Open access • Book Chapter • DOI:10.1007/978-3-642-31912-9_3

Cryptanalysis of round-reduced HAS-160 — Source link

Florian Mendel, Tomislav Nad, Martin Schläffer

Institutions: Graz University of Technology

Published on: 30 Nov 2011 - International Conference on Information Security and Cryptology

Topics: Collision resistance, Collision attack, Hash function, Security of cryptographic hash functions and SHA-2

Cryptanalysis of Round-Reduced HAS-160

Florian Mendel, Tomislav Nad, and Martin Schläffer

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria.
Tomislav.Nad@iaik.tugraz.at

Abstract. HAS-160 is an iterated cryptographic hash function that is standardized by the Korean government and widely used in Korea. In this paper, we present a semi-free-start collision for 65 (out of 80) steps of HAS-160 with practical complexity. The basic attack strategy is to construct a long differential characteristic by connecting two short ones by a complex third characteristic. The short characteristics are constructed using techniques from coding theory. To connect them, we are using an automatic search algorithm for the connecting characteristic utilizing the nonlinearity of the step function.

Keywords: differential attack, hash function, coding theory, collision

1 Introduction

In the last years research in cryptanalysis of hash function has made significant progress. Weaknesses have been shown in many commonly used hash functions as SHA-1 [19] and MD5 [18]. These breakthrough results in the cryptanalysis of hash functions were the motivation for intensive research in this field. Especially, in the ongoing SHA-3 [12] competition several new design strategies and attack techniques have been proposed. However, it also draws the attention away from currently used hash function standards, whereas it is important to analyze these standards to achieve a better understanding of the security margin in critical applications like e-commerce and e-government systems. In this paper, we focus on the hash function HAS-160. It is standardized by the Korean government (TTAS.KO-12.0011/R1) [17] and hence widely used in Korea. It is an iterated cryptographic hash function that produces a 160-bit hash value. The design of HAS-160 is similar to SHA-1 and MD5.

In [22], Yun et al. applied the techniques invented by Wang et al. in the cryptanalysis of MD5 and SHA-1 to the HAS-160 hash function. They show that a collision can be found for HAS-160 reduced to 45 steps with a complexity of about 2^{12}. This attack was later extended by Cho et al. [3] to HAS-160 reduced to 53 steps. The attack has a complexity of about 2^{55} 53-step HAS-160 computations. Mendel and Rijmen [10] improved the attack and reduced the complexity to 2^{35} and presented an actual colliding message pair for HAS-160 reduced to 53 steps. Furthermore, they presented a theoretical attack on 59 steps. Finally, preimage attacks on 52 steps by Sasaki and Aoki [16] and on 68 steps by Hong

H. Kim (Ed): ICISC 2011, LNCS 7259, pp. 33–47, 2012.
The original publication is available at http://www.springerlink.com/content/23578166216hl623/

et al. [6] have been presented. Both attacks have only theoretical complexity and are only slightly faster than the generic attack which has complexity 2^{160}.

In this paper, we combine different techniques to construct a semi-free start collision for 65 (out of 80) steps of HAS-160 with practical complexity. A semi-free-start collision is a collision attack where the adversary can choose the value of the initial value (IV). The basic idea of our attack is similar to the attack on a DES based hash function by Rijmen and Preneel [15] and to the recent attack on the SHA-3 candidate Skein by Yu et al. [21]. The idea is to construct a long differential characteristic by connecting two short ones by a complex third characteristic. We show how this idea can be applied on HAS-160 resulting in a semi-free start collision. Furthermore, we present an actual colliding message pair and IV fulfilling all conditions of the differential characteristics. This is so far the best attack in terms of number of steps on HAS-160 with practical complexity.

The remainder of this paper is structured as follows. A description of the hash function is given in Section 2. In Section 3 we describe the basic attack strategy. In Section 4 the search for two short differential characteristics and the determination of a good position for the connection is explained. In Section 5 we connect the short characteristics and present the final differential path. Finally, we present a colliding message pair in Section 5.3 and conclude in Section 6.

2 Description of HAS-160

HAS-160 is an iterative hash function that processes 512-bit input message blocks, operates on 32-bit words 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 [17].

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 shown in Table 1.
For the ordering of the expanded message words W_i the permutation in Table 2 is used.

State Update Transformation. The state update transformation of HAS-160 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.
Note that the function f is different in each round: f_0 is used in the first round,

Table 1. Message expansion of HAS-160.

	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}	$W_{12} \oplus W_{13} \oplus W_{14} \oplus W_{15}$	$W_7 \oplus W_{10} \oplus W_{13} \oplus W_0$	$W_8 \oplus W_1 \oplus W_{10} \oplus W_3$	$W_{11} \oplus W_6 \oplus W_1 \oplus W_{12}$

Table 2. Permutation of the message words.

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

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 $K_j \in \{\texttt{0}, \texttt{5a827999}, \texttt{6ed9eba1}, \texttt{8f1bbcdc}\}$ is added in every step and is different for each round. 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 in Table 3.

Table 3. Permutation of the message words.

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). 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.

