Colliding Message Pair for 53-Step HAS-160*

Florian Mendel

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

Florian.Mendel@iaik.TUGraz.at

Abstract. We present a collision attack on the hash function HAS-160 reduced to 53-steps. The attack has a complexity of about 2^{35} hash computations. The attack is based on the work of Cho *et al.* presented at ICISC 2006. In this article, we improve their attack complexity by a factor of about 2^{20} using a slightly different strategy for message modification in the first 20 steps of the hash function.

Keywords: cryptanalysis, collision attack, hash functions

1 Introduction

At ICISC 2006, Cho *et al.* presented a collision attack for HAS-160 reduced to 53 steps. Their attack has a complexity of about 2^{55} 53-step HAS-160 computations, which is barely feasible on an ordinary PC in practice. In this article, we show how to improve their attack by using a slightly different message modification technique to fulfill the conditions on the state variables in the first 20 steps of the hash function. With our method, we find a colliding message pair with a complexity of about 2^{35} 53-step HAS-160 computations. This improves the attack complexity of the original attack of Cho *et al.* by a factor of 2^{20}, which makes the attack feasible in practice.

The remainder of this article is structured as follows. A description of the hash function is given in Section 2. The collision attack of Cho *et al.* is described in Section 3. In Section 4, we describe the new improved collision attack. A sample colliding message pair is given in Section 5. Finally, conclusions are presented in Section 6.

2 Description of the HAS-160

HAS-160 is an iterative hash function that processes 512-bit input message blocks and produces a 160-bit hash value. The design of HAS-160 is similar to the design principles of MD5 and SHA-1. In the following, we briefly describe the hash function. It basically consists of two parts: message expansion and state update transformation. A detailed description of the HAS-160 hash function is given in [1].

* This work is in part supported by the Austrian Science Fund (FWF), project P18138.

Message Expansion. The message expansion of HAS-160 is a permutation of 20 expanded message words w_i in each round. The 20 expanded message words w_i used in each round are constructed from the 16 input message words m_i as follows.

	Round 1	Round 2	Round 3	Round 4
w_0	m_0	m_0	m_0	m_0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
w_{15}	m_{15}	m_{15}	m_{15}	m_{15}
w_{16}	$w_0 \oplus w_1 \oplus w_2 \oplus w_3$	$w_3 \oplus w_6 \oplus w_9 \oplus w_{12}$	$w_{12} \oplus w_5 \oplus w_{14} \oplus w_7$	$w_7 \oplus w_2 \oplus w_{13} \oplus w_8$
w_{17}	$w_4 \oplus w_5 \oplus w_6 \oplus w_7$	$w_{15} \oplus w_2 \oplus w_5 \oplus w_8$	$w_0 \oplus w_9 \oplus w_2 \oplus w_{11}$	$w_3 \oplus w_{14} \oplus w_9 \oplus w_4$
w_{18}	$w_8 \oplus w_9 \oplus w_{10} \oplus w_{11}$	$w_{11} \oplus w_{14} \oplus w_1 \oplus w_4$	$w_4 \oplus w_{13} \oplus w_6 \oplus w_{15}$	$w_{15} \oplus w_{10} \oplus w_5 \oplus w_0$
w_{19}	$w_{12} \oplus w_{13} \oplus w_{14} \oplus w_{15}$	$w_7 \oplus w_{10} \oplus w_{13} \oplus w_0$	$w_8 \oplus w_1 \oplus w_{10} \oplus w_3$	$w_{11} \oplus w_6 \oplus w_1 \oplus w_{12}$

For the ordering of the expanded message words w_i the following permutation is used:

step i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Round 1	18	0	1	2	3	19	4	5	6	7	16	8	9	10	11	17	12	13	14	15
Round 2	18	3	6	9	12	19	15	2	5	8	16	11	14	1	4	17	7	10	13	0
Round 3	18	12	5	14	7	19	0	9	2	11	16	4	13	6	15	17	8	1	10	3
Round 4	18	7	2	13	8	19	3	14	9	4	16	15	10	5	0	17	11	6	1	12

State Update Transformation. The state update transformation of HAS-160 hash function starts from a (fixed) initial value IV of five 32-bit registers and updates them in 4 rounds of 20 steps each. Figure 1 shows one step of the state update transformation of the hash function.

