 7, 24)$ in Table 8 (except X_{57}, ..., X_{64}) that are not modified. Moreover, the uncontrolled probability of the left branch until step 56 is $2^{-5.4}$ by experiment.

Table 7: A Starting Point for the semi-free-start collision attack on 48-step RIPEMD-160

i	X_i	$m_{\pi^l(i)}$	$\pi^l(i)$	i	Y_i	$m_{\pi^r(i)}$	$\pi^r(i)$
12	-------- -------- -------- --------			12	-------- -------- -------- --------		
13	-------- -------- -------- --------			13	-------- -------- -------- --------		
14	-------- -------- -------- --------			14	-------- -------- -------- --------		
15	-------- -------- -------- --------			15	-------- -------- -------- --------		
16	-------- -------- -------- --------	x------- -------- -------- --------	7	16	-------- -------- -------- --------	-------- -------- -------- --------	6
17	-------- -------- -------- -n------	-------- -------- -------- --------	4	17	-------- -------- -------- --------	11111001 10100110 10111011 00110110	11
18	-------- -------- -------- -0------	-------- -------- -------- --------	13	18	10011111 01110000 00100001 11010101	10001001 00001100 01001111 00101111	3
19	-------- -------1 -------- -1------	-------- -------- -------- --------	1	19	00010101 01000110 01100101 01001011	x------- -------- -------- --------	7
20	-------- -------0 -------- --------	00101010 11001000 10111110 10110011	10	20	11001101 10000100 00011101 1u100001	-------- -------- -------- --------	0
21	-------- -------n -------- --------	-------- -------- -------- --------	6	21	11101000 10001u10 11011011 11010111	-------- -------- -------- --------	13
22	10010001 11111100 00100100 11100100	10000110 01101010 01001100 10011100	15	22	010unn00 10001101 000nuuuu 0101110u	00100101 00000010 11100101 11011110	5
23	11110111 11001110 11110000 01001000	10001001 00001100 01001111 00101111	3	23	11100un0 0110uuu1 11010100 1un01001	00101010 11001000 10111110 10110011	10
24	11011101 01000010 00010111 10110101	10010101 01100110 00100100 11000101	12	24	u11n1000 11111111 1100u111 01nn0010	00111010 10111010 00100010 11010111	14
25	11011n00 01110111 01100001 11000101	-------- -------- -------- --------	0	25	00u11011 n0001111 1101u101 1110n1nu	10000110 01101010 01001100 10011100	15
26	00110001 100101un nnnn11nu u1000110	01101100 00010111 11001101 00110000	9	26	00011111 11100011 000n1nnn 0nnnu110	11111111 10001101 10101011 01001001	8
27	1n01n0un 0u000100 00001011 10100101	00100101 00000010 11100101 11011110	5	27	10001u00 00uu001n 10000010 10000unn	10010101 01100110 00100100 11000101	12
28	10100011 101000nu 00n01nu1 u0un0110	10001011 01001111 00011100 11010101	2	28	1101u100 00n10000 00001100 00n000n1	-------- -------- -------- --------	4
29	00nun001 110u010u 01010101 10u0nnu1	00111010 10111010 00100010 11010111	14	29	00------ ---u--1- 0u-11u-- ---0---0	01101100 00010111 11001101 00110000	9
30	10001111 uu010n10 1u11nu1n 100nn000	11111001 10100110 10111011 00110110	11	30	00--n--- --n----- 1---1--- --------	-------- -------- -------- --------	1
31	1n10010u 0uu10011 000011n0 0001n100	11111111 10001101 10101011 01001001	8	31	-1------ --1----- -----u-- --1-----	10001011 01001111 00011100 11010101	2
32	11011000 0010010u 01111u10 10000111	10001001 00001100 01001111 00101111	3	32	-------- --u----- -----1-- --0-----	10000110 01101010 01001100 10011100	15
33	1n11101u 11u01111 11111100 10110011	00101010 11001000 10111110 10110011	10	33	-------- --1----- -------- --------	00100101 00000010 11100101 11011110	5
34	01101000 10011011 00101110 u0011000	00111010 10111010 00100010 11010111	14	34	-------- -------- -------- --------	-------- -------- -------- --------	1
35	1000000n 00n00000 10101111 10010000	-------- -------- -------- --------	4	35	-------- -------- -------- --------	10001001 00001100 01001111 00101111	3
36	-------1 --0----- ----0--- u----1--	01101100 00010111 11001101 00110000	9	36	-------- -------- -------- --------	x------- -------- -------- --------	7
37	-------- --n---1- ----1111 11111011	10000110 01101010 01001100 10011100	15	37	-------- -------- -------- --------	00111010 10111010 00100010 11010111	14
38	-------- --0---0- -------- --------	11111111 10001101 10101011 01001001	8	38	-------- -------- -------- --------	-------- -------- -------- --------	6
39	-------- -------- -------- --------	-------- -------- -------- --------	1	39	-------- -------- -------- --------	01101100 00010111 11001101 00110000	9
40	-------- -------- -------- --------	10001011 01001111 00011100 11010101	2	40	-------- -------- -------- --------	11111001 10100110 10111011 00110110	11
41	-------- -------- -------- --------	x------- -------- -------- --------	7	41	-------- -------- -------- --------	11111111 10001101 10101011 01001001	8
42	-------- -------- -------- --------	-------- -------- -------- --------	0	42	-------- -------- -------- --------	10010101 01100110 00100100 11000101	12
43	-------- -------- -------- --------	-------- -------- -------- --------	6	43	-------- -------- -------- --------	10001011 01001111 00011100 11010101	2
44	-------- -------- -------- --------	-------- -------- -------- --------	13	44	-------- -------- -------- --------	00101010 11001000 10111110 10110011	10
45	-------- -------- -------- --------	11111001 10100110 10111011 00110110	11	45	-------- -------- -------- --------	-------- -------- -------- --------	0
46	-------- -------- -------- --------	00100101 00000010 11100101 11011110	5	46	-------- -------- -------- --------	-------- -------- -------- --------	4
47	-------- -------- -------- --------	10010101 01100110 00100100 11000101	12	47	-------- -------- -------- --------	-------- -------- -------- --------	13
48	-------- -------- -------- --------	-------- -------- -------- --------	1	48	-------- -------- -------- --------	11111111 10001101 10101011 01001001	8
49	-------- -------- -------- --------	01101100 00010111 11001101 00110000	9	49	-------- -------- -------- --------	-------- -------- -------- --------	6
50	-------- -------- -------- --------	11111001 10100110 10111011 00110110	11	50	-------- -------- -------- --------	-------- -------- -------- --------	4
51	-------- -------- -------- --------	00101010 11001000 10111110 10110011	10	51	-------- -------- -------- --------	-------- -------- -------- --------	1
52	-------- -------- -------- --------	-------- -------- -------- --------	0	52	-------- -------- -------- --------	10001001 00001100 01001111 00101111	3
53	-------- -------- -------- --------	11111111 10001101 10101011 01001001	8	53	-------- -------- -------- --------	11111001 10100110 10111011 00110110	11
54	-------- -------- -------- --------	10010101 01100110 00100100 11000101	12	54	-------- -------- -------- --------	10000110 01101010 01001100 10011100	15
55	-------- -------- -------- --------	-------- -------- -------- --------	4	55	-------- -------- -------- --------	-------- -------- -------- --------	0
56	-------- -------- -------- --------	-------- -------- -------- --------	13	56	-------- -------- -------- --------	00100101 00000010 11100101 11011110	5
57	-----0-- -------- -------- --------	10001001 00001100 01001111 00101111	3	57	-------- -------- -------- --------	10010101 01100110 00100100 11000101	12
58	-----1-- -------- -------- --------	x------- -------- -------- --------	7	58	-------- -------- -------- --------	10001011 01001111 00011100 11010101	2
59	-------- -------- -------- ---n----	10000110 01101010 01001100 10011100	15	59	-------- -------- -------- --------	-------- -------- -------- --------	13
60	-------- -------- -------- --------	00111010 10111010 00100010 11010111	14	60	-------- -------- -------- --------	01101100 00010111 11001101 00110000	9
61	-------- -------- -------- --01----	00100101 00000010 11100101 11011110	5	61	-------- -------- -------- --------	x------- -------- -------- --------	7
62	-------- -------- -------- --------	-------- -------- -------- --------	6	62	-------- -------- -------- --u-----	00101010 11001000 10111110 10110011	10
63	-------- -------- nu------ --------	10001011 01001111 00011100 11010101	2	63	-------- -------- -------- ---0----	00111010 10111010 00100010 11010111	14
64	-------- -------- -------- --------			64	-------- -------- -------- --------		

