New Records in Collision Attacks on RIPEMD-160 and SHA-256 (Preliminary Version)

Yingxin Li[1], Fukang Liu[2,3], Gaoli Wang[1]

[1] Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China
[2] Tokyo Institute of Technology, Tokyo, Japan
[3] University of Hyogo, Hyogo, Japan
`liyx1140@163.com,liufukangs@gmail.com,glwang@sei.ecnu.edu.cn`

Abstract. RIPEMD-160 and SHA-256 are two hash functions used to generate the bitcoin address. In particular, RIPEMD-160 is an ISO/IEC standard and SHA-256 has been widely used in the world. Due to their complex designs, the progress to find (semi-free-start) collisions for the two hash functions is slow. Recently at EUROCRYPT 2023, Liu et al. presented the first collision attack on 36 steps of RIPEMD-160 and the first MILP-based method to find collision-generating signed differential characteristics. We continue this line of research and implement the MILP-based method with a SAT/SMT-based method. Furthermore, we observe that the collision attack on RIPEMD-160 can be improved to 40 steps with different message differences. We have practically found a colliding message pair for 40-step RIPEMD-160 in 16 hours with 115 threads. Moreover, we also report the first semi-free-start (SFS) colliding message pair for 39-step SHA-256, which can be found in about 3 hours with 120 threads. These results update the best (SFS) collision attacks on RIPEMD-160 and SHA-256. Especially, we have made some progress on SHA-256 since the last update on (SFS) collision attacks on it at EUROCRYPT 2013, where the first practical SFS collision attack on 38-step SHA-256 was found.

Keywords: practical collisions · RIPEMD-160· SHA-256· SAT/SMT

1 Introduction

Before the devastating attacks in 2005 [23,26,24,25] on the MD-SHA hash family, there is a trend to design efficient hash functions with a similar structure to MD4, including MD5, SHA-0, SHA-1, SHA-2, RIPEMD-128 and RIPEMD-160, just to name a few. After 2005, we have witnessed collision attacks on full MD4 [23], MD5 [25], SHA-0 [26,1], and SHA-1 [24,7,18,6] as well as the semi-free-start collision attack on full RIPEMD-128 [5]. Only RIPEMD-160 and SHA-2 survived this game and hence it becomes important to further understand their collision resistance.

In particular, RIPEMD-160 is an ISO/IEC standard and is now used to generate the bitcoin address with SHA-256. As for SHA-256, it has been widely deployed around the world. In this sense, studying their security is of great importance. The difficulty to analyze RIPEMD-160 and SHA-2 seems to exist in their relatively complex designs. For SHA-2, its round function and message expansion are much more complex than that of SHA-1, which makes it difficult to find (correct) collision-generating differential characteristics for a large number of steps [13]. For RIPEMD-160, its special dual-branch structure makes it difficult to perform the message modification on both branches simultaneously and finding differential characteristics is also not easy because its round function is also more complex than that of MD5, SHA-1 and RIPEMD-128, as indicated in [15].

To improve the collision attacks on SHA-2, we have seen enormous efforts to use complex message differences to improve the attacks [13,2,14,3]. However, the tools used to search for the corresponding differential characteristics are not publicly available and few details are known.

For many existing collision attacks on RIPEMD-160, the used message differences are not always that complex. Specifically, for the semi-free-start collision attacks on 42 and 48 steps of RIPEMD-160 starting from a middle step [15,22], the attacker only injects the difference in 1 out of 16 message words. For a series of (semi-free-start) collision attacks on RIPEMD-160 starting from the first step [10,8,9], the difference is still injected in 1 message word. Recently, the collision attack on RIPEMD-160 is improved to 36 steps for the first time [11], where the difference is injected in 3 message words. This seems to indicate that there is potential to further improve the attack by using more complex message differences.

For tools developed for the MD-SHA hash family, only Stevens's tools developed for MD5 and SHA-1 are open-source [17,19,18], but whether they can be useful for RIPEMD-160 and SHA-2 is unclear due to the relatively complex design in the two hash functions. To make finding collision-generating signed differential characteristics easier, Liu et al. have put many efforts to invent a novel MILP-based method [11] and it works quite well for RIPEMD-160. As can be observed in [11], two main techniques are how to describe signed difference propagations through each component of the step function and how to automatically detect contradictions in an efficient way. At the end of [11], the authors left an interesting problem whether it is possible to apply this technique to SHA-2 because contradictions in SHA-2 differential characteristics occur more easily as indicated in [13].