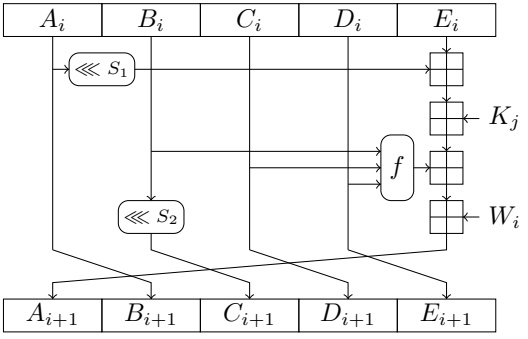

Fig. 1. The step function of HAS-160.

2.1 Alternative Description of HAS-160

As one can see in the description of the step update transformation (see Figure 1) only the state variable A_i is updated in each step. The values of the other state variables are defined by A_i. Therefore, we can redefine the state update such that only one state variable is used.

$$\begin{aligned} A_{i+1} = & A_{i-4} \ggg s_2 + A_i \lll s_1 + \\ & f(A_{i-1}, A_{i-2} \ggg s_2, A_{i-3} \ggg s_2) + \\ & K_j + W_i \end{aligned} \tag{1}$$

Note that s_2 need to be adapted accordingly if the update uses A's between two rounds. The chaining values are represented by $A_0, A_{-1}, A_{-2}, A_{-3}, A_{-4}$.

3 Basic Attack Strategy

In this section, we briefly describe the attack strategy to construct a semi-free start collision for 65 steps of HAS-160. A similar attack was done on a DES based hash function by Rijmen and Preneel [15] and recently on Skein by Yu et al. [21]. The main idea is to construct a long differential characteristic by connecting two short ones. First, proper differences in the expanded message words need to be chosen, such that they result in two short linear characteristics with low Hamming weight and hence hold with high probability. Second, we connect the two short differential characteristics by a third one. This one can have low probability, since we can use message modification to fulfill the conditions. Figure 2 illustrates the strategy.

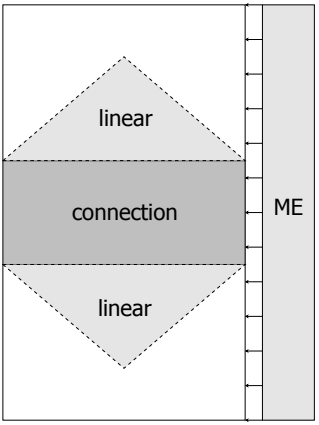

Fig. 2. Basic attack strategy. Differences occur only in the parts with background color.

The attack can be summarized as follows:

1. Choose an optimal position for the connection and find two differential characteristics, which hold with high probability.
2. Find a connecting differential characteristic.
3. Find inputs fulfilling the conditions and use message modification to improve the attack complexity.

To find two good characteristics and to determine an optimal position, we use a linearized model of the hash function. Finding a characteristic in a linearized hash function is not difficult. However, we aim for characteristics with high probability such that the available freedom can be used for the connection. The probability that the linear characteristic holds in the original hash function is related to the Hamming weight of the characteristic. In general, a differential characteristic with low Hamming weight has a higher probability than one with a high Hamming weight. Finding a characteristic with high probability (low Hamming weight) is related to finding a low weight word in linear codes. Therefore, we use a probabilistic algorithm from coding theory to find good characteristics. It has been shown in the past, for instance the cryptanalysis of SHA-0 [2], SHA-1 [13], EnRUPT [7] or SIMD [8] that this technique works well for finding differential characteristics with low Hamming weight.

We are constructing different linear codes for different positions and lengths of the connecting part to determine the optimal choice. Afterwards, we use an automatic search technique to find a connecting differential characteristic. Finally, we use message modification, introduced by Wang et al. in [20], to find inputs fulfilling all conditions.

4 Finding Two Short Characteristics

As mentioned before the problem of finding characteristics for a linearized hash function which hold with high probability for the original function is related to coding theory [8,13,14]. In order to find such characteristics for HAS-160 we need to linearize the hash function.

4.1 Linearization of HAS-160

Since the message expansion is already linear, only the step update transformation has to be linearized. The nonlinear parts of this function are the modular additions and the Boolean functions f_0 and f_2 (f_1 is linear). In the attack, we replace all modular addition by XORs. For the Boolean functions we tried several different linearizations. However, the following variant turned out to be the best. The function f_0 (IF) is replaced by the 0-function, i.e. we block each input difference in f_0. This has probability 1/2 in most cases (cf. [4]). One can see that there is exactly one input difference for f_0 where the output difference is always one. In that case we discard the characteristic. f_2 is approximated by its second input. which holds with probability higher than 1/2. In summary we get the following approximation for the Boolean functions:

$$f_0'(x, y, z) = 0$$
$$f_2'(x, y, z) = y$$

4.2 Construction of the Generator matrix

In this section we explain the standard approach to find collision producing characteristics for a linearized hash function. As observed by Rijmen and Oswald [14], all differential characteristics for a linearized hash function can be seen as the codewords of a linear code. Our goal is to find codewords with low Hamming weight, i.e. characteristics with high probability. Therefore, we have to include all intermediate chaining values where differences could decrease the success probability in the linear code. Based on the alternative description of HAS-160 (see Section 2.1) we include only A_i in the linear code, since the other state variables do not add any additional information to the code. This decreases the length of the code significantly and therefore also the running time of the search algorithm.