The corresponding conditions on the positions which are red are inconvenient to express by the notations in [DCR06], and they are presented in Table 9.

3. In the right branch, the conditions $Y_{30,i}$ ($i = 15, 21, 27, 30, 31$) and $Y_{32,20}$ in Table 9 are satisfied in the starting points. Therefore, there are only 24 uncontrolled conditions between Y_{29} and Y_{33}. Moreover, the uncontrolled probability of the right branch until step 59 is $2^{-29.6}$ by experiment.

4. The collision probability of the differential path in steps 57-64 of the left branch and in steps 60-64 of the right branch is $2^{-11.3}$ by experiment [MPS+13].

Therefore, the uncontrolled probability is $2^{-32} \times 2^{-5.4} \times 2^{-29.6} \times 2^{-11.3} \times 0.9 = 2^{-78.5}$ in total.

Complexity evaluation.

1. The $2^{78.5}$ starting points do not need to be generated from the beginning. A new start point can be produced by randomizing m_6. Once all the possible choices of m_6 have been used, the freedom degrees of m_0, m_9 and m_5 can be used. According to Table 7, m_6 is used to fulfill one condition on X_{17}. In order to use the freedom degree of m_6, we can randomly choose X_{17} satisfying the condition $X_{17,6} = 0$ and deduce m_6. Therefore, the complexity of finding a new starting point from a known one is about $2 \div (48 \times 2) \approx 2^{-5.6}$ of the 48-step compression function. For randomizing m_0, m_9 and m_5, the corresponding number of times we have to regenerate them is not the bottleneck of the attack on 48-step RIPEMD-160. Therefore, the complexity of generating all the required starting points is $2^{78.5} \times 2^{-5.6} = 2^{72.9}$.

2. The process of merging both branches is the same as the merging process proposed by Mendel *et al.* [MPS+13]. The values of m_{13} and m_4 can be deduced from $X_{16} = Y_{16}$ and $X_{13} = Y_{13}$ respectively, and the complexity can be negligible. The bottleneck of the merging complexity is finding m_1 and m_7 from the conditions $X_{15} = Y_{15}$ and $X_{14} = Y_{14}$. The problem of finding the values of m_1 and m_7 is equivalent to solving the equation $X + C_0 = (C_1 + X \lll 8) \ggg 15$, where C_0 and C_1 are constants. The attackers pre-compute m_1 and m_7 and store them in a table for all the 2^{64} possible values of C_0 and C_1, with a time complexity of 2^{73} and memory complexity of 2^{64}. Moreover, in the merging process, the table lookup is estimated using a RAM access and the implementation of one merging needs about 145 cycles, while the OPENSSL implementation of RIPEMD-160 compression function is about 1040 cycles. Therefore, one merging costs about $145 \div (1040 \times 48 \div 80) \approx 2^{-2.1}$ calls of the 48-step compression function.

