A heuristic for finding compatible differential paths with application to HAS-160

Aleksandar Kircanski, Riham AlTawy and Amr M. Youssef

Concordia Institute for Information Systems Engineering,
Concordia University, Montréal, Quebéc, H3G 1M8, CANADA.

Abstract. The question of compatibility of differential paths plays a central role in second order collision attacks on hash functions. In this context, attacks typically proceed by starting from the middle and constructing the middle-steps quartet in which the two paths are enforced on the respective faces of the quartet structure. Finding paths that can fit in such a quartet structure has been a major challenge and the currently known compatible paths extend over a suboptimal number of steps for hash functions such as SHA-2 and HAS-160. In this paper, we investigate a heuristic that searches for compatible differential paths. The application of the heuristic in case of HAS-160 yields a practical second order collision over all of the function steps, which is the first practical result that covers all of the HAS-160 steps. An example of a colliding quartet is provided.

1 Introduction

Whenever two probabilistic patterns are combined for the purpose of passing through maximal number of rounds of a cryptographic primitive, a natural question that arises is the question of compatibility of the two patterns. A notable example is the question of compatibility of differential paths in the context of boomerang attacks. In 2011, Murphy [23] has shown that care should be exercised when estimating the boomerang attack success probability, since there may exist dependency between the events that the two paths behave as required by the boomerang setting. The extreme case is the impossibility of combining the two paths, where the corresponding probability is equal to 0.

In the context of constructing second order collisions for compression functions using the start-from-the-middle technique, due to availability of message modification in the steps where the primitive follows the two paths, the above mentioned probability plays less of a role as long as it is strictly greater than 0. In that case, the two paths are said to be compatible. Several paths that were previously believed to be compatible have been shown to be incompatible in the previously described sense, e.g., by Leurent [13] and Sasaki [27] for BLAKE and RIPEMD-160 hash functions, respectively.

The compatibility requirement in this context can be stated with more precision as follows. Let ϕ and ω be two differential paths over some number of steps of an iterative function $f = f_{j+n} \circ \ldots \circ f_j$. If there exists a quartet of f inputs x_0, x_1, x_2 and x_3 such that computations (x_0, x_1) and (x_2, x_3) follow ϕ whereas (x_0, x_2) and (x_1, x_3) follow ω, we say that ϕ and ω are compatible. Usually the path ϕ is left unspecified over the last k steps (backward path) and ω is unspecified over the remaining steps (forward path). Such paths have also been previously called *independent* [4]. Another closely related notion is the concept of *non-interleaving* paths in the context of biclique attacks [9].

Our Contributions. In this paper, we present a heuristic that allows us to search for compatible differential paths. The heuristic builds on the previous de Cannière and Rechberger automatic differential path search method. Instead of working with pairs, our proposed heuristic operates on quartets of hash executions and includes cross-path propagations. We present detailed examples of particular propagations applied during the search. As an application of our proposed heuristic, a second order collision for the full HAS-160 compression function is found. The best previous practical distinguisher for this function covered steps 5 to 80 [28]. This is the first practical distinguisher for the full HAS-160. This particular hash function is relevant as it is standardized by the Korean government (TTAS.KO-12.0011/R1) [1].

Related Work. The differential paths used in groundbreaking attacks on MD4, MD5 and SHA-1 [33, 32] were found manually. Subsequently, several techniques for automatic differential path search have been studied [29, 7, 30, 5]. The de Cannière and Rechberger heuristic [5] was subsequently applied to many MDx/SHA-x based hash functions, such as RIPEMD-128, HAS-160, SHA-2 and SM3 [19, 17, 18, 20]. To keep track of the current information in the system, the heuristic relies on 1-bit constraints that express the relations between pairs of bits in the differential setting. This was generalized to multi-bit constraints by Leurent [13], where finite state machine approach allowed uniform representation of different constraint types. Multi-bit constraints have been used in the context of differential path search in [14].