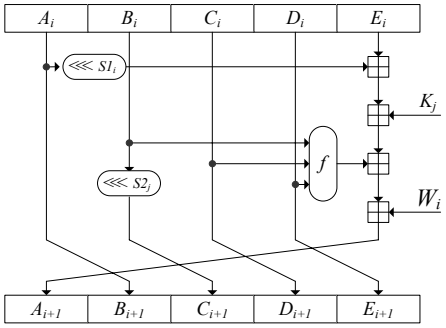

Fig. 1. The step function of HAS-160.

The function f is different in each round: f_0 is used in the first round, f_1 is used in round 2 and round 4, and f_2 is used in round 3.

$$f_0(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$f_1(x, y, z) = x \oplus y \oplus z$$
$$f_2(x, y, z) = (x \vee \neg z) \oplus y$$

A step constant K_j is added in every step; the constant is different for each round. For the actual values of the constants we refer to [1]. While rotation value $s_2 \in \{10, 17, 25, 30\}$ is different in each round of the hash function, the rotation value s_1 is different in each step of a round. The rotation value s_1 for each step of a round is given below.

step i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
s_1	5	11	7	15	6	13	8	14	7	12	9	11	8	15	6	12	9	14	5	13

After the last step of the state update transformation, the initial value and the output values of the last step are combined, resulting in the final value of one iteration known as Davies-Meyer hash construction (feed forward). In detail, the feed forward is a word-wise modular addition of the IV and the output of the state update transformation. The result is the final hash value or the initial value for the next message block.

3 The Attack on 53-step HAS-160

In this section, we will briefly describe the attack of Cho *et al.* on 53-step HAS-160. A detailed description of the attack is given in [2]. For a good understanding of our results it is recommended to study it carefully.

The attack is based on recent results in cryptanalysis of hash functions [3,4]. It can be basically described as follows.

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

3.1 Characteristic for 53-step HAS-160

Finding a *good* characteristic is the most difficult part of the attack. In Table 1, the characteristic which is used by Cho *et al.* in the collision attack on 53-step

HAS-160 is given. Note that we use the same differential path for our improved collision-attack in Section 4.

Signed bit differences [3] are used to describe the differential path for 53-step HAS-160. To improve readability, only the differences in the expanded message words and state variable A for each step are given. Note that the differences of the state variables B, C, D, E are defined by the differences in state variable A.

Table 1. Characteristic for 53-step HAS-160 (*cf.* [2]).

step	ΔA	Δw
1	-32	32
2	$11, -12$	.
3	$18, \ldots, 21, -22$	.
4	$1, \ldots, 16, -17$	.
5	$7, 8, -9, 18, -19, 32$	32
6	$3, -4, 17, -20, -21, 22$	32
7	$-14, \ldots, -17, 18, 22, 29$	.
8	$4, -10, 11$	.
9	$13, -15, 16, 18, -19, 30, 31, -32$	32
10	$-10, -11, 12, -17, -24, 25$	.
11	$-1, 2, -13, \ldots, -15, 16, 28, -32$	32
12	$-8, 9, -17, -21, 22, 26$	32
13	$8, 10, 11, -12, 26, 27, -28$	.
14	$-10, 17, 18, -19, 28, \ldots, 30, -31$	.
15	$11, \ldots, 14, -15, -16, 20, -23, 26, -27$	.
16	$3, 5, 6, -7, -11, -12, 13, -20, 21, -22, -31, 32$	32
17	$-11, 18, -20, -26, 27$	.
18	$1, 5, 18, 20, 22, 27$	.
19	$4, 13, 30$	.
20	$-4, 11$	32
21	11	.
22	-30	32
23	.	32
24	15	.
25	-15	.
26	.	.
27	.	32
28	.	.
29	.	.
30	.	32
31	.	.
$\vdots$	$\vdots$	$\vdots$
53	.	.