Our contributions. We briefly summarize our contributions as follows:

1. We report the first practical colliding message pair for 40-step RIPEMD-160. This is the first time to practically break half of the total number of steps of RIPEMD-160 since its proposal at FSE 1996[4].

---

[4] We consider (SFS) collision attacks starting from the first step and the distinguishing attacks are not taken into account.

2. We demonstrate for the first time that the technique developed in [11] can be applied to SHA-256 and this obviously gives a positive answer to the question left in [11]. We thus believe that it is meaningful to further study the technique in [11].
3. We report the first practical SFS colliding message pair for 39-step SHA-256 and this updates the record kept by Mendel et al. at EUROCRYPT 2013 [14] after 10 years.

Table 1. Summary of the attack on RIPEMD-160

Attack type	Steps	Time	Memory	References
preimage	34	$2^{158.91}$	\	[21]
Distinguishing	43	2^{151}	\	[20]
Distinguishing	52*	2^{151}	\	[20]
SFS collision	36*	*practical*	*practical*	[12]
SFS collision	42*	$2^{75.5}$	2^{64}	[15]
SFS collision	48*	$2^{76.5}$	2^{64}	[22]
SFS collision	36/37	*practical*	*practical*	[9]
SFS collision	40	$2^{74.6}$	*negligible*	[9]
collision	30/31	*practical*	*practical*	[8]
collision	36	$2^{64.5}$	2^{24}	[11]
collision	40	*practical*	*negligible*	this paper

* An attack starts at an intermediate step.

Table 2. Summary of (SFS) collision attacks on SHA-256

Attack type	Steps	Time	Memory	References
collision	28	*practical*	\	[14]
collision	31	$2^{65.5}$	\	[14]
SFS collision	38	*practical*	\	[14]
SFS collision	39	*practical*	\	this paper

2 The Automatic Method in [11]

The form of the step function of RIPEMD-160 can be described as below:

$$d_{i+5} = (d_{i+1} \lll 10) \boxplus (F(d_{i+4}, d_{i+3}, d_{i+2} \lll 10) \boxplus (d_i \lll 10) \boxplus m \boxplus c_i) \lll s,$$

where $(d_i, \ldots, d_{i+5}, m)$ are all 32-bit variables, c is a 32-bit constant, $s \in [0, 31]$ is an integer and F is a Boolean function.

Denote the signed difference of $(d_i, \ldots, d_{i+5}, m)$ by $(\Delta d_i, \ldots, \Delta d_{i+5}, \Delta m)$. Then, each of $(\Delta d_i, \ldots, \Delta d_{i+5}, \Delta m)$ can be represented with a vector of size 32.

In this sense, it is only required to study the following step function because the rotation ($\lll 10$) only affects the order of variables:

$$a_5 = a_1 \boxplus (F(a_4, a_3, a_2) \boxplus a_0 \boxplus m \boxplus c) \lll s. \tag{1}$$

With some intermediate 32-bit variables $(b_0, \dots, b_5)$, Equation 1 can be further decomposed as:

$$\begin{aligned}
b_0 &= m \boxplus c,\\
b_1 &= F(a_4, a_3, a_2),\\
b_2 &= b_0 \boxplus b_1,\\
b_3 &= b_2 \boxplus a_0,\\
b_4 &= b_3 \lll s,\\
b_5 &= a_1 \boxplus b_4,\\
a_5 &= b_5.
\end{aligned}$$

In [11], the authors described how to model the signed difference transitions through the step function, i.e. how to use constraints to describe the propagation:

$$(\Delta a_0, \dots, \Delta a_4, \Delta m) \to \Delta a_5.$$

In particular, the model can be briefly summarized as follows:

- Model the deterministic signed difference addition $\Delta z = \Delta x \boxplus \Delta y$. Specifically, although we indeed have many possible Δz for a given $(\Delta x, \Delta y)$, we only consider one valid Δz. This is based on the feature of the step function of the MD-SHA hash family.
- Model the signed difference transitions for the Boolean function F, i.e. $(\Delta a_4, \Delta a_3, \Delta a_2) \to \Delta b_1$. This is captured by the so-called *fast filtering model* for F in [11]
- Model the signed difference transitions for $\Delta z = 0 \boxplus \Delta z'$, i.e. this is called *modelling the expansion of the modular difference*. In other words, for a given $\Delta z'$, how to compute all possible Δz such that they correspond to the same modular difference.
- Model the update $a_5 = a_1 \boxplus b_3 \lll s$. The authors [11] introduced two different ways to model it, i.e. *the first strategy* and *the second strategy*, such that the model can handle as many cases as possible.