From the reasoning above, we can conclude that the complexity of the semi-free start collision attack on 48-step RIPEMD-160 is $2^{78.5-2.1} = 2^{76.4}$.

4 Conclusions

In this paper, we present a method and give a partial answer to calculate the theoretical probability of a given differential path in RIPEMD-160, the step function of which is no longer a T-function. Furthermore, we propose a method to carry out the message modification such that a step differential path holds with probability 1 after message modification. Meanwhile, we propose a semi-free start collision attack on 48-step RIPEMD-160, and our result improves the previously best known semi-free start collision by 6 rounds. Our semi-free start collision attack uses the differential path constructed by Mendel *et al.* and improves the probabilistic part by a factor of about 2^{10}. The future work includes completely answering the open problem of computing the theoretical probability of a given differential path in RIPEMD-160.

Acknowledgments

We would like to thank Florian Mendel and the anonymous reviewers who helped improve this paper. This work is supported by the National Natural Science Foundation of China (Nos. 61572125, 61632012, 61373142), and Shanghai High-Tech Field Project (No. 16511101400).

References

[BP95] Antoon Bosselaers and Bart Preneel. *Integrity Primitives for Secure Information Systems: Final Ripe Report of Race Integrity Primitives Evaluation*. Number 1007. Springer Science & Business Media, 1995.

[Dam89] Ivan Damgård. A design principle for hash functions. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 416–427, 1989.

[Dau05] Magnus Daum. *Cryptanalysis of Hash functions of the MD4-family*. PhD thesis, Ruhr-Universität Bochum, Universitätsbibliothek, 2005.

[DBP96] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *International Workshop on Fast Software Encryption*, pages 71–82. Springer, 1996.

[DCR06] Christophe De Cannière and Christian Rechberger. Finding SHA-1 characteristics: General results and applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2006.

[DEM14] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Analysis of SHA-512/224 and SHA-512/256. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 612–630. Springer, 2014.

[Dob97] Hans Dobbertin. RIPEMD with two-round compress function is not collision-free. *Journal of Cryptology*, 10(1):51–69, 1997.

[FLN] Pierre-Alain Fouque, Gaetan Leurent, and Phong Nguyen. Automatic search of differential path in md4. ecrypt hash worshop–cryptology eprint archive, report, 2007/206 (2007).

[Int11] International Standardization of Organization (ISO). International Standard- ISO/IEC 29192-2, Information technology-Security techniques-Lightweight cryptography -Part 2: Block ciphers. 2011.

[Kli06] Vlastimil Klima. Tunnels in hash functions: Md5 collisions within a minute. *IACR Cryptology ePrint Archive*, 2006:105, 2006.

[LP13] Franck Landelle and Thomas Peyrin. Cryptanalysis of full ripemd-128. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 228–244. Springer, 2013.

[LP16] Franck Landelle and Thomas Peyrin. Cryptanalysis of full RIPEMD-128. *Journal of Cryptology*, 29(4):927–951, 2016.

[Mer89] Ralph C Merkle. One way hash functions and DES. In *Conference on the Theory and Application of Cryptology*, pages 428–446. Springer, 1989.

[MNS11] Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding SHA-2 characteristics: searching through a minefield of contradictions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 288–307. Springer, 2011.

[MNS12] Florian Mendel, Tomislav Nad, and Martin Schläffer. Collision attacks on the reduced dual-stream hash function RIPEMD-128. In *Fast Software Encryption*, pages 226–243. Springer, 2012.

[MNS13] Florian Mendel, Tomislav Nad, and Martin Schläffer. Improving local collisions: new attacks on reduced SHA-256. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 262–278. Springer, 2013.

[MNSS12] Florian Mendel, Tomislav Nad, Stefan Scherz, and Martin Schläffer. Differential attacks on reduced RIPEMD-160. In *International Conference on Information Security*, pages 23–38. Springer, 2012.

[MPRR06] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. On the collision resistance of RIPEMD-160. In *International Conference on Information Security*, pages 101–116. Springer, 2006.

[MPS+13] Florian Mendel, Thomas Peyrin, Martin Schläffer, Lei Wang, and Shuang Wu. Improved cryptanalysis of reduced RIPEMD-160. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 484–503. Springer, 2013.

[OSS10] Chiaki Ohtahara, Yu Sasaki, and Takeshi Shimoyama. Preimage attacks on step-reduced RIPEMD-128 and RIPEMD-160. In *International Conference on Information Security and Cryptology*, pages 169–186. Springer, 2010.

[SHA] NIST SHA. SHA-3 cryptographic hash algorithm competition.

[SKP16] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full SHA-1. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 459–483. Springer, 2016.

[SW12] Yu Sasaki and Lei Wang. Distinguishers beyond three rounds of the RIPEMD-128/-160 compression functions. In *International Conference on Applied Cryptography and Network Security*, pages 275–292. Springer, 2012.

[ver16a] Verification of the computation of $x_{37,i}$ $(i = 2, 7, 17, 21)$ and $x_{38,i}$ $(i = 17, 21)$. 2016. https://github.com/Crypt-CNS/RIPEMD-160/tree/Reference1.