3.2 Set of Sufficient Conditions

In order to guarantee that the message follows the characteristic given in Table 1, a set of conditions on the state variables have to be fulfilled. In Table 2, the set of sufficient conditions for the first 25 steps of the hash function is given.

Table 2. A set of sufficient conditions on A_i for the differential path given in Table 1, where '**a**' denotes a condition $A_{i,j} = A_{i-1,j}$, '**b**' denotes a condition $A_{i,j} \neq A_{i-1,j}$, '**c**' denotes a condition $A_{i,j} \neq A_{i-1,j+7}$, '**d**' denotes a condition $A_{i,j} = A_{i-1,j-10}$, '**e**' denotes a condition $A_{i,j} = A_{i-1,j-17}$, and '**f**' denotes a condition $A_{i,j} \neq A_{i-1,j-17}$.

state variable	condition on bits 32-25	24-17	16-9	8-1	#conditions
A_1	1-------	--------	----110-	1-----aa	7
A_2	0100---1	--------	----100a	01---1--	12
A_3	1100aaa0	aa10000-	------10	10aaa0aa	25
A_4	11000-11	--10---1	00000000	00000000	26
A_5	01110111	00110100	00100111	00-11---	28
A_6	0--00111	110111-0	0000001-	00a010-0	27
A_7	1010101-	--0-1001	1111--10	-------1	19
A_8	100-0000	1-1a0---	-111-011	a0aa0a11	25
A_9	100-0101	10-0010-	0110----	-11-0-00	22
A_{10}	--000-10	10a01-01	1-1-0110	000010--	24
A_{11}	1a100010	-1001-10	01110001	01---001	27
A_{12}	1-1--100	10011111	1---0000	1-0--110	23
A_{13}	00--1001	11-00000	1---1001	0a110-11	25
A_{14}	010001-0	--101100	0--a101-	--111a00	24
A_{15}	---11101	-1000-10	11000001	01100a00	27
A_{16}	01010111	00101--0	1--01110	010010-1	27
A_{17}	111--011	01011-00	00-10110	0--10--1	24
A_{18}	01--00--	-00-0-00	11-1--11	---00b-0	18
A_{19}	--0-----	---d-c--	-0-0-1--	----0---	7
A_{20}	----0-b-	---b----	-----0--	---f1---	6
A_{21}	--f-0---	--------	-f-a-0--	--------	5
A_{22}	--1-----	---e----	--------	--------	2
A_{23}	----f---	--------	-e------	--------	2
A_{24}	--b-----	--------	-0------	--------	2
A_{25}	--------	--------	-1------	--------	1

3.3 Finding a Colliding Message Pair

In order to find a colliding message pair, we have to find a message, that fulfills all the conditions given in Table 2. In total there are 434 conditions on the

state variables. Therefore, the probability that a random message follows the characteristic can be estimated by 2^{-434}.

However, the probability can be improved by using message modification techniques. The main idea of message modification is to use the degrees of freedom one has in the choice of the message words to fulfill conditions on the state variables. This improves the probability of the characteristic. It is clear that the number of conditions that can not be fulfilled by message modification techniques determine the final attack complexity.

In [2], Cho *et al.* describe an algorithm for finding a colliding message pair for 53-step HAS-160. The algorithm has a complexity of about 2^{55} 53-step HAS-160 computations. It can be summarized as follows:

1. Use basic message modification techniques to fulfill all conditions on state variables $A_1, \ldots, A_{10}$. This determines the message words m_0, m_1, m_2, m_3, m_4, m_5 ,m_6, m_7 and the values for $m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$ and $m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$. This adds only small additional cost to the attack complexity. Only 10 steps of HAS-160 have to be computed to determine all these message words.
2. At step 11, the dependent expanded message word $w_{16} = m_0 \oplus m_1 \oplus m_2 \oplus m_3$ is already fixed by the first step of the attack. Since there are 27 conditions on A_{11} in step 11, we may fulfill them with a probability of 2^{-27}. Hence, we have to repeat step 1 of the attack about 2^{27} times to find message words satisfying all conditions in steps 1-11.
   Finishing this step of the attack has a complexity of about $2^{27} \cdot 11$ step computations of HAS-160.
3. Use again basic message modification techniques to fulfill all conditions on A_{12}, A_{13}, A_{14}. This determines the message words m_8, m_9, m_{10}. Since these message words can be chosen freely, this adds only small additional cost (3 step computations of HAS-160) to the attack complexity.
4. At step 15 and 16, the dependent words m_{11} and $m_4 \oplus m_5 \oplus m_6 \oplus m_7$ are already fixed by the previous steps of the attack. Since there are $27+27 = 54$ conditions on A_{15} and A_{16} we may fulfill these conditions with a probability of 2^{-54}. Since there are about 2^{24} possible choices for m_8, m_9, m_{10} in the third step of the attack (see Table 2), step 3 of the attack can only be repeated 2^{24} times. Hence, the whole attack has to be repeated about 2^{30} times to find message words following the characteristic in steps 1-16.
   Finishing this step of the attack has a complexity of about $2^{30} \cdot (2^{27} \cdot 11 + 2^{24} \cdot 5)$ step computations of HAS-160. This is approximately about 2^{55} 53-step HAS-160 computations.
5. To fulfill the conditions on A_{17}, A_{18}, A_{19} basic message modification techniques are used again. This determines the message words m_{12}, m_{13}, m_{14}. Since these values can be chosen freely, this adds only small additional cost (3 step computations of HAS-160) to the attack complexity.
6. After step 19, all the message words has been determined. Since there are still 18 conditions on $A_{20}, \ldots, A_{25}$ (see Table 2), we may satisfy the with probability of 2^{-18}. Therefore, step 5 of the attack has to be repeated about 2^{18} times to fulfill all the conditions in steps 20-25. This adds negligible cost to the final attack complexity.

With this method a collision can be found in 53-step HAS-160 with a complexity of about 2^{55} 53-step HAS-160 computations. For a detail of the description of the attack we refer to [2].

4 Improved Collision Attack

In this section, we show how the attack complexity can be reduced to 2^{35}. In the attack, we use the differential path of Cho *et al.* given in Section 3. To improve the attack complexity, we use a slightly modified strategy for message modification in the first 16 steps of the hash function. The main idea of our new method is to reduce the complexity of the collision search algorithm in steps 1-16. Therefore, we use the fact that an additional (first) message block can be used to generate an arbitrary IV (for the second block). This additional degree of freedom we use to reduce the complexity of the attack. The attack can be summarized as follows.

1. Choose arbitrary values for A_2, A_3, A_4, A_5, A_6 satisfying all conditions.
2. Apply message modification techniques to steps 7-15. This determines the message words $m_4, m_5, m_6, m_7, m_8, m_9, m_{10}, m_{11}$ and the value for $m_0 \oplus m_1 \oplus m_2 \oplus m_3$. At step 16, the dependent words m_4, m_5, m_6, m_7 are already used. Since there are 27 conditions at that step (see Table 2), we may satisfy all the conditions from step 7 up to step 16 with a probability of 2^{-27}. Hence, we have to compute about $2^{27} \cdot 10$ steps of HAS-160 to find message words that follow the characteristic from step 7 to 16.
3. Repeat step 2 of the attack about 2^7 times to get about 2^7 different values for $m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$ and save them in a list L. We will need these values in the next step of the attack.
   Finishing this step of the attack has a complexity of about $2^7 \cdot 2^{27} \cdot 10$ step computations of HAS-160. This is equivalent to about $2^{31.6}$ 53-step HAS-160 computations.
4. Use an arbitrary (first) message block to get a suitable IV (for the second block). Note that we have to calculate on average 2 IVs, since 1 condition on the IV has to be fulfilled to guarantee that the characteristic holds in the following steps.
   Calculate A_1 (for all values in L) and check if the 7 conditions on A_1 are satisfied. Since there are 7 conditions on A_1, we always expect to meet the conditions after trying all 2^7 values in the list L.
   Once we have determined A_1, this also determines m_0, m_1, m_2, m_3 and $m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$. Since $m_0 \oplus m_1 \oplus m_2 \oplus m_3$ has already been fixed in the second step of the attack, this step of the attack succeeds with probability 2^{-32}.
   Hence, we have to repeat this step of the attack about 2^{32} times to find message words following the characteristic in steps 1-16.
   Finishing this step of the attack has a complexity of about $2^{32} \cdot (2^7 + 5 + 53 \cdot 2)$ step computations of HAS-160. This is approximately $2^{34.2}$ 53-step HAS-160 computations.