The boomerang attack [31], originally applied to block ciphers, has been adapted to the hash function setting independently by Biryukov *et al.* [4] and by Lamberger and Mendel [12]. In particular, in [4], a distinguisher for the 7-round BLAKE-32 was provided, whereas in [12] a distinguisher for the 46-step reduced SHA-2 compression function was provided. The latter SHA-2 result was extended to 47 steps [3]. Subsequently, boomerang distinguishers have been applied to many hash functions, such as HAVAL, RIPEMD-160, SIMD, HAS-160, SM3 and Skein [25, 27, 28, 16, 11, 34, 15]. Outside of the boomerang context, zero-sum property as a distinguishing property was first used by Aumasson [2].

As for the previous HAS-160 analysis, in 2005, Yun *et al.* [35] found a practical collision for the 45-step (out of 80) reduced hash function. Their attack was extended in 2006 to 53 steps by Cho *et al.* [6], however, with computational complexity of 2^{55} 53-step compression function computations. In 2007, Mendel and Rijmen [21] improved the latter attack complexity to 2^{35}, providing a practical two-block message collision for the 53-step compression function. Preimage attacks on 52-step HAS-160 with complexity of 2^{152} was provided in 2008 by Sasaki and Aoki [26]. Subsequently, in 2009, this result was extended by Hong *al.* to 68 steps [8] where the attack required a complexity of $2^{156.3}$. In 2011, Mendel *et al.* provided a practical semi-free-start collision for 65-step reduced compression function [17]. Finally, in 2012, Sasaki *et al.* [28] provided a theoretical boomerang distinguisher for the full HAS-160 compression function, requiring $2^{76.6}$ steps function computations. In the same work, a practical second order collision was given for steps 5 to 80 of the function.

Paper Outline. In the next section, we provide the review of boomerang distingiushers and the recapitulation of the de Cannière and Rechberger search heuristic, along with the HAS-160 specification. In Section 3, the general form of the our search heuristic is provided and its application to HAS-160 is discussed. The three propagation types used in the heuristic are explained in Section 4. Concluding remarks are in given Section 5.

2 Review of related work and the specification of HAS-160

In the following subsections, we provide a description of a commonly used strategy to construct second order collisions, an overview of the de Cannière and Rechberger path search heuristic and finally the specification of HAS-160 hash function.

2.1 Review of boomerang distinguishers for hash functions

First, we provide a generic definition of the property used for distinguishing the compression function from a random function. Let h be a function with n-bit output. A *second order collision* for h is a set $\{x, \Delta, \nabla\}$ consisting of an input for h and two differences, such that

$$h(x + \Delta + \nabla) - h(x + \Delta) - h(x + \nabla) + h(x) = 0 \tag{1}$$

As explained in [3], the query complexity for finding a second order collision is $3 \cdot 2^{n/3}$ where n denotes the bit-size of the output of the function f. By the query complexity, the number of queries required to be made to h function is considered. On the other hand, for the computational complexity, which would include evaluating h around $3 \cdot 2^{n/3}$ times and finding a quartet that sums to 0, the best currently known algorithm runs in complexity no better than $2^{n/2}$. If for a particular function a second order collision is obtained with a complexity lower than $2^{n/2}$, then this hash function deviates from the random function oracle.