[ver16b] Verification of the computation of $y_{30,i}$ $(i = 9, 15, 21, 27, 30, 31)$ and $y_{32,20}$. 2016. https://github.com/Crypt-CNS/RIPEMD-160/tree/Reference2.

[ver16c] Verification of the instance in table 5. 2016. https://github.com/Crypt-CNS/RIPEMD-160/tree/Reference3.

[ver16d] Verification of the instance in table 6. 2016. https://github.com/Crypt-CNS/RIPEMD-160/tree/Reference4.

[ver16e] Verification of the instance in table 7. 2016. https://github.com/Crypt-CNS/RIPEMD-160/tree/Reference5.

[Wan14] Gaoli Wang. Practical collision attack on 40-step RIPEMD-128. In *Cryptographers' Track at the RSA Conference*, pages 444–460. Springer, 2014.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–18. Springer, 2005.

[WS14] Gaoli Wang and Yanzhao Shen. (pseudo-) preimage attacks on step-reduced HAS-160 and RIPEMD-160. In *International Conference on Information Security*, pages 90–103. Springer, 2014.

[WW08] Gaoli Wang and Meiqin Wang. Cryptanalysis of reduced RIPEMD-128. *Journal of Software*, 19(9):2442–2448, 2008.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 19–35. Springer, 2005.

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

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *Annual International Cryptology Conference*, pages 17–36. Springer, 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2005.

[YWYP06] Hongbo Yu, Xiaoyun Wang, Aaram Yun, and Sangwoo Park. Cryptanalysis of the full HAVAL with 4 and 5 passes. In *International Workshop on Fast Software Encryption*, pages 89–110. Springer, 2006.

[YWZW05] Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The second-preimage attack on MD4. In *International Conference on Cryptology and Network Security*, pages 1–12. Springer, 2005.

A Some propositions of computing the differential probability

Proposition 2 Let $y = x_1 + (x_2 + x_3) \lll s$, $y' = x_1 + (x_2 + x'_3) \lll s$, here x_1, x_2, x_3 and x'_3 are 32-bit words. If $\Delta x_3 = [i, j]$ (i.e., $x_{3,i} = 0$, $x_{3,j} = 0$, $x'_{3,i} = 1$, $x'_{3,j} = 1$, $x_{3,k} = x'_{3,k}$ $(0 \leq k \leq 31, k \neq i, j)$) and $i < j \leq 31 - s$, then the probability that $\Delta^+ y$ equals to $2^{i+s} + 2^{j+s}$ can be computed as follows:

$$Pr[\Delta^+ y = 2^{i+s} + 2^{j+s}] = 1 - \frac{1}{2^{32-s-i}} - \frac{1}{2^{32-s-j}}.$$

Proof. Let $r_1 = x_2 + x_3$, $r_2 = r_1 \lll s$, $r'_1 = x_2 + x'_3$, $r'_2 = r'_1 \lll s$, if the i-th bit difference of Δx_3 results in

$$\begin{array}{rcl} \Delta r_1 & = & [i], \\ or\ \ \Delta r_1 & = & [-i, i+1], \\ or\ \ \Delta r_1 & = & [-i, -(i+1), i+2], \\ & ... & \\ or\ \ \Delta r_1 & = & [-i, -(i+1), ..., -(j-2), j-1], \end{array}$$

and the j-th bit difference of Δx_3 results in

$$\begin{array}{rcl} \Delta r_1 & = & [j], \\ or\ \ \Delta r_1 & = & [-j, j+1], \\ or\ \ \Delta r_1 & = & [-j, -(j+1), j+2], \\ & ... & \\ or\ \ \Delta r_1 & = & [-j, -(j+1), ..., -(30-s), 31-s], \end{array}$$

then $\Delta^+ y = 2^{i+s} + 2^{j+s}$ hold obviously.

Meanwhile, besides the above circumstances, it is not difficult to observe that $\Delta^+ r_2 = 2^{i+s} + 2^{j+s}$ (i.e., $\Delta^+ y = 2^{i+s} + 2^{j+s}$) is equivalent to

$$\begin{array}{rcl} \Delta r_1 & = & [-i, -(i+1), ..., -(j-1), j+1], \\ or\ \ \Delta r_1 & = & [-i, -(i+1), ..., -(j-1), -(j+1), j+2], \\ & ... & \\ or\ \ \Delta r_1 & = & [-i, -(i+1), ..., -(j-1), -(j+1), -(j+2), ..., -(30-s), 31-s], \end{array}$$

Therefore,

$$\begin{array}{rl} & Pr[\Delta^+ y = 2^{i+s} + 2^{j+s}] \\ = & (\frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{j-i}}) \times (\frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{32-s-j}}) + \frac{1}{2^{j-i+1}} + \frac{1}{2^{j-i+2}} + ... + \frac{1}{2^{31-s-i}} \\ = & 1 - \frac{1}{2^{32-s-i}} - \frac{1}{2^{32-s-j}}. \end{array}$$

□

From Proposition 2, it is easy to get the following corollary.