However, simply using the above models is insufficient because contradictions easily occur, especially in the Boolean function. Hence, they introduced the so-called monitoring variable, which can be used to monitor whether contradictions occur in the Boolean function over different steps. Briefly speaking, by using three additional variables (a_4, a_3, a_2) and constructing another model to only capture the relations between $(\Delta a_4, \Delta a_3, \Delta a_2, \Delta b_1)$ and (a_4, a_3, a_2), it is possible to detect the contradictions in the Boolean function. In [11], if (a_4, a_3, a_2) is involved, it is called the *full model* for F.

Another place where contradictions occur is at the operation

$$a_5 = a_1 \boxplus b_3 \lll s,$$

especially when the conditions on (a_5, a_1) are dense. This is a special operation in RIPEMD-160 and makes it more difficult to find valid signed differential characteristics. Detecting the contradictions in this operation is a bit complex and we omit the details. We emphasize that in our search for valid SHA-256 differential characteristics, we only consider the monitoring technique, i.e. detecting contradictions in the Boolean function.

In [11], all constraints are described with linear inequalities, i.e. the MILP-based method. In this work, we have implemented them with a SAT/SMT-based method, i.e. we will use Conjunctive Normal Form (CNF) to describe the corresponding constraints.

3 New Collision Attacks on RIPEMD-160

We observe that the feasibility of the collision attack on 36-step RIPEMD-160 [11] is mainly due to a well-constructed local collision on left branch of RIPEMD-160. Specifically, by injecting differences in the message words

$$(m_0, m_6, m_9),$$

it is possible to construct a local collision for the first 10 steps and the last 15 steps on the left branch. By carefully analyzing the step function and the message expansion of RIPEMD-160, we find that by injecting differences in the message words

$$(m_0, m_2, m_{11}, m_{12}),$$

we can improve the local collisions on the left branch such that a collision attack on 40-step RIPEMD-160 is possible. With our SAT/SMT-based tool, we have found the corresponding 40-step differential characteristic, as shown in Table 5. To find the conforming message pairs, we mainly use the technique in [16] and the dedicated freedom degree utilization technique in [11]. More details will be given in the full version. As the evidence, we present the first colliding message pair for 40-step RIPEMD-160 in Table 3, which was found in about 16 hours with 115 threads.

4 SFS Collisions for 39-Step SHA-256

To find the SFS collisions for 39-step SHA-256, we are mainly based on the SFS collision attack on 39-step SHA-512 [2]. Specifically, we use the same strategy to inject the message differences as in [2]. In this way, we have found a differential characteristic for 39-step SHA-256, as shown in Table 6. As already mentioned, in the search, we only use the monitoring technique to detect the contradictions caused by the Boolean function over different steps. Although contradictions

Table 3. The colliding message pair for 40 steps of RIPEMD-160 where we use two message blocks (M_0, M_1) to generate a collision

M_0	4b1de304	f52a5a3e	bbd7d814	6454a1d6	a5571007	6c4151f5	8970f768	32c48fd1
	54c428ea	113b00cf	3db1bb85	1d2b2de6	89157118	89157118	d22f990b	6db9f321
M_1	a179ed0	582e9fee	8c68cd3d	d120a6e	de43af57	df2e7a6f	2b40967e	df302947
	ee7f066f	d7b7707d	9f1cc8a9	eaecfcb8	b449f1a	ec058b69	996ee0d2	994ef6b1
M_1'	a159ed0	582e9fee	8c48cd3d	d120a6e	de43af57	df2e7a6f	2b40967e	df302947
	ee7f066f	d7b7707d	9f1cc8a9	eaecfd38	b451f1a	ec058b69	996ee0d2	994ef6b1
hash	a76b7982	e39826f9	52eb6b63	6b48ecdd	4ddca6c5			

more easily occur in SHA-256 as indicated in [13], we found that as long as the differential characteristic on one branch is sparse, by minimizing the hamming weight of the whole differential characteristic, we can expect to obtain a valid differential characteristic. To verify the correctness of this differential characteristic, we use a SAT/SMT-based method to find the conforming message pair, as shown in Table 4 or Table 7. Finding such a message pair takes about 3 hours with 120 threads. More details will be given in the full version.