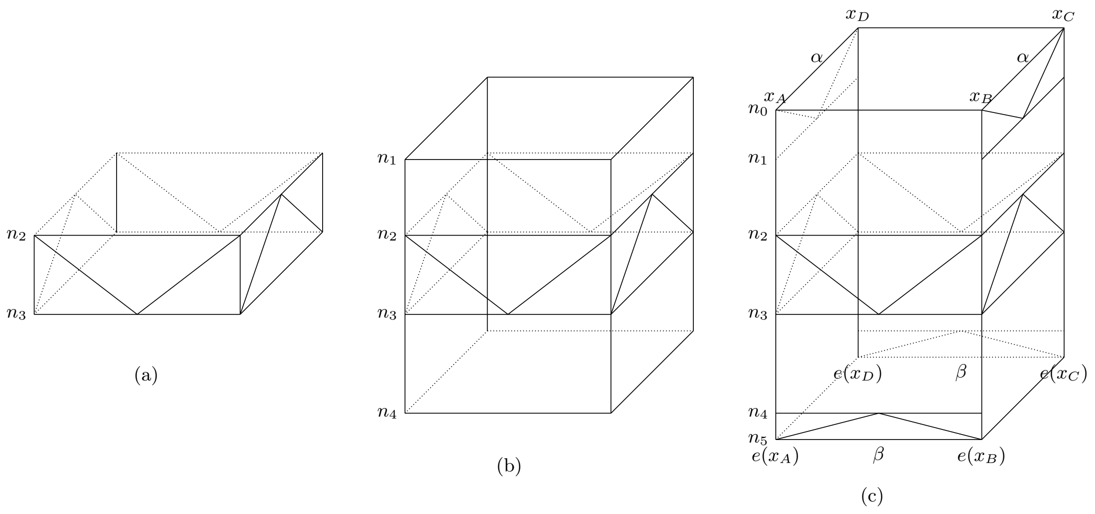

Fig. 1. Start-from-the-middle approach for constructing second-order collisions

Next, we explain the strategy to construct quartets satisfying (1) for Davies-Meyer based functions, as commonly applied in the previous literature. An overview of the strategy is provided in Fig. 1. We write $h(x) = e(x) + x$, where e is an iterative function consisting of n steps. The goal is to find four inputs x_A, x_B, x_C and x_D that constitute the inputs in (1) according to Fig. 1 (c). In particular, the goal is to have

$$\begin{gathered} x_A - x_D = x_B - x_C \\ e(x_A) - e(x_B) = e(x_D) - e(x_C) \end{gathered} \tag{2}$$

where the two values specified by (2) are denoted respectively by α and β in Fig. 1 (c). In this case, we have $h(x_A) - h(x_B) + h(x_C) - h(x_D) = e(x_A) + x_A - e(x_B) - x_B + e(x_C) + x_C - e(x_D) - x_D = 0$. Now, one can put $x_A = x$, $\Delta = x_D - x_A$ and $\nabla = x_B - x_A$ and (1) is satisfied.

A preliminary step is to decide on two paths, called the *forward* path and the *backward* path. As shown on Fig. 1, these paths are chosen so that for some $n_0 < n_1 < n_2 < n_3 < n_4 < n_5$, the forward path has no active bits between steps n_3 and n_4 and the backward path has no active bits between steps n_1 and n_2. The forward path is enforced on faces (x_A, x_B) and (x_D, x_C) (front and back) whereas the backward differential is enforced on faces (x_A, x_D) and (x_B, x_C) (left and right). In the case of MDx-based designs, the particular n values depend mostly on the message schedule specification.

The procedure can be summarized as follows:

(a) The first step is to construct the middle part of the quartet structure, as shown in Fig. 1 (a). The forward and backward paths end at steps n_3 and n_2, respectively. On steps n_2 to n_3, the two paths need to be compatible for this stage to succeed.
(b) Following Fig. 1 (b), the paths are extended to steps n_1 backward and to n_4 forward with probability 1, due to the absence of disturbances in the corresponding steps.
(c) Some of the middle-step words are randomized and the quartet is recomputed backward and forward, verifying if (2) is satisfied. If yes (see Fig. 1 (c)), return the quartet, otherwise, repeat this step.

This strategy, with variations, has been applied in several previous works, such as [3, 28, 27, 25]. In Table 1, we provide the forward/backward path parameters for the previous boomerang distinguishers on some of the MDx/SHA-x based compression functions following the single-pipe design strategy.

Compression function	n_0	n_1	n_2	n_3	n_4	n_5	Reference	Message block size
SHA-2	0	6	22	31	47	47	[3]	16×32
HAVAL	0	2	61	97	157	160	[25]	32×32
HAS-160	5	13	38	53	78	80	[28]	16×32

Table 1. Overview of some of the previously used boomerang paths

In [3, 28], the number of steps in the middle was 9 and 16 steps, respectively. It can be observed that these number of middle steps are suboptimal, since the simple message modification allows trivially satisfying 16 steps in case of SHA-2 and HAS-160. Since the forward and the backward paths are sparse towards steps n_3 and n_2, one can easily imagine satisfying more than 16 steps, while there remains enough freedom to randomize the inner state although some penalty in probability has to be paid. In case of HAVAL [25], the simple message modification allows passing through 32 steps and the middle part consists of as many as 36 steps. However, it should be noted that this is due to the particular property of HAVAL which allows narrow paths [10].