Corollary 2 Let $y = x_1 + (x_2 + x_3) \lll s$, $y' = x_1 + (x_2 + x'_3) \lll s$, here x_1, x_2, x_3 and x'_3 are 32-bit words. If $\Delta x_3 = [-i, -j]$ (i.e., $x_{3,i} = 1$, $x_{3,j} = 1$, $x'_{3,i} = 0$, $x'_{3,j} = 0$, $x_{3,k} = x'_{3,k}$ $(0 \leq k \leq 31, k \neq i, j)$) and $i < j \leq 31 - s$, then the probability that $\Delta^+ y$ equals to $-2^{i+s} - 2^{j+s}$ can be computed as follows:

$$Pr[\Delta^+ y = -2^{i+s} - 2^{j+s}] = 1 - \frac{1}{2^{32-s-i}} - \frac{1}{2^{32-s-j}}.$$

Proposition 3 Let $y = x_1 + (x_2 + x_3) \lll s$, $y' = x_1 + (x_2 + x_3') \lll s$, here x_1, x_2, x_3 and x_3' are 32-bit words. If $\Delta x_3 = [i, j]$ (i.e., $x_{3,i} = 0$, $x_{3,j} = 0$, $x'_{3,i} = 1$, $x'_{3,j} = 1$, $x_{3,k} = x'_{3,k}$ $(0 \le k \le 31, k \ne i, j)$) and $32 \le i + s < j + s$, then the probability that $\Delta^+ y$ equals to $2^{i+s-32} + 2^{j+s-32}$ can be computed as follows:

$$Pr[\Delta^+ y = 2^{i+s-32} + 2^{j+s-32}] = 1 - \frac{1}{2^{32-i}} - \frac{1}{2^{32-j}}.$$

Proof. Let $r_1 = x_2 + x_3$, $r_1' = x_2 + x_3'$, from $\Delta^+ y = 2^{i+s-32} + 2^{j+s-32}$, we can get that the i-th bit difference of Δx_3 results in

$$\begin{aligned} \Delta r_1 &= [i], \\ or\ \ \Delta r_1 &= [-i, i+1], \\ or\ \ \Delta r_1 &= [-i, -(i+1), i+2], \\ &\ldots \\ or\ \ \Delta r_1 &= [-i, -(i+1), ..., -30, 31], \end{aligned}$$

and the j-th bit difference of Δx_3 results in

$$\begin{aligned} \Delta r_1 &= [j], \\ or\ \ \Delta r_1 &= [-j, j+1], \\ or\ \ \Delta r_1 &= [-j, -(j+1), j+2], \\ &\ldots \\ or\ \ \Delta r_1 &= [-j, -(j+1), ..., -30, 31]. \end{aligned}$$

It is obvious that the above circumstances except for $\Delta r_1 = [-i, -(i+1), ..., -30, 31]$ (from the i-th bit difference of Δx_3) and $\Delta r_1 = [-j, -(j+1), ..., -30, 31]$ (from the j-th bit difference of Δx_3) will lead to $\Delta^+ y = 2^{i+s-32} + 2^{j+s-32}$. Therefore,

$$\begin{aligned} & Pr[\Delta^+ y = 2^{i+s-32} + 2^{j+s-32}] \\ = \ & (\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{32-i}}) \times (\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + ... + \frac{1}{2^{32-j}}) - \frac{1}{2^{32-i}} \times \frac{1}{2^{32-j}} \\ = \ & (1 - \frac{1}{2^{32-i}}) \times (1 - \frac{1}{2^{32-j}}) - \frac{1}{2^{32-i}} \times \frac{1}{2^{32-j}} \\ = \ & 1 - \frac{1}{2^{32-i}} - \frac{1}{2^{32-j}}. \end{aligned}$$

□

Proposition 4 For the i-th step function $X_i = (X_{i-4} \lll 10) + \Big((X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m + k\Big) \lll s$, where m, k and s depend on the step i. If the input difference is (ΔX_{i-5}, ΔX_{i-4}, ΔX_{i-3}, ΔX_{i-2}, ΔX_{i-1})=(0, 0, 0, 0, 0), and the message difference Δm=[j], then there are the following properties:

1. When $j + s \le 31$, the probability of $\Delta X_i = [-(j+s), -(j+s+1), ..., -(j+s+t-1), j+s+t]$ is $\frac{1}{2^{t+1}} \times (1 - \frac{1}{2^{32-s-j}})$, where $j + s + t \le 31$.

2. When $j + s > 31$, the probability of $\Delta X_i = [-(j+s-32), -(j+s+1-32), ..., -(j+s+t-1-32), j+s+t-32]$ is $\frac{1}{2^{t+1}} \times (1 - \frac{1}{2^{32-j}})$, where $j + s + t - 32 \le 31$.

Especially, when $t = 0$, the probability of $\Delta X_i = [j + s(mod 32)]$ is $\frac{1}{2} \times (1 - \frac{1}{2^{32-s-j}})$ $(j + s \le 31)$ or $\frac{1}{2} \times (1 - \frac{1}{2^{32-j}})$ $(j + s > 31)$.

Proof. (1) When $j + s \leq 31$,

$$\Delta X_i = [-(j+s), -(j+s+1), ..., -(j+s+t-1), j+s+t] \Longleftrightarrow \begin{cases} \Delta^+ X_i = 2^{j+s} \\ X_{i,j+s} = 1 \\ X_{i,j+s+1} = 1 \\ \cdots \\ X_{i,j+s+t-1} = 1 \\ X_{i,j+s+t} = 0 \end{cases}$$

Combined with Proposition 1, we can get the probability of $\Delta X_i = [-(j+s), -(j+s+1), ..., -(j+s+t-1), j+s+t]$ is $\frac{1}{2^{t+1}} \times (1 - \frac{1}{2^{32-s-j}})$.