Let $\Delta A_i \in \{0,1\}^{32}$ be the difference vector of the chaining value A_i in bit representation at step i. Then the vector

$$cw := (\Delta A_1, \cdots, \Delta A_n), \tag{2}$$

where $cw \in \{0,1\}^{n \cdot 32}$, represents the differences in the chaining value A_i after each step of n steps of HAS-160. cw is one codeword of the linear code and therefore a differential characteristic. To construct the generator matrix for the linear code, we proceed as follows:

1. Compute cw_j with the input difference $\Delta M = e_j$, where $e_j \in \{0,1\}^{512}$ is the j-th unit vector and ΔM the difference of the message block in bit representation.
2. Repeat the computation for $j = 1, \ldots, 512$.

The resulting generator matrix of the linear code representing linearized HAS-160 is defined in the following way:

$$G_{512 \times n \cdot 32} := \begin{pmatrix} cw_1 \\ \vdots \\ cw_{512} \end{pmatrix}. \tag{3}$$

Since we are aiming for a collision in the last step, we need to apply code shortening on the last 160 bits, i.e. ensuring that all code words are zero in the last 160 bits. This reduces the dimension and length of the code to 352 and $(n \cdot 32 - 160)$, respectively.

Using this matrix one can search for low Hamming weight codewords over all n steps. As explained in Section 3 we are looking for two short characteristics, which will be connected later. Therefore, we need to modify the linear code to include this requirement.

Modification. The easiest way to define a linear code for both characteristics simultaneously and ensuring that both use the same expanded message, is the following. Firstly, ignore t steps in the middle. Hence, we change the vector (2) to:

$$cw := (\Delta A_1, \cdots, \Delta A_l, \Delta A_{l+t+1}, \cdots, \Delta A_n). \tag{4}$$

At the beginning of the second characteristic (after step $l+t$), the state variables can have any difference, since the differences in the steps before are yet undefined. Therefore, we need to add the information to the code that after step $l+t$ all differences are possible. Hence, we add the chaining variables at step $l+t+1$ to the linear code. The construction of the generator matrix changes to:

1. Compute cw_j with the input difference $\Delta M = e_j$, where $e_j \in \{0,1\}^{512}$ is the j-th unit vector and ΔM the difference of the message block in bit representation.
2. Repeat the computation for $j = 1, \ldots, 512$.
3. Compute cw_{512+k} as follows:
   (a) Set $\Delta M = 0$ and $cw_s = e_k$, where $e_k \in \{0,1\}^{160}$ is the k-th unit vector and
   $$cw_s = (\Delta A_{l+t-3}, \Delta A_{l+t-2}, \Delta A_{l+t-1}, \Delta A_{l+t}, \Delta A_{l+t+1}).$$
   (b) Compute ΔA_i for $(l+t+1) < i \leq n$ with cw_s and ΔM as input. Hence, we get following codeword:
   $$cw_{512+k} := (\Delta A_1 = 0, \cdots, \Delta A_l = 0, cw_s, \Delta A_{l+t+2}, \cdots, \Delta A_n).$$
4. Repeat the computation for $k = 1, \ldots, 160$.

Note that $\Delta B_{l+t+1} = \Delta A_{l+t}, \Delta C_{l+t+1} = \Delta A_{l+t-1}, \Delta D_{l+t+1} = \Delta A_{l+t-2}$ and $\Delta E_{l+t+1} = \Delta A_{l+t-3}$ and therefore all possible chaining values after step $l + t$ are included in the code. The resulting generator matrix is

$$G_{672\times(n-t+4)\cdot 32} := \begin{pmatrix} cw_1 \\ \vdots \\ cw_{672} \end{pmatrix}. \tag{5}$$

Again code shorting is applied to ensure that all codewords result in a collision after n steps.

Determining l, t and n. There exist several possible choices for the parameters l, t and n of the linear code. First of all we limit $t \leq 21$. The reason for this is simple. We have 21 words (16 message words and 5 IV words) which can be choosen freely and hence can be used for message modification to fulfill all conditions in the connecting part which is usually the most expensive part of the attack. However, we aimed for a smaller t to reduce the search space for the connecting part as well.

For the search we constructed generator matrices for $21 \leq l \leq (n - 21)$ and $t = 21$. If we have found two characteristics with high probability we reduce t.

4.3 Searching for Low Hamming Weight Codewords

We use the publicly available CodingTool Library [11] which contains all tools needed to search for codewords with low Hamming weight. It implements the probabilistic algorithm from Canteaut and Chabaud [1] to search for codewords with low Hamming weight. This iterative algorithm basically looks for small Hamming weight codewords in a smaller code. Such a codeword is considered as a good candidate for a low Hamming weight codeword for the whole code. The algorithm randomly selects σ columns of it and splits the selection in two sub-matrices of equal size. By computing all linear combination of p rows (usually 2 or 3) for each sub-matrix and storing their weight, the algorithm searches for a collision of both weights which allow to search for codewords of $2p$. Then two randomly selected columns are interchanged, followed by one Gaussian elimination step. This procedure is repeated until a sufficiently small Hamming weight is found. With this tool we can find good characteristics for different choices of l and t in few seconds on a standard PC. In Table 4 we present the best (lowest Hamming weight) characteristics we have found for different parameters. As one can see after 65 steps the Hamming weight is getting too high such that we cannot find a characteristic and conforming inputs with practical complexity.