2.2 Review of de Cannière and Rechberger search heuristic

This search heuristic is used to find differential paths that describe pairs of compression function executions. The symbols used for expressing differential paths are provided in Table 2. For example, when we write `-x-u`, we mean a set of 4-bit pairs

$$\texttt{-x-u} = \{T, T' \in F_2^4 | T_3 = T_3', T_2 \neq T_2', T_1 = T_1', T_0 = 0, T_0' = 1\}$$

where T_i denotes i-th bit in word T.

$\delta(x, x')$	meaning	(0,0)	(0,1)	(1,0)	(1,1)
?	anything	√	√	√	√
-	$x = x'$	√	-	-	√
x	$x \neq x'$	-	√	√	-
0	$x = x' = 0$	√	-	-	-
u	$(x, x') = (0, 1)$	-	√	-	-
n	$(x, x') = (1, 0)$	-	-	√	-
1	$x = x' = 1$	-	-	-	√
#		-	-	-	-

$\delta(x, x')$	meaning	(0,0)	(0,1)	(1,0)	(1,1)
3	$x = 0$	√	√	-	-
5	$x' = 0$	√	-	√	-
7		√	√	√	-
A	$x' = 1$	-	√	-	√
B		√	√	-	√
C	$x = 1$	-	-	√	√
D		√	-	√	√
E		-	√	√	√

Table 2. Symbols used to express 1-bit conditions [5]

Next, an example of *condition propagation* is provided. Suppose that a small differential path over one modular addition is given by

$$\texttt{----} + \texttt{---x} = \texttt{---x} \tag{3}$$

Here (3) describes a pair of additions: $x + y = z$ and $x' + y' = z'$, and from this "path" we have that $x = x'$ and also that y and y' are different only on the least significant bit (same for z and z'). However, this can happen only if $x_0 = x_0' = 0$, i.e. if the lsb of x and x' is equal to 0. We thus *propagate a condition* by substituting (3) with

$$\texttt{---0} + \texttt{---x} = \texttt{---x}$$

The de Cannière and Rechberger heuristic [5] searches for differential paths over some number of compression function steps. It starts from a partially specified path which typically means that the path is fully specified at some steps (i.e., consisting of symbols {`-`,`u`,`n`}) and unspecified at other steps (i.e., symbol '`?`'). The heuristic attempts to complete the path, so that the final result is non-contradictory by proceeding as follows:

- *Guess*: select randomly a bit position containing '`?`' or '`x`'. Substitute the symbol in the chosen bit position by '`-`' and {`u`,`n`}, respectively.
- *Propagate*: deduce new information introduced by the *Guess* step.

When a contradiction is detected, the search backtracks by jumping back to one of the guesses and attempts different choices.

2.3 HAS-160 specification

The HAS-160 hash function follows the MDx/SHA-x hash function design strategy. Its compression function can be seen as a block cipher in Davies-Meyer mode, mapping 160-bit chaining values and 512-bit messages into 160-bit digests. To process arbitrary-length messages, the compression function is plugged in the Merkle-Damgård mode.

Before hashing, the message is padded so that its length becomes multiple of 512 bits. Since padding is not relevant for this paper, we refer the reader to [1] for further details. The underlying HAS-160 block cipher consists of two parts: message expansion and state update transformation.