(2) When $j + s > 31$,

$$\Delta X_i = [-(j+s-32), -(j+s+1-32), ..., -(j+s+t-1-32), j+s+t-32]$$

$$\Longleftrightarrow \begin{cases} \Delta^+ X_i = 2^{j+s-32} \\ X_{i,j+s-32} = 1 \\ X_{i,j+s+1-32} = 1 \\ \cdots \\ X_{i,j+s+t-1-32} = 1 \\ X_{i,j+s+t-32} = 0 \end{cases}$$

Combined with Proposition 1, we can get the probability of $\Delta X_i = [-(j+s-32), -(j+s+1-32), ..., -(j+s+t-1-32), j+s+t-32]$ is $\frac{1}{2^{t+1}} \times (1 - \frac{1}{2^{32-j}})$. □

B The second example to illustrate the process of message modification

The first example of the message modification is shown in Section 2.2. Following is the second example to illustrate the message modification.

Example 2. For the i-th step function $X_i = (X_{i-4} \lll 10) + \Big((X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m + k\Big) \lll s$, where m, k and s depend on the step i. Denote $r_1 = (X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m + k$, $r'_1 = (X_{i-5} \lll 10) + f(X_{i-1}, X_{i-2}, (X_{i-3} \lll 10)) + m' + k$, $r_2 = r_1 \lll s$ and $r'_2 = r'_1 \lll s$. Let $\Delta m = [26]$ and $s = 3$, in order to ensure $\Delta X_i = [29]$, the message modification can be processed as follows.

Step 1. If $\Delta^+ X_i \neq 2^{29}$, from the proof of Proposition 1, we know that $\Delta r_1 = [-26, -27, -28, 29]$, $[-26, -27, -28, -29, 30]$, $[-26, -27, -28, -29, -30, 31]$ or $[-26, -27, -28, -29, -30, -31]$. In order to ensure $\Delta^+ X_i = 2^{29}$, the message word m can be modified as $m \longleftarrow m \pm 2^{28}$, $m \longleftarrow m \pm 2^{27}$ or $m \longleftarrow m \pm 2^{26}$. The reasons are explained as follows.

(1) If $r_{1,31} = 1$ and $r'_{1,31} = 0$, in order to make $\Delta^+ X_i = 2^{29}$, m can be modified in the following ways:

① The message word m can be modified as $m \longleftarrow m - 2^{28}$, then the most six significant bits of r_1 and r'_1 are $r_{1,t} = 1$ $(t = 26, 27, 29, 30, 31)$, $r_{1,28} = 0$, $r'_{1,t} = 1$ $(t = 28, 29, 30, 31)$, and $r'_{1,t} = 0$ $(t = 26, 27)$, which means $\Delta r_1 = [-26, -27, 28]$, thus $\Delta r_2 = [-29, -30, 31]$. Therefore, $\Delta^+ X_i = -2^{29} - 2^{30} + 2^{31} = 2^{29}$.

② Similarly, m can be modified as $m \longleftarrow m + 2^{28}$, then $r_{1,t} = 1$ $(t = 26, 27)$, $r_{1,t} = 0$ $(t = 28, 29, 30, 31)$, $r'_{1,t} = 0$ $(t = 26, 27, 29, 30, 31)$, and $r'_{1,28} = 1$, which means $\Delta r_1 = [-26, -27, 28]$, thus $\Delta^+ X_i = 2^{29}$.

③ The message word m can be modified as $m \longleftarrow m - 2^{27}$, then the most six significant bits of r_1 and r'_1 are $r_{1,t} = 1$ ($t = 26, 28, 29, 30, 31$), $r_{1,27} = 0$, $r'_{1,t} = 1$ ($t = 27, 28, 29, 30, 31$), and $r'_{1,26} = 0$, which means $\Delta r_1 = [-26, 27]$, thus $\Delta r_2 = [-29, 30]$. Therefore, $\Delta^+ X_i = -2^{29} + 2^{30} = 2^{29}$.

④ Similarly, m can be modified as $m \longleftarrow m + 2^{27}$, then $r_{1,t} = 0$ ($t = 27, 28, 29, 30, 31$), $r_{1,26} = 1$, $r'_{1,t} = 0$ ($t = 26, 28, 29, 30, 31$), and $r'_{1,27} = 1$, which means $\Delta r_1 = [-26, 27]$, thus $\Delta^+ X_i = 2^{29}$.

⑤ The message word m can be modified as $m \longleftarrow m - 2^{26}$, then the most six significant bits of r_1 and r'_1 are $r_{1,t} = 1$ ($t = 27, 28, 29, 30, 31$), $r_{1,26} = 0$ and $r'_{1,t} = 1$ ($t = 26, 27, 28, 29, 30, 31$), which means $\Delta r_1 = [26]$, thus $\Delta r_2 = [29]$. Therefore, $\Delta^+ X_i = 2^{29}$.

⑥ Similarly, m can be modified as $m \longleftarrow m + 2^{26}$, then $r_{1,t} = 0$ ($t = 26, 27, 28, 29, 30, 31$), $r'_{1,26} = 1$ and $r'_{1,t} = 0$ ($t = 27, 28, 29, 30, 31$), which means $\Delta r_1 = [26]$, thus $\Delta^+ X_i = 2^{29}$.

(2) If Δr_1 equals to $[-26, -27, -28, 29]$, $[-26, -27, -28, -29, 30]$ or $[-26, -27, -28, -29, -30, 31]$, similar to Step (1), the message word m can be modified according to one of the following three ways: $m \longleftarrow m \pm 2^{28}$, $m \longleftarrow m \pm 2^{27}$ or $m \longleftarrow m \pm 2^{26}$. After the message modification, we will get $\Delta r_1 = [-26, -27, 28]$, $\Delta r_1 = [-26, 27]$ or $\Delta r_1 = [26]$, respectively. Thus $\Delta^+ X_i = 2^{29}$ can be obtained.