Note that decreasing t always increases the Hamming weight, since more state variables with differences are included in the linear code. Furthermore, the Hamming weight in Table 4 includes only differences in A. To estimate the probability one has to take the differences in all state variables into account. Therefore, the probability for the linear characteristic can be roughly estimated by four times the Hamming weight of A.

Table 4. Results for the low weight search.

n	l	t	Hamming weight
53	18	21	3
60	18	21	3
65	18	21	3
66	19	21	25
67	18	21	25
68	18	21	72
69	18	21	72
70	18	21	119
75	19	21	123
80	19	21	247

Using this general approach we can cover the whole (linear) search space and allow arbitrary differences in the message words. However, it turned out that the best characteristics we have found are indeed the trivial ones which have only few differences in the message words and only a one bit difference per message word.

4.4 Short Differential Characteristics

To describe the differential characteristics we use generalized conditions which are explained in Section 5.1. We have found several different characteristics, depending on the choice of l and t. In Table 8 of Appendix A we present two short characteristics, where t is kept small. To improve readability, we used the alternative description of HAS-160 (see Section 2.1)

5 Finding Connecting Characteristics

In this section, we show how one can find a connecting differential characteristic which is the most expensive part in our attack. The main idea to find a connecting characteristic is to use the nonlinearity of the step update function. Constructing such complex characteristics is a difficult task. In [5], De Cannière and Rechberger proposed a new method to find complex characteristics for SHA-1 in an efficient way. In their concept they allow characteristics to impose arbitrary conditions on the pairs of bits (referred to as generalized conditions). Based on this they presented an efficient probabilistic search algorithm. Recently, Mendel *et al.*[9] extended this technique and applied it successfully on SHA-2. The basic idea of the search algorithm is to randomly pick a bit position and impose a zero-difference. Afterwards, it is calculated how this condition propagates. This is repeated until an inconsistency is found or all unrestricted bits are eliminated.

5.1 Generalized Conditions

To describe the search algorithm in more detail we first repeat the notation of generalized conditions which was introduced in [5]. Inspired by signed-bit differences, generalized conditions for differences take all 16 possible conditions on a pair of bits into account. Table 5 lists all these possible conditions and introduces notations for the various cases.

Table 5. Notation for possible generalized conditions on a pair of bits [5].

(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$	(X_i, X_i^*)	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

For example, all pairs of 8-bit words X and X^* that satisfy

$$\{(X, X^*) \in \{0,1\}^8 \times \{0,1\}^8 \mid X_7 \cdot X_7^* = 0, X_i = X_i^* \text{ for } 1 \leq i \leq 5, X_0 \neq X_0^*\},$$

can be conveniently written in the form

$$\nabla X = \texttt{[7?-----x]}.$$

5.2 Application to HAS-160

Due to the similarities of HAS-160 to SHA-1 the adaption of the above concept can be done in a straightforward manner and can be used to find the connecting characteristic. For more details see [5,9]. We proceed as follow:

1. Pick a random unrestricted bit (`?`) or an unsigned difference (`x`).
2. Impose a zero-difference (`-`) or randomly a sign (`u` or `n`), respectively.
3. Check how the new condition propagates.
4. If an inconsistency occurs jump back to the point where the last sign was imposed and make a different decision.
5. Repeat this until all unrestricted bits are eliminated

Using a small number of unrestricted words reduces the search space and running time of the algorithm significantly. Therefore, we reduced this number by extending the two short linear characteristics linearly. Since there are only few differences at the end of the first linear characteristic and at the beginning of the second linear characteristic, we can extend them forward and backward respectively, without increasing the Hamming weight too much. In fact for the

characteristic in Table 8 in Appendix A we extended the linear characteristics linearly forward by two and backwards by ten steps. Table 6 shows the starting point of the search algorithm using the notation of generalized conditions leaving only five words unrestricted.

Table 6. Steps free of conditions at the beginning of the search algorithm.

step	∇A	∇W
⋮	⋮	⋮
20	x----------x--x---------	--------------------------------
21	????????????????????????????????	--------------------------------
22	????????????????????????????????	--------------------------------
23	????????????????????????????????	--------------------------------
24	????????????????????????????????	--------------------------------
25	????????????????????????????????	x-------------------------------
26	-x-x------x---x-xxx--x-------x--	--------------------------------
⋮	⋮	⋮

Applying the above algorithm on this starting point the algorithm converges already after an hour (on a standard PC) to a complete characteristic for 65 steps. Determining the complexity of the probabilistic algorithm in general is still an open problem. Among others it depends on the hash function, search strategy, start characteristic and implementation. The complete characteristic is given in Table 8 of Appendix A. Note that with this approach we can find several different characteristics.