Table 4. The SFS colliding message pair for 39 steps of SHA-256

CV	02b19d5a	88e1df04	5ea3c7b7	f2f7d1a4	86cb1b1f	c8ee51a5	1b4d0541	651b92e7
M	c61d6de7	755336e8	5e61d618	18036de6	a79f2f1d	f2b44c7b	4c0ef36b	a85d45cf
	f72b8c2f	0def947c	a0eab159	8021370c	4b0d8011	7aad07f6	33cd6902	3bad5d64
M'	c61d6de7	755336e8	5e61d618	18036de6	a79f2f1d	f2b44c7b	4c0ef36b	a85d45cf
	e72b8c2f	0fcf907c	b0eab159	81a1bfc1	4b098611	7aad07f6	33cd6902	3bad5d64
hash	431cadcd	ce6893bb	d6c9689a	334854e8	3baae1ab	038a195a	ccf54a19	1c40606d

Remark 1. It is interesting to notice that although the authors of [2] reported the first SFS collision attack on 39-step SHA-512 by improving the automatic tools in [14,3], nothing has been reported for 39-step SHA-256 and the largest number of attacked steps still remains 38 in [14]. A reasonable guess may be that it is infeasible for the dedicated tools developed for SHA-2 in [13,2,14,3] to find a valid differential characteristic for 39-step SHA-256. We have to emphasize that SHA-512 is different from SHA-256 and a SFS collision attack on 39-step SHA-512 does not imply a SFS collision attack on 39-step SHA-256. This seems to indicate the our SAT/SMT-based method can somehow beat the dedicated tools [2]. Anyway, we give a positive answer to the problem left in [11] by applying the technique to the SHA-2 family. In particular, the new attack on SHA-256 demonstrates the effectiveness of the technique developed by Liu et al. in [11] and we believe it is worth further investigations.

5 Conclusion

By continuing the line of research in [11], we present the first practical collision attack on 40-step RIPEMD-160 and SFS collision attack on 39-step SHA-256. These results update the best cryptanalysis records in (SFS) collision attacks on RIPEMD-160 and SHA-256. Especially, the results for RIPEMD-160 can be viewed as major progress since its proposal in FSE 1996. Moreover, with the results for SHA-256, we demonstrate for the first time that the technique in [11] can also be efficiently applied to SHA-256 and it may even outperform the dedicated tools.

In particular, similar to the quantum collision attacks on 38-step SHA-256 and 39-step SHA-512 by converting SFS collisions into collisions in the quantum setting [4], based on our new attack on 39-step SHA-256, one may expect a valid quantum collision attack on 39-step SHA-256 with the same technique. However, there are indeed different perspectives to interpret the quantum collision attack in [4] and the actual advantage in the quantum setting may be too small. Anyway, our new attack on 39-step SHA-256 updates the best attacks on 38-step SHA-256 in both the classical and quantum settings.

References

1. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, and W. Jalby. Collisions of SHA-0 and reduced SHA-1. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 36–57. Springer, 2005.
2. C. Dobraunig, M. Eichlseder, and F. Mendel. Analysis of SHA-512/224 and SHA-512/256. In T. Iwata and J. H. Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 612–630. Springer, 2015.
3. M. Eichlseder, F. Mendel, and M. Schläffer. Branching heuristics in differential collision search with applications to SHA-512. In C. Cid and C. Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 473–488. Springer, 2014.
4. A. Hosoyamada and Y. Sasaki. Quantum Collision Attacks on Reduced SHA-256 and SHA-512. In T. Malkin and C. Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 616–646. Springer, 2021.
5. F. Landelle and T. Peyrin. Cryptanalysis of full RIPEMD-128. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 228–244. Springer, 2013.