5. Do steps 17 to 25 as described in the original attack (see Section 3). The complexity of these steps can be neglected for the final attack complexity.

Hence, we can find a colliding message pair for 53-step HAS-160 with a complexity of about $2^{31.6} + 2^{34.2} \approx 2^{35}$ 53-step HAS-160 computations. Note that the complexity of the attack can be slightly improved by increasing the size of the list L. A colliding message pair for 53-step HAS-160 is given in the next section.

5 A Colliding Message for 53-step HAS-160

Applying our improved collision attack, we can construct a collision with a complexity of about 2^{35} 53-step HAS-160 computations. The colliding message pair is given in Table 3. Note that h_0 is the initial value, h_1 is the intermediate hash value after the first block, and h_2 is the final hash value after the second block.

Table 3. A colliding message pair for HAS-160.

h_0	67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0
M_0	34338ECF ED111A03 EB2EE891 763594E3 96080160 4558A929 EC731044 B7BADD0B BC637C76 B21FA220 47493D4D B2AEAB79 A68354CF 5833D227 46DE18D7 F9FF5F3B
h_1	40D4B34F F1185C20 ADE02611 9B666A7E 34769338
M_1	4E8F4717 D8E79F84 89D8FE81 04B34CA7 01EA3C40 A364A502 059F6AB9 22774031 9F3E80CE D647A926 1F61242A A1E224AB 901A5AEE 1BCEEEB1 EDEAA891 31BDFF9A
M_1'	4E8F4717 D8E79F84 89D8FE81 84B34CA7 01EA3C40 A364A502 859F6AB9 22774031 1F3E80CE D647A926 1F61242A A1E224AB 901A5AEE 1BCEEEB1 EDEAA891 B1BDFF9A
ΔM_1	00000000 00000000 00000000 80000000 00000000 00000000 80000000 00000000 80000000 00000000 00000000 00000000 00000000 00000000 00000000 80000000
h_2	96D30020 DA815BDF DF265AB5 819CDE2E 5B887F3E
h_2'	96D30020 DA815BDF DF265AB5 819CDE2E 5B887F3E

6 Conclusion

In this article, we presented an improved collision attack on 53-step HAS-160 and an actual colliding message pair. Our new improved attack has a complexity of about 2^{35} 53-step HAS-160 computations. In the attack, we used a slightly modified strategy to do message modification in HAS-160. With this method, we can improve the previous attack by a factor of 2^{20}, which makes the attack feasible in practice.

However, it still remains an topic of further research if the attack of Cho *et al.* on 53-step HAS-160 can be extended to the full HAS-160 hash function. This is work in progress.

Acknowledgment

The author would like to thank Aaram Yun for fruitful discussions on this article.

References

1. Telecommunications Technology Association. *Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160)*, TTAS.KO-12.0011/R1, December 2000.
2. Hong-Su Cho, Sangwoo Park, Soo Hak Sung, and Aaram Yun. Collision Search Attack for 53-Step HAS-160. Accepted for *Information Security and Cryptology - ICISC 2006, 9th International Conference, Busan, Korea, November 30 - December 1, 2006*, to appear in *LNCS* 4296. Springer 2006.
3. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald 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 *LNCS*, pages 19–35. Springer, 2005.
4. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.