5.3 Finding a Message Pair

Almost all of the differences in the characteristic of Table 8 in Appendix A are within 21 steps. Since we can choose up to 21 words (16 message and 5 IV) freely we can use message modification to find efficiently inputs which fulfill all the conditions of the characteristic. The conditions for the characteristic are listed in Table 9 in Appendix A. The resulting colliding message pair and IV is given in Table 7.

6 Conclusions

The progress in the cryptanalysis of hash functions in the last years shows that the security of existing standards need to be reevaluated. Therefore, we analyze in this paper the Korean hash function standard (TTAS.KO-12.0011/R1) HAS-160. The main idea of our attack is to construct two short linear differential characteristics which hold with high probability and connect them by a complex third characteristic by using the nonlinearity of the state update function.

Table 7. A colliding message pair and IV for HAS-160.

IV	ed3c8ca6 38127dc3 bcf7b374 264eeb2b 73be1247
M	467d7948 3c433177 981f570c 6bf43c12 3dc04b7c cb85a46d 3356206e bff3ea04 9603f6ca 252c37eb 3a1d6197 479ca8d1 badbe3d9 4e23c48c c52a6189 53f1ea06
M'	467d7948 3c433177 981f570c 6bf43c12 3dc04b7c cb85a46d 3356206e bff3ea04 9603f6ca 252c37eb 3a1d6197 479ca8d1 3adbe3d9 4e23c48c 452a6189 53f1ea06
ΔM	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 80000000 00000000 80000000 00000000
h	4b0a28ae bc82dbb1 a4805bfd cd226435 7cb7eb52
h'	4b0a28ae bc82dbb1 a4805bfd cd226435 7cb7eb52

We use techniques from coding theory to search efficiently for the short characteristics and simultaneously determine an optimal position and length of the connecting characteristic. In a second step we use an automatic search algorithm to find a connecting characteristic taking the nonlinearity of the state update into account.

We present a semi-free-start collision for 65 (out of 80) steps HAS-160 with practical complexity. Extending the attack to more rounds seems to be difficult. One can always extend the size of the connecting part, but this also increases the complexity of finding the connecting characteristic, which running time is hard to estimate. If we limit the length of the connecting part to 21 steps, then the best short characteristics we can find with probability below the generic complexity of a collision attack, are for up to 65 steps.

Even though we only present a semi-free-start collision, it is a step forward in the analysis of HAS-160. This is so far the best known attack with practical complexity in terms of attacked steps for HAS-160.

Acknowledgments

The work in this paper has been supported by the European Commission under contract ICT-2007-216646 (ECRYPT II) and by the Austrian Science Fund (FWF, project P21936).

References

1. Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.
2. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
3. Hong-Su Cho, Sangwoo Park, Soo Hak Sung, and Aaram Yun. Collision Search Attack for 53-Step HAS-160. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC*, volume 4296 of *LNCS*, pages 286–295. Springer, 2006.

4. Magnus Daum. *Cryptanalysis of Hash Functions of the MD4-Family.* PhD thesis, Ruhr-Universität Bochum, May 2005. Available online: `http://www.cits.rub.de/imperia/md/content/magnus/dissmd4.pdf`.
5. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.
6. Deukjo Hong, Bonwook Koo, and Yu Sasaki. Improved Preimage Attack for 68-Step HAS-160. In Donghoon Lee and Seokhie Hong, editors, *ICISC*, volume 5984 of *Lecture Notes in Computer Science*, pages 332–348. Springer, 2009.
7. Sebastiaan Indesteege and Bart Preneel. Practical Collisions for EnRUPT. In Orr Dunkelman, editor, *FSE*, volume 5665 of *LNCS*, pages 246–259. Springer, 2009.
8. Florian Mendel and Tomislav Nad. A Distinguisher for the Compression Function of SIMD-512. In Bimal K. Roy and Nicolas Sendrier, editors, *INDOCRYPT*, volume 5922 of *LNCS*, pages 219–232. Springer, 2009.
9. Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding SHA-2 Characteristics: Searching Through a Minefield of Contradictions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, LNCS. Springer, 2011. To appear.
10. Florian Mendel and Vincent Rijmen. Colliding Message Pair for 53-Step HAS-160. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 324–334. Springer, 2007.
11. Tomislav Nad. The CodingTool Library. Workshop on Tools for Cryptanalysis 2010, 2010. `http://www.iaik.tugraz.at/content/research/krypto/codingtool/`.
12. National Institute of Standards and Technology. Cryptographic Hash Algorithm Competition, November 2007. Available online: `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`.
13. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
14. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
15. Vincent Rijmen and Bart Preneel. Improved Characteristics for Differential Cryptanalysis of Hash Functions Based on Block Ciphers. In Bart Preneel, editor, *FSE*, volume 1008 of *LNCS*, pages 242–248. Springer, 1994.
16. Yu Sasaki and Kazumaro Aoki. A Preimage Attack for 52-Step HAS-160. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, volume 5461 of *LNCS*, pages 302–317. Springer, 2008.
17. Telecommunications Technology Association. Hash Function Standard Part 2: Hash Function Algorithm Standard (HAS-160), TTAS.KO-12.0011/R1, 2008.
18. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
19. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
20. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
21. Hongbo Yu, Jiazhe Chen, Ketingjia, and Xiaoyun Wang. Near-Collision Attack on the Step-Reduced Compression Function of Skein-256. Cryptology ePrint Archive, Report 2011/148, 2011.