Message expansion: The input to the compression function is a message $m = (m_0, \dots m_{15})$ represented as 16 32-bit words. The output of the message expansion is a sequence of 32-bit words $W_0, \dots W_{79}$. The expansion is specified in Table 3. For example, $W_{26} = m_{15}$.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$	m_0	m_1	m_2	m_3	$m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$	m_4	m_5	m_6	m_7	$m_0 \oplus m_1 \oplus m_2 \oplus m_3$	m_8	m_9	m_{10}	m_{11}	$m_4 \oplus m_5 \oplus m_6 \oplus m_7$	m_{12}	m_{13}	m_{14}	m_{15}
$m_{11} \oplus m_{14} \oplus m_1 \oplus m_4$	m_3	m_6	m_9	m_{12}	$m_7 \oplus m_{10} \oplus m_{13} \oplus m_0$	m_{15}	m_2	m_5	m_8	$m_3 \oplus m_6 \oplus m_9 \oplus m_{12}$	m_{11}	m_{14}	m_1	m_4	$m_{15} \oplus m_2 \oplus m_5 \oplus m_8$	m_7	m_{10}	m_{13}	m_0
$m_4 \oplus m_{13} \oplus m_6 \oplus m_{15}$	m_{12}	m_5	m_{14}	m_7	$m_8 \oplus m_1 \oplus m_{10} \oplus m_3$	m_0	m_9	m_2	m_{11}	$m_{12} \oplus m_5 \oplus m_{14} \oplus m_7$	m_4	m_{13}	m_6	m_{15}	$m_0 \oplus m_9 \oplus m_2 \oplus m_{11}$	m_8	m_1	m_{10}	m_3
$m_{15} \oplus m_{10} \oplus m_5 \oplus m_0$	m_7	m_2	m_{13}	m_8	$m_{11} \oplus m_6 \oplus m_1 \oplus m_{12}$	m_3	m_{14}	m_9	m_4	$m_7 \oplus m_2 \oplus m_{13} \oplus m_8$	m_{15}	m_{10}	m_5	m_0	$m_3 \oplus m_{14} \oplus m_9 \oplus m_4$	m_{11}	m_6	m_1	m_{12}

Table 3. Message expansion in HAS-160

State update: One compression function step is schematically described by Fig. 2 (a). The Boolean functions f used in each step are given by

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

where f_0 is used in steps 0-19, f_1 is used in steps 20-39 and 60-79 and f_2 is used in steps 40-59. The constant K_i that is added in each step changes every 20 steps, taking the values 0, $5a827999$, $6ed9eba1$ and $8f1bbcdc$. The rotational constant s_1^i is specified by the following table

i mod 20	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
s_1^i	5	11	7	15	6	13	8	14	7	12	9	11	8	15	6	12	9	14	5	13

The other rotational constant s_2^i changes only each 20 steps and $s_2^i \in \{10, 17, 25, 30\}$. According to the Davies-Meyer mode, the feedforward is applied and the output of the compression is

$$(A_{80} + A_0, B_{80} + B_0, C_{80} + C_0, D_{80} + D_0, E_{80} + E_0)$$

Alternative description of HAS-160: In Fig. 2 (b), the compression function is shown as a recurrence relation, where A_{i+1} plays the role of A in the usual step representation. Namely, A can be considered as the only new computed word, since the rotation that is applied to B can be compensated by properly adjusting the rotation constants in the recurrence relation specification. One starts from A_{-4}, A_{-3}, A_{-2}, A_{-1} and A_0, putting these

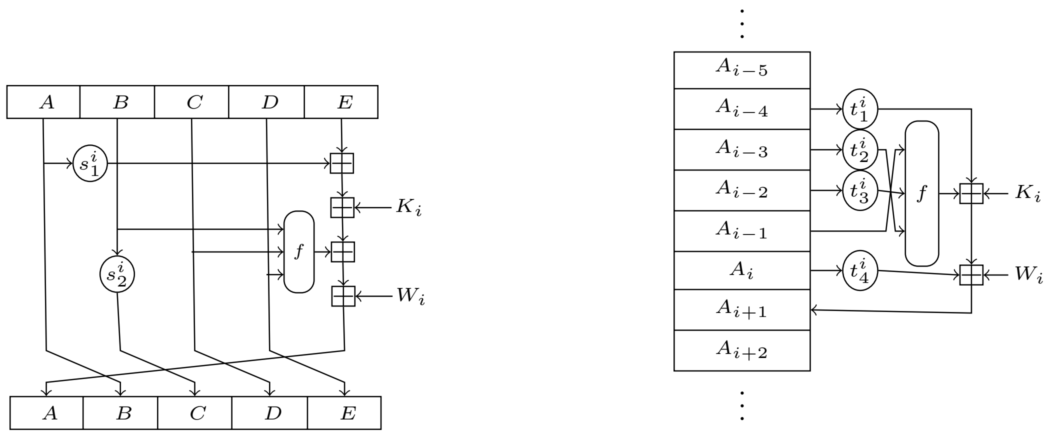

Fig. 2. Two equivalent representations of the state update

values to the previous chaining value (or the IV for the first message block) and computes the recurrence until A_{80} according to