Table 5. The differential characteristic for 40 steps of RIPEMD-160, where $\delta m_0 = 0 \boxminus 2^{17}, \delta m_2 = 0 \boxminus 2^{21}, \delta m_{11} = 2^7, \delta m_{12} = 2^{15}$

i	ΔX_i	π_i^l	i	ΔY_i	π_i^r
-5	================================		-5	================================	
-4	================================		-4	================================	
-3	================================		-3	================================	
-2	================================		-2	================================	
-1	================================		-1	================================	
0	unnn============================	0	0	================================	5
1	===============nuuuu=n==========	1	1	================================	14
2	u=uun=u==========n==u====un=nnnn	2	2	=============0==================	7
3	=====nnn===unun==u==u====nn=====	3	3	0==u============================	0
4	u=u==uu==u==================n=nu	4	4	0====1===========0==1=n==1===010	9
5	============nuuu=n======u=n=====	5	5	101====u==0=00=1===0000=100u0000	2
6	==u====nuu======================	6	6	0110=1===nnuu1nuuuuuuuuuu10100=0	11
7	================unnnnnnnnnn=====	7	7	1unnnnn11000unn00unn10nunn11=110	4
8	================================	8	8	=1011nu001nu111nuu=unnn0101nuuuu	13
9	================================	9	9	00u==nu00u010===1000101u=0101n0=	6
10	================================	10	10	111====0==u=n10=0u01=1n01=010==1	15
11	================================	11	11	0=0=n1=0=10n0===u====n1=1===0===	8
12	================================	12	12	11u===10=0=1u=0======1==0=1u===0	1
13	================================	13	13	==0======1==0=n=10=0====1=10===n	10
14	================================	14	14	==1====0========u1=1===========u	3
15	================================	15	15	=======1=n==============n=======	12
16	================================	7	16	=============================u==	6
17	================================	4	17	================================	11
18	================================	13	18	================================	3
19	================================	1	19	=====================0==========	7
20	================================	10	20	=====================1==========	0
21	================================	6	21	===========u====================	13
22	================================	15	22	================================	5
23	================================	3	23	===========1====================	10
24	=========n======================	12	24	===========1=============010000=	14
25	==u======0======================	0	25	=u=======================111111=	15
26	==0============================1	9	26	===============nuuuuu===========	8
27	========================1=======	5	27	======1=========================	12
28	================================	2	28	======0=========================	4
29	================================	14	29	================================	9
30	================================	11	30	================================	1
31	================================	8	31	================================	2
32	================================	3	32	================================	15
33	================================	10	33	================================	5
34	================================	14	34	================================	1
35	================================	4	35	================================	3
36	================================	9	36	================================	7
37	================================	15	37	================================	14
38	================================	8	38	================================	6
39	================================	1	39	================================	9

$Y_{15}[10] = Y_{14}[10], Y_{15}[27] = Y_{14}[27], Y_{16}[10] = Y_{15}[10], Y_{16}[25] = Y_{15}[25]$
$Y_{17}[0] = Y_{16}[0], Y_{17}[17] = Y_{16}[17], Y_{18}[12] = Y_{17}[12], Y_{23}[30] = Y_{22}[30],$
$Y_{27}[8] = Y_{26}[8], Y_{28}[i] = Y_{27}[i] (i \in \{21, 22, 23, 24, 26\})$
$X_{23}[22] = X_{22}[12], X_{24}[29] = X_{23}[19]$

Table 6. The differential characteristic for 39 steps of SHA-256

i	ΔA_i	ΔE_i	ΔW_i
-4	================================	================================	
-3	================================	================================	
-2	================================	================================	
-1	================================	================================	
0	================================	================================	================================
1	================================	================================	================================
2	================================	================================	================================
3	================================	================================	================================
4	================================	================================	================================
5	================================	================================	================================
6	================================	===0============================	================================
7	================================	===1=========1======11======0===	================================
8	===u============================	unnn1=1110=0=0101==000=====1110=	===u============================
9	==============n=u====u======n===	010n0n0111010nu01001un011n10n=10	======n===u==========u==========
10	================================	0101u1n=1n0n010=u0=11nuu=1u00=n1	===n============================
11	================================	=100010==0=0101=0===0010=10=1=0=	=======nn=======n===n===nn==uu=n
12	================================	=unn010000=1000011=00011==0=101=	=============u=======nn=========
13	================================	10110nuuuuuuuuu0u101un000010n111	================================
14	================================	=111=0000000000=0=1=001111111=1=	================================
15	============================n===	11001101101000000001nuuuuuuuu001	================================
16	=======u=u=======u==============	010100unu000001001u1000110unn=n1	======n===u==========u==========
17	================================	11=0111u00nn=100110=u1u00unn000n	===n============================
18	===n============================	uuu1uuuu01000=110n000111101=0101	================================
19	================================	000u0n1000101=0un01=1100=u11n000	================================
20	================================	011100un0u001unnnn11000000001111	================================
21	================================	=110=111=0===000=1=======1==1===	================================
22	================================	=nuu==0110===00101=0110=====110=	================================
23	================================	=000============================	================================
24	================================	=111============================	=======n=n=======n==============
25	================================	================================	================================
26	================================	================================	===u============================
27	================================	================================	================================
28	================================	================================	================================
29	================================	================================	================================
30	================================	================================	================================
31	================================	================================	================================
32	================================	================================	================================
33	================================	================================	================================
34	================================	================================	================================
35	================================	================================	================================
36	================================	================================	================================
37	================================	================================	================================
38	================================	================================	================================