22. Aaram Yun, Soo Hak Sung, Sangwoo Park, Donghoon Chang, Seokhie Hong, and Hong-Su Cho. Finding Collision on 45-Step HAS-160. In Dongho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *LNCS*, pages 146–155. Springer, 2005.

A Characteristic

Table 8. Characteristic for 65 steps HAS-160 using generalized conditions. The rows with darkgray background represent the connecting part. The rows with lightgray background represent the two linear characteristics. All conditions can be fulfilled using message modification.

step	∇A	∇W
-4	--------------------------------	
-3	--------------------------------	
-2	--------------------------------	
-1	--------------------------------	
0	--------------------------------	
⋮	⋮	⋮
16	--------------------------------	--------------------------------
17	u--------------------------------	x-------------------------------
18	----------------u---------------	--------------------------------
19	n----------u-------------------	x-------------------------------
20	u----------------u--u----------	--------------------------------
21	-------n-uuuuuu--u----n----u----	--------------------------------
22	u--n---uu-nu---uu---nn--------uu	--------------------------------
23	--n-n-nnnu-n-u--nu------nu------	--------------------------------
24	uuun-nu--u-u----n-n-unnuuuuuuu-n	--------------------------------
25	--n----uu---uu-un-u-----nu-n-n--	x-------------------------------
26	-n-n------n---n-uun--u-------n--	--------------------------------
27	-unu------u-n---uu---u-n-u-u---n	--------------------------------
28	--n---u---u---u--u-n---u-----u-n	--------------------------------
29	--------n---u--------n-------u-n	--------------------------------
30	--u-----n-u----------u----------	--------------------------------
31	--n-------n-----------------n-n--	--------------------------------
32	-------------n--------------n---	--------------------------------
33	----------n-----------------------	x-------------------------------
34	----------n----------------u---	--------------------------------
35	----------------------------u---	--------------------------------
36	--------------------------------	--------------------------------
37	--------------------------------	--------------------------------
38	--------------------------------	--------------------------------
39	------------------------n------	--------------------------------
40	--------------------------------	--------------------------------
41	--------------------------------	--------------------------------
42	--------------------------------	x-------------------------------
43	--------------------------------	--------------------------------
44	--------------------------------	x-------------------------------
45	--------------------------------	--------------------------------
⋮	⋮	⋮
65	--------------------------------	--------------------------------

Table 9. Set of conditions for the semi-free-start collision for 65 steps.