Step 2. After Step 1, we can get $\Delta^+ X_i = 2^{29}$, which means Δr_1 equals to $[26]$, $[-26, 27]$ or $[-26, -27, 28]$. If $X_{i,29} \neq 0$, the message word m can be modified as $m \longleftarrow m + 2^{26}$ or $m \longleftarrow m - 2^{26}$ according to different circumstances. We give two examples to illustrate the modification.

(1) If $r'_{1,31\sim26} = 000001$ and $r_{1,31\sim26} = 000000$, then the modification of m as $m \longleftarrow m - 2^{26}$ will violate the condition $\Delta^+ X_i = 2^{29}$. Therefore, m can only be modified as $m \longleftarrow m + 2^{26}$. This modification will ensure $\Delta^+ X_i = 2^{29}$ and $X_{i,29}$ is flipped.

(2) If $r'_{1,28\sim26} = 101$ and $r_{1,28\sim26} = 100$, then m can be modified as $m \longleftarrow m + 2^{26}$ or $m \longleftarrow m - 2^{26}$. Both modification will ensure $\Delta^+ X_i = 2^{29}$ and $X_{i,29}$ is flipped.

C The Process of Computing $Y_{30,i}$ ($i = 15, 21, 27, 30, 31$) and $Y_{32,20}$

C.1 The Process of Computing $Y_{30,15}$

Y_{29} is unknown, from $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) = (Y_{29} \land (Y_{27} \lll 10)) \lor (Y_{28} \land \neg(Y_{27} \lll 10))$ we can get that bit 0 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ can be calculated by adding the extra conditions $Y_{27,22} = 0$. Meanwhile, $Y_{29,i} = 0$ ($i = 30, 31$) hold in the conditions of the differential path, which will make bits 30, 31 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ can be calculated. Let

$$T_5 = (Y_{25} \lll 10) + (Y_{28} \land (\neg Y_{27} \lll 10) \land \texttt{0xc0000001}) + m_9 + k_2^r,$$

$$R_4 = (Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r, \qquad Q_4 = R_4 \lll 15,$$

and force $T_{5,i} = 0$ ($i = 30, 31$). Then bits 0, 31 of R_4 can be calculated correctly, which means bits 14, 15 of Q_4 can be computed correctly. Furthermore, we add the extra condition $Y_{26,4} = 0$ (which has been hold in the differential path), which will make bit 15 of

$$Y_{30} = (Y_{26} \lll 10) + \Big((Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r\Big) \lll 15$$

can be calculated correctly by

$$(Y_{26} \lll 10) + \Big((Y_{25} \lll 10) + (Y_{28} \land (\neg Y_{27} \lll 10) \land \texttt{0xc0000001}) + m_9 + k_2^r\Big) \lll 15$$

since the carry is stopped in bit 14 when computing Y_{30}.

C.2 The Process of Computing $Y_{30,21}$

Y_{29} is unknown, then bit 6 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ can be calculated by adding the extra condition $Y_{27,28} = 0$. There is $Y_{27,27} = 1$ in the differential path, so bit 5 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ can not be calculated. From $Y_{28,4} = 0$ and $Y_{29,4} = 0$ in the differential path, we can get that bit 4 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ equals to zero. Similarly, by adding the conditions $Y_{27,25} = 0$ and $Y_{28,3} = 0$, we can get that bit 3 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ equals to zero. Let

$$R_5 = (Y_{25} \lll 10) + m_9 + k_2^r,$$

and force $R_{5,i} = 0$ $(i = 3, 4, 5)$, then there is no carry from bit 3 to 4 when computing R_6, and $R_{6,4} = 0$, $Q_{6,19} = 0$, where

$$R_6 = (Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r, \qquad Q_6 = R_6 \lll 15.$$

Combined with the conditions $Y_{26,i} = 0$ $(i = 9, 10)$ (which have been hold in the differential path), $Y_{30,21}$ can be calculated correctly by

$$(Y_{26} \lll 10) + \Big((Y_{25} \lll 10) + (Y_{28} \wedge (\neg Y_{27} \lll 10) \wedge \texttt{0x78}) + m_9 + k_2^r\Big) \lll 15$$

because the carry can be stopped at bit 20 when computing Y_{30}.

C.3 The Process of Computing $Y_{30,i}$ $(i = 27, 30, 31)$

Y_{29} has no influence on the bits 10,11,13,...,16 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$ by adding the conditions $Y_{27,i} = 0$ $(i = 0, 1, 3, ..., 6)$ (in fact, $Y_{27,i} = 0$ $(i = 0, 1)$ already hold in the conditions of the differential path). Thanks to the condition $Y_{29,12} = 1$ in the differential path, we can compute bit 12 of $F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10))$. Let

$$T_7 = (Y_{25} \lll 10) + ((Y_{28} \wedge \texttt{0x1fc00}) \vee \texttt{0x1000}) + m_9 + k_2^r,$$

and force $T_{7,10} = 0$ and $T_{7,11} = 1$. Let

$$R_7 = (Y_{25} \lll 10) + F_4(Y_{29}, Y_{28}, (Y_{27} \lll 10)) + m_9 + k_2^r, \qquad Q_7 = R_7 \lll 15,$$

thus, $R_{7,i}$ $(i = 11, ..., 16)$ can be obtained correctly since the carry is stopped in bit 10 when computing R_7. Obviously, $Q_{7,i}$ $(i = 26, ..., 31)$ can be obtained correctly and $Q_{7,26} = 1$ hold. Combined with the condition $Y_{26,16} = 1$ (which has been satisfied in the conditions of the differential path), we can deduce that $Y_{30,i}$ $(i = 27, ..., 31)$ can be obtained correctly by

$$(Y_{26} \lll 10) + \Big((Y_{25} \lll 10) + ((Y_{28} \wedge \texttt{0x1fc00}) \vee \texttt{0x1000}) + m_9 + k_2^r\Big) \lll 15$$

since there is always a carry in bit 26 when computing Y_{30}.