Table 7. The solution to the differential characteristic for 39 steps of SHA-256

i	ΔA_i	ΔE_i	ΔW_i
-4	11110010111101111101000110100100	01100101000110111001001011100111	
-3	01011110101000111100011110110111	00011011010011010000010101000001	
-2	10001000111000011101111100000100	11001000111011100101000110100101	
-1	00000010101100011001110101011010	10000110110010110001101100011111	
0	00110110101000101011101001011101	01110010111011111001001000011011	11000110000111010110110111100111
1	11111110010110011101000100110000	01000100101100111000101000110011	01110101010100110011011011101000
2	01111010110000101101011101111001	11111011100111011101011110101010	01011110011000011101011000011000
3	11000110101111000000100100001001 0	01011100100001000101010111010000	00011000000000110110110111100110
4	00110110001110110001101011010111	10011100100100111000011110000110	10100111100111110010111100011101
5	00100010010111011001101101101111	10110010101000010010111010010111	11110010101101000100110001111011
6	00011110110111100010001111011010	00001101111000000000110000011110	01001100000011101111001101101011
7	01101000111100101101111111001010	00010111011001111011111110000001	10101000010111010100010111001111
8	001u0011001101101111111001110110	unnn1111100000101010001110111100	111u0111001010111000110000101111
9	111111101111100n1u1111u001111n011	010n0n0111010nu01001un011n10n010	000011n111u0111110010u0001111100
10	10011001011010101101111001000101	0101u1n11n0n0101u0111nuu11u001n1	101n0000111010101011000101011001
11	11010101100110101100110000010111	01000100000010110100001011001000	1000000nn0100001n011n111nn00uu0n
12	11101101101111010000000100100000 0	0unn0100000100001110001111011010	0100101100001u0110000nn000010001
13	01110000000101000110110000100011 1	10110nuuuuuuuuuu0u101un000010n111	01111010101011010000011111110110
14	10001001100111111111110001001100 0	01110000000000001011000111111111	00110011110011010110100100000010
15	0111101100000100111101010010n001	11001101101000000001nuuuuuuuu001	00111011101011010101110101100100
16	1100000u1u1100001u00011111100101	010100unu000001001u1000110unn0n1	000000n110u1010110001u0100001100
17	10110100101111101010010100010010	1100111u00nn01001101u1u00unn000n	011n1010010001010100100100101100
18	111n0110100001100011101010100001	uuu1uuuu010000110n00011110110101	01101001111101001110100000011010
19	00110000000111110001110001110011	000u0n100010100un01011001u11n000	01010101001101011100100110001001
20	10111000111010010010111110000001	011100un0u001unnnn11000000001111	00000110000011010101100111001011
21	00100010000000100110010111010111	01101111101010000100000111111110	11101000000100101100100110101001 1
22	01101010110011010011011000011101	0nuu0001100110010110110011111101	01101010110000101010100110000101
23	01100110000110111111001101001100	10001100111100010111100111100101	11001000000011010010111010100100
24	00001110101010011011010100101001	01111001010111001101111110111001	1000000n0n0100101n10110011001010
25	01001111000011011101101010111110	10110110010011111100110101001011	10110101101001000110011001010101
26	10111001000011100010010010111011	01100100011101010110111110101111	001u1110110000111101010000001001
27	10010000011011101100011111110101	11110100011100000101011011001011	10010001111011111001000000010111
28	01010101000101001010000101110101	11101010000001101010101100010101	01100110000110101000000000001010
29	10111110000011110000111110001010 0	01101101101100101111111000110001	01110111111110111010110110101010
30	00100110101110000011000100110111	10011011110010011001100010111000	10100100010101001001000100010001
31	00110011011101001100010110111011	11101110111100110111011101100101	11011010101101000111000011101101
32	10100010010000000110101001010101	10001111000001101011000010001110	10011001101101010010000000001011
33	10100100101001101101101100111001	00000010110000000001110101110111	01100011010100110000011011110011
34	00100010100000101011001100001110	01111011010011011010001110100010	10011010011101011101001011000100
35	01000000010100001000001101000100	10110111001001001100110110000110	01011111110011101100001010011110
36	01111000000100101101000001110001 1	10110001101010000100010011011000	10101111001001010100110110111010
37	01000101100001101011010010110111	00111010100110111100011110110101	11110011011100011100011011100101
38	01000000011010110001000000111001 1	10110100110111111100011010001100	01101111111100010111001100100001