step	set of conditions	#
16	$A_{16,3} = 0$, $A_{16,21} = A_{15,21}$	2
17	$A_{17,3} = 1$, $A_{17,31} = 1$	2
18	$A_{18,9} = 1$, $A_{18,13} = 1$, $A_{18,8} \neq A_{17,8}$	3
19	$A_{19,18} = 1$, $A_{19,31} = 0$, $A_{19,23} \neq A_{17,13}$, $A_{19,27} \neq A_{18,2}$, $A_{19,9} \neq A_{18,31}$, $A_{19,24} = A_{18,31}$	6
20	$A_{20,9} = 1$, $A_{20,12} = 1$, $A_{20,31} = 1$, $A_{20,16} \neq A_{18,6}$, $A_{20,3} = A_{18,25}$, $A_{20,0} \neq A_{19,0}$, $A_{20,1} = A_{19,1}$, $A_{20,2} = A_{19,2}$, $A_{20,3} = A_{19,3}$, $A_{20,4} = A_{19,4}$, $A_{20,5} = A_{19,5}$, $A_{20,23} \neq A_{19,6}$, $A_{20,7} = A_{19,7}$, $A_{20,19} = A_{19,19}$, $A_{20,24} = A_{19,24}$, $A_{20,29} \neq A_{19,29}$	16
21	$A_{21,4} = 1$, $A_{21,9} = 0$, $A_{21,14} = 1$, $A_{21,17} = 1$, $A_{21,18} = 1$, $A_{21,19} = 1$, $A_{21,20} = 1$, $A_{21,21} = 1$, $A_{21,22} = 1$, $A_{21,24} = 0$, $A_{21,26} \neq A_{19,9}$, $A_{21,29} = A_{19,12}$, $A_{21,3} \neq A_{20,3}$, $A_{21,6} \neq A_{20,6}$, $A_{21,7} \neq A_{20,7}$, $A_{21,11} \neq A_{20,11}$, $A_{21,15} \neq A_{20,15}$, $A_{21,16} \neq A_{20,16}$, $A_{21,3} \neq A_{20,18}$, $A_{21,25} \neq A_{20,25}$, $A_{21,26} = A_{20,26}$, $A_{21,30} = A_{20,30}$	22
22	$A_{22,0} = 1$, $A_{22,1} = 1$, $A_{22,10} = 0$, $A_{22,11} = 0$, $A_{22,15} = 1$, $A_{22,16} = 1$, $A_{22,20} = 1$, $A_{22,21} = 0$, $A_{22,23} = 1$, $A_{22,24} = 1$, $A_{22,28} = 0$, $A_{22,31} = 1$, $A_{22,2} = A_{20,17}$, $A_{22,3} = A_{20,18}$, $A_{22,4} \neq A_{20,19}$, $A_{22,5} \neq A_{20,20}$, $A_{22,6} = A_{20,21}$, $A_{22,7} \neq A_{20,22}$, $A_{22,9} = A_{20,24}$, $A_{22,3} \neq A_{21,3}$, $A_{22,5} = A_{21,5}$, $A_{22,6} \neq A_{21,6}$, $A_{22,7} = A_{21,7}$, $A_{22,8} = A_{21,8}$, $A_{22,12} \neq A_{21,12}$, $A_{22,29} \neq A_{21,12}$, $A_{22,29} = A_{21,29}$, $A_{22,30} = A_{21,30}$	28
23	$A_{23,6} = 1$, $A_{23,7} = 0$, $A_{23,14} = 1$, $A_{23,15} = 0$, $A_{23,18} = 1$, $A_{23,20} = 0$, $A_{23,22} = 1$, $A_{23,23} = 0$, $A_{23,24} = 0$, $A_{23,25} = 0$, $A_{23,27} = 0$, $A_{23,29} = 0$, $A_{23,17} = A_{21,0}$, $A_{23,28} \neq A_{21,11}$, $A_{23,0} \neq A_{21,15}$, $A_{23,1} \neq A_{21,16}$, $A_{23,8} = A_{21,23}$, $A_{23,13} = A_{21,28}$, $A_{23,16} = A_{21,31}$, $A_{23,3} = A_{22,3}$, $A_{23,21} \neq A_{22,4}$, $A_{23,5} = A_{22,5}$, $A_{23,8} = A_{22,8}$, $A_{23,9} \neq A_{22,9}$, $A_{23,26} = A_{22,9}$, $A_{23,12} \neq A_{22,12}$, $A_{23,13} \neq A_{22,13}$, $A_{23,2} = A_{22,17}$, $A_{23,17} \neq A_{22,17}$, $A_{23,3} = A_{22,18}$, $A_{23,4} \neq A_{22,19}$, $A_{23,19} \neq A_{22,19}$, $A_{23,26} \neq A_{22,26}$, $A_{23,30} \neq A_{22,30}$	34
24	$A_{24,0} = 0$, $A_{24,2} = 1$, $A_{24,3} = 1$, $A_{24,4} = 1$, $A_{24,5} = 1$, $A_{24,6} = 1$, $A_{24,7} = 1$, $A_{24,8} = 1$, $A_{24,9} = 0$, $A_{24,10} = 0$, $A_{24,11} = 1$, $A_{24,13} = 0$, $A_{24,15} = 0$, $A_{24,20} = 1$, $A_{24,22} = 1$, $A_{24,25} = 1$, $A_{24,26} = 0$, $A_{24,28} = 0$, $A_{24,29} = 1$, $A_{24,30} = 1$, $A_{24,31} = 1$, $A_{24,23} = A_{22,6}$, $A_{24,24} \neq A_{22,7}$, $A_{24,12} = A_{22,27}$, $A_{24,14} = A_{22,29}$, $A_{24,17} \neq A_{23,0}$, $A_{24,1} \neq A_{23,1}$, $A_{24,18} \neq A_{23,1}$, $A_{24,27} = A_{23,10}$, $A_{24,12} \neq A_{23,12}$, $A_{24,1} = A_{23,16}$, $A_{24,17} \neq A_{23,17}$, $A_{24,19} = A_{23,19}$, $A_{24,21} = A_{23,21}$, $A_{24,16} \neq A_{23,31}$	35