C.4 The Process of Computing $Y_{32,20}$

From the step function, we know that

$$Y_{32} = (Y_{28} \lll 10) + \Big((Y_{27} \lll 10) + F_4(Y_{31}, Y_{30}, (Y_{29} \lll 10)) + m_2 + k_2^r\Big) \lll 11.$$

There is $Y_{29,31} = 0$ in the conditions of the differential path, which makes $Y_{31,9}$ has no influence on the output of the 9th bit of $F_4(Y_{31}, Y_{30}, (Y_{29} \lll 10))$. Meanwhile, we can compute $Y_{30,9}$ in the

Table 8: The Conditions of Steps 36-38, 57-64 in Left Branch

Chaining Variable	Conditions on the Chaining Variable
X_{36}	$X_{36,i} = 0$ $(i = 11, 21)$, $X_{36,i} = 1$ $(i = 2, 7, 24)$
X_{37}	$X_{37,i} = 0$ $(i = 2, 21)$, $X_{37,i} = 1$ $(i = 7, 17)$
X_{38}	$X_{38,i} = 0$ $(i = 17, 21)$
X_{57} to X_{59}	$X_{57,26} = 0$, $X_{58,26} = 1$, $X_{59,4} = 0$
X_{61} to X_{63}	$X_{61,4} = 1$, $X_{61,5} = 0$, $X_{61,14} = X_{60,14}$, $X_{63,14} = 1$, $X_{63,15} = 0$

Table 9: The Conditions of Steps 29-33, 60-64 in Right Branch

Chaining Variable	Conditions on the Chaining Variable
Y_{29}	$Y_{29,i} = 0$ $(i = 0, 4, 15, 30, 31)$, $Y_{29,26} = Y_{28,26}$, $Y_{29,i} = 1$ $(i = 10, 11, 12, 14, 17, 20)$
Y_{30}	$Y_{30,i} = 0$ $(i = 21, 27, 30, 31)$, $Y_{30,i} = 1$ $(i = 11, 15)$, $Y_{30,5} = Y_{29,5}$
Y_{31}	$Y_{31,i} = 1$ $(i = 5, 10, 21, 30)$, $Y_{31,20} \neq Y_{30,20}$, $Y_{31,24} = Y_{30,24}$
Y_{32}	$Y_{32,5} = 0$, $Y_{32,i} = 1$ $(i = 10, 21)$
Y_{33}	$Y_{33,21} = 1$, $Y_{33,20} = 1$ or $Y_{32,20} = 0$
Y_{60} to Y_{64}	$Y_{61,4} = Y_{60,26}$, $Y_{62,4} = 1$, $Y_{63,4} = 0$

previous computations. Therefore, the output of bit 9 of $F_4(Y_{31}, Y_{30}, (Y_{29} \lll 10))$ can be obtained. Let

$$T_8 = (Y_{27} \lll 10) + (Y_{30} \wedge \texttt{0x200}) + m_2 + k_2^r,$$

$$R_8 = (Y_{27} \lll 10) + F_4(Y_{31}, Y_{30}, (Y_{29} \lll 10)) + m_2 + k_2^r, \quad Q_8 = R_8 \lll 11,$$

$$R_9 = (Y_{27} \lll 10) + m_2 + k_2^r,$$

and force $R_{9,i} = 0$ $(i = 6, 7, 8)$.

Moreover, we add the conditions that $Y_{28,i} = 0$ $(i = 6, 7, 8, 9)$, then $Y_{32,20}$ can be obtained correctly with a probability of more than 0.9 by

$$(Y_{28} \lll 10) + ((Y_{27} \lll 10) + (Y_{30} \wedge \texttt{0x200}) + m_2 + k_2^r) \lll 11.$$

D The conditions on some chaining variables

Table 10: The Extra Conditions of the Generalized Message Modification in the Left Branch

Chaining Variable	Extra Conditions on the Chaining Variable
X_{33}	$X_{33,i} = 0$ $(i = 6, 8, 9)$
X_{34}	$X_{34,6} = 0$
X_{35}	$X_{35,i} = 0$ $(i = 0, ..., 3, 5, 6, 17, ..., 30)$
X_{37}	$X_{37,i} = 1$ $(i = 0, 1, 3, ..., 12)$
	$T_{1,i} = 0$ $(i = 2, 3, 5)$, $T_{2,17} = 0$, $T_{3,7} = 0$, $S_{1,i} = 0$ $(i = 16, 17)$, $S_{2,16} = 0$

Table 11: The Extra Conditions of the Generalized Message Modification in the Right Branch

Chaining Variable	Extra Conditions on the Chaining Variable
Y_{26}	$Y_{26,i} = 0$ ($i = 29, 30$)
Y_{27}	$Y_{27,i} = 0$ ($i = 3, ..., 6, 13, 14, 22, 25$)
Y_{28}	$Y_{28,i} = 0$ ($i = 3, 6, ..., 9, 23$)
	$T_{4,i} = 0$ ($i = 23, 24$), $T_{5,i} = 0$ ($i = 30, 31$) , $T_{7,i} = 0$ ($i = 9, 10$) $R_{2,i} = 0$ ($i = 23, 24, 25$), $R_{5,i} = 0$ ($i = 3, 4, 5$), $R_{9,i} = 0$ ($i = 6, 7, 8$)