6. G. Leurent and T. Peyrin. From collisions to chosen-prefix collisions application to full SHA-1. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 527–555. Springer, 2019.
7. G. Leurent and T. Peyrin. SHA-1 is a shambles: First chosen-prefix collision on SHA-1 and application to the PGP web of trust. In S. Capkun and F. Roesner, editors, *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 1839–1856. USENIX Association, 2020.
8. F. Liu, C. Dobraunig, F. Mendel, T. Isobe, G. Wang, and Z. Cao. Efficient collision attack frameworks for RIPEMD-160. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 117–149. Springer, 2019.
9. F. Liu, C. Dobraunig, F. Mendel, T. Isobe, G. Wang, and Z. Cao. New semi-free-start collision attack framework for reduced RIPEMD-160. *IACR Trans. Symmetric Cryptol.*, 2019(3):169–192, 2019.
10. F. Liu, F. Mendel, and G. Wang. Collisions and semi-free-start collisions for round-reduced RIPEMD-160. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 158–186. Springer, 2017.
11. F. Liu, G. Wang, S. Sarkar, R. Anand, W. Meier, Y. Li, and T. Isobe. Analysis of RIPEMD-160: New Collision Attacks and Finding Characteristics with MILP. 2023. To appear at EUROCRYPT 2023.
12. F. Mendel, T. Nad, S. Scherz, and M. Schläffer. Differential attacks on reduced RIPEMD-160. In D. Gollmann and F. C. Freiling, editors, *Information Security - 15th International Conference, ISC 2012, Passau, Germany, September 19-21, 2012. Proceedings*, volume 7483 of *Lecture Notes in Computer Science*, pages 23–38. Springer, 2012.
13. F. Mendel, T. Nad, and M. Schläffer. Finding SHA-2 characteristics: Searching through a minefield of contradictions. In D. H. Lee and X. Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 288–307. Springer, 2011.
14. F. Mendel, T. Nad, and M. Schläffer. Improving local collisions: New attacks on reduced SHA-256. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 262–278. Springer, 2013.
15. F. Mendel, T. Peyrin, M. Schläffer, L. Wang, and S. Wu. Improved cryptanalysis of reduced RIPEMD-160. In K. Sako and P. Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013,*

Proceedings, Part II, volume 8270 of *Lecture Notes in Computer Science*, pages 484–503. Springer, 2013.

16. I. Mironov and L. Zhang. Applications of SAT solvers to cryptanalysis of hash functions. In A. Biere and C. P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 102–115. Springer, 2006.
17. M. Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In T. Johansson and P. Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 245–261. Springer, 2013.
18. M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The first collision for full SHA-1. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596. Springer, 2017.
19. M. Stevens, A. K. Lenstra, and B. de Weger. Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In M. Naor, editor, *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.
20. G. Wang, F. Liu, B. Cui, F. Mendel, and C. Dobraunig. Improved (semi-free-start/near-) collision and distinguishing attacks on round-reduced RIPEMD-160. *Des. Codes Cryptogr.*, 88(5):887–930, 2020.
21. G. Wang and Y. Shen. (pseudo-) preimage attacks on step-reduced HAS-160 and RIPEMD-160. In S. S. M. Chow, J. Camenisch, L. C. K. Hui, and S. Yiu, editors, *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, volume 8783 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2014.
22. G. Wang, Y. Shen, and F. Liu. Cryptanalysis of 48-step RIPEMD-160. *IACR Trans. Symmetric Cryptol.*, 2017(2):177–202, 2017.
23. X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.
24. X. Wang, Y. L. Yin, and H. Yu. Finding collisions in the full SHA-1. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.
25. X. Wang and H. Yu. How to break MD5 and other hash functions. In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

26. X. Wang, H. Yu, and Y. L. Yin. Efficient collision search attacks on SHA-0. In V. Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.