$$A_{i+1} = A_{i-4} \lll t_1^i + K_i + f_i(A_{i-1}, A_{i-2} \lll t_3^i, A_{i-3} \lll t_2^i) + W_i + A_i \lll t_4^i \qquad (4)$$

The rotational values t_j^i, $1 \leq j \leq 4$ are derived from s_1^i and s_2^i, where the constants related to the rotation of B in the usual representation change around the steps $20 \times k$, $k = 0, 1, 2, 3$. For instance, to compute A_{42}, we have $t_1^{41} = 17$, $t_2^{41} = 17$, $t_3^{41} = 25$ and $t_4^{41} = 11$.

3 Compatible paths search heuristic and application to HAS-160

In this section, we provide a new search heuristic that can be used to find compatible paths in the boomerang setting. The particular colliding quartet found by applying the heuristic on HAS-160 is provided in Table 4.

Message quartet								
M_A	F6513317	810F1084	FFB71009	78CC955E	C3C09F18	5379FC99	435586DA	9C9AD3B4
	00440C80	E174316A	006D1670	2B5CF68A	AB3DE600	02C9E9D3	5FE95AFF	E351DE04
M_B	F6513317	810F1084	FFB71009	78CC955E	C3C09f18	5379FC99	435786DA	9C9AD3B4
	00440C80	E174316A	006D1670	2B5CF68A	AB3FE600	02C9E9D3	5FE95AFF	E351DE04
M_C	76513317	010F1084	FFB71009	78CC955E	43C09F18	5379FC99	435786DA	1C9AD3B4
	00440C80	E174316A	006D1670	2B5CF68A	AB3FE600	02C9E9D3	5FE95AFF	E351DE04
M_D	76513317	010F1084	FFB71009	78CC955E	43C09f18	5379FC99	435586DA	1C9AD3B4
	00440C80	E174316A	006D1670	2B5CF68A	AB3DE600	02C9E9D3	5FE95AFF	E351DE04
Chaining values quartet								
IV_A	1143BE75	9A9CA381	85B3F526	DA6ABE66	70EBE920			
IV_B	3AF7BD99	D08E2E63	245C2AF0	C4456954	CAC046EA			
IV_C	3AF7B599	D08E2E63	B45C2AF0	C425694C	3BE146F2			
IV_D	1143B675	9A9CA381	15B3F526	DA4ABE5E	E20CE928			

Table 4. Second order collision for the full HAS-160 compression function

The heuristic uses quartets of 1-bit conditions from Table 2 to keep track of the bit differences in each of the four compression function executions. Apart from the single-path propagations proposed in [5], two additional types of boomerang (cross-path) propagations are added. These boomerang propagations have been previously listed in [13].