25	$A_{25,2} = 0$, $A_{25,4} = 0$, $A_{25,6} = 1$, $A_{25,7} = 0$, $A_{25,13} = 1$, $A_{25,15} = 0$, $A_{25,16} = 1$, $A_{25,18} = 1$, $A_{25,19} = 1$, $A_{25,23} = 1$, $A_{25,24} = 1$, $A_{25,29} = 0$, $A_{25,17} \neq A_{23,0}$, $A_{25,20} = A_{23,3}$, $A_{25,21} = A_{23,4}$, $A_{25,22} \neq A_{23,5}$, $A_{25,25} \neq A_{23,8}$, $A_{25,26} \neq A_{23,9}$, $A_{25,27} = A_{23,10}$, $A_{25,28} = A_{23,11}$, $A_{25,30} = A_{23,13}$, $A_{25,11} = A_{23,26}$, $A_{25,17} = A_{24,17}$, $A_{25,3} = A_{24,18}$, $A_{25,8} = A_{24,23}$, $A_{25,9} = A_{24,24}$, $A_{25,12} = A_{24,27}$	27
26	$A_{26,2} = 0$, $A_{26,10} = 1$, $A_{26,13} = 0$, $A_{26,14} = 1$, $A_{26,15} = 1$, $A_{26,17} = 0$, $A_{26,21} = 0$, $A_{26,28} = 0$, $A_{26,30} = 0$, $A_{26,1} = A_{24,16}$, $A_{26,3} = A_{24,18}$, $A_{26,4} \neq A_{24,19}$, $A_{26,8} = A_{24,23}$, $A_{26,9} = A_{24,24}$, $A_{26,20} \neq A_{25,3}$, $A_{26,22} \neq A_{25,5}$, $A_{26,25} \neq A_{25,8}$, $A_{26,26} = A_{25,9}$, $A_{26,27} = A_{25,10}$, $A_{26,11} = A_{25,11}$, $A_{26,12} = A_{25,12}$, $A_{26,5} = A_{25,20}$, $A_{26,7} = A_{25,22}$, $A_{26,25} \neq A_{25,25}$, $A_{26,11} = A_{25,26}$, $A_{26,16} \neq A_{25,31}$	26
27	$A_{27,0} = 0$, $A_{27,4} = 1$, $A_{27,6} = 1$, $A_{27,8} = 0$, $A_{27,10} = 1$, $A_{27,14} = 1$, $A_{27,15} = 1$, $A_{27,19} = 0$, $A_{27,21} = 1$, $A_{27,28} = 1$, $A_{27,29} = 0$, $A_{27,30} = 1$, $A_{27,27} \neq A_{25,10}$, $A_{27,2} = A_{25,17}$, $A_{27,13} = A_{25,28}$, $A_{27,23} \neq A_{26,6}$, $A_{27,24} = A_{26,7}$, $A_{27,12} \neq A_{26,12}$, $A_{27,1} \neq A_{26,16}$, $A_{27,3} \neq A_{26,18}$, $A_{27,23} = A_{26,23}$, $A_{27,9} = A_{26,24}$, $A_{27,27} \neq A_{26,27}$	23
28	$A_{28,0} = 0$, $A_{28,2} = 1$, $A_{28,8} = 1$, $A_{28,12} = 0$, $A_{28,14} = 1$, $A_{28,17} = 1$, $A_{28,21} = 1$, $A_{28,25} = 1$, $A_{28,29} = 0$, $A_{28,23} = A_{26,6}$, $A_{28,4} = A_{26,19}$, $A_{28,19} \neq A_{27,2}$, $A_{28,30} \neq A_{27,13}$	13
29	$A_{29,0} = 0$, $A_{29,2} = 1$, $A_{29,10} = 0$, $A_{29,19} = 1$, $A_{29,23} = 0$, $A_{29,29} = A_{27,12}$, $A_{29,4} \neq A_{28,4}$, $A_{29,21} \neq A_{28,4}$, $A_{29,6} = A_{28,6}$, $A_{29,27} \neq A_{28,10}$, $A_{29,4} = A_{28,19}$, $A_{29,13} = A_{28,28}$, $A_{29,15} = A_{28,30}$	13
30	$A_{30,10} = 1$, $A_{30,21} = 1$, $A_{30,23} = 0$, $A_{30,29} = 1$, $A_{30,27} = A_{28,10}$, $A_{30,4} = A_{28,19}$, $A_{30,8} \neq A_{28,23}$, $A_{30,4} = A_{29,4}$, $A_{30,25} = A_{29,8}$, $A_{30,12} \neq A_{29,12}$, $A_{30,2} \neq A_{29,17}$, $A_{30,17} = A_{29,17}$, $A_{30,6} \neq A_{29,21}$, $A_{30,14} \neq A_{29,29}$	14
31	$A_{31,2} = 0$, $A_{31,4} = 0$, $A_{31,21} = 0$, $A_{31,29} = 0$, $A_{31,6} \neq A_{29,21}$, $A_{31,14} = A_{29,29}$, $A_{31,0} \neq A_{30,0}$, $A_{31,17} \neq A_{30,0}$, $A_{31,19} \neq A_{30,2}$, $A_{31,17} \neq A_{30,17}$	10
32	$A_{32,2} = 0$, $A_{32,17} = 0$, $A_{32,19} = A_{30,2}$, $A_{32,21} = A_{30,4}$, $A_{32,27} = A_{31,10}$, $A_{32,8} \neq A_{31,23}$	6
33	$A_{33,21} = 0$, $A_{33,2} \neq A_{31,17}$, $A_{33,4} \neq A_{32,4}$, $A_{33,6} = A_{32,21}$, $A_{33,14} = A_{32,29}$	5
34	$A_{34,2} = 1$, $A_{34,21} = 0$, $A_{34,6} \neq A_{32,21}$, $A_{34,19} \neq A_{33,2}$, $A_{34,17} = A_{33,17}$	5
35	$A_{35,2} = 1$, $A_{35,19} = A_{33,2}$	2
36	$A_{36,6} \neq A_{35,21}$	1
37	$A_{37,21} = 0$, $A_{37,19} \neq A_{36,2}$	2
39	$A_{39,6} = 0$	1
41	$A_{41,31} = 1$	1