To specify the problem on which the heuristic is applied in the context of HAS-160, the forward and backward message differentials are provided next. Let the forward message differential consist of a one-bit difference in messages m_6 and m_{12} and the backward differential of a one-bit difference in m_0, m_1, m_4 and m_7, as shown in Table 5. The particular bit-position of differences is left unspecified. The choice of these difference positions is justified by the following start/end points of the expanded message differences, expressed in terms of the notation used in Fig. 1: $(n_0, n_1, n_2, n_3, n_4, n_5) = (0, 8, 34, 53, 78, 80)$. It can be observed that the middle part consists of 20 steps.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
$m_8 \oplus m_9 \oplus m_{10} \oplus m_{11}$	$\textcircled{m_0}$	$\textcircled{m_1}$	m_2	m_3	$m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15}$	$\textcircled{m_4}$	m_5	m_6	$\textcircled{m_7}$	$\textcircled{m_0} \oplus \textcircled{m_1} \oplus m_2 \oplus m_3$	m_8	m_9	m_{10}	m_{11}	$\textcircled{m_4} \oplus m_5 \oplus m_6 \oplus \textcircled{m_7}$	m_{12}	m_{13}	m_{14}	m_{15}
$m_{11} \oplus m_{14} \oplus \textcircled{m_1} \oplus \textcircled{m_4}$	m_3	m_6	m_9	m_{12}	$\textcircled{m_7} \oplus m_{10} \oplus m_{13} \oplus \textcircled{m_0}$	m_{15}	m_2	m_5	m_8	$m_3 \oplus m_6 \oplus m_9 \oplus m_{12}$	m_{11}	m_{14}	$\textcircled{m_1}$	$\textcircled{m_4}$	$m_{15} \oplus m_2 \oplus m_5 \oplus m_8$	$\textcircled{m_7}$	m_{10}	m_{13}	$\textcircled{m_0}$
$m_4 \oplus m_{13} \oplus \boxed{m_6} \oplus m_{15}$	$\boxed{m_{12}}$	m_5	m_{14}	m_7	$m_8 \oplus m_1 \oplus m_{10} \oplus m_3$	m_0	m_9	m_2	m_{11}	$\boxed{m_{12}} \oplus m_5 \oplus m_{14} \oplus m_7$	m_4	m_{13}	$\boxed{m_6}$	m_{15}	$m_0 \oplus m_9 \oplus m_2 \oplus m_{11}$	m_8	m_1	m_{10}	m_3
$m_{15} \oplus m_{10} \oplus m_5 \oplus m_0$	m_7	m_2	m_{13}	m_8	$m_{11} \oplus \boxed{m_6} \oplus m_1 \oplus \boxed{m_{12}}$	m_3	m_{14}	m_9	m_4	$m_7 \oplus m_2 \oplus m_{13} \oplus m_8$	m_{15}	m_{10}	m_5	m_0	$m_3 \oplus m_{14} \oplus m_9 \oplus m_4$	m_{11}	$\boxed{m_6}$	m_1	$\boxed{m_{12}}$

Table 5. Message differentials. Backward: steps 0-39, forward: steps 40-79

Now, the particular problem schematically described by Fig. 1 (a) is represented more specifically by Table 7, where the backward and forward message differentials are indicated in the first and the last column, respectively. At this point, the only information that is present in the system is that the two paths end at the corresponding steps $n_2 = 34$ and $n_3 = 53$. The output of the heuristic in case of HAS-160 is given in Table 8. The full specifications of the two paths intersect on 5 steps, which is the number of inner state registers in HAS-160. Provided that the paths are compatible, one can now start from step 42 and apply the usual message modification technique to satisfy both paths, which resolves the middle of the boomerang as shown in Fig. 1 (a).

3.1 Search strategy

The approach consists of variating the position of the message difference bit, gradually extending the two paths, propagating the conditions in the quartet and backtracking in case of a contradiction. In more detail, the heuristic proceeds as follows:

(1) Randomize the positions of active bits in the active message words.
(2) Extend the specification of the forward/backward path backward/forward, respectively. Ensure that paths are randomized over different step invocations.
(3) Propagate all new conditions. In case of contradiction, backtrack
(4) If path are fully specified on a sufficient number of steps, return the two paths

In step (1), the message disturbance position in the two differentials is randomized to achieve variation in the paths. Alternatively, one position can be fixed to bit 31 and the other position randomized at each step invocation. As for step (2), at the point where the probability of contradiction between the two paths is negligible, one can extend paths simply by randomly sampling them in required steps and discarding non-narrow ones.

1.	`????` $\mapsto$ `--??`
2.	`??--` $\mapsto$ `----`
3.	`??xx` $\mapsto$ `--xx`
4.	`xx??` $\mapsto$ {`uu10`,`nn01`}
5.	`xx--` $\mapsto$ {`uu10`,`nn01`}
6.	`xxxx` $\mapsto$ {`unnu`,`nuun`}

1.	`????` $\mapsto$ `??--`
2.	`--??` $\mapsto$ `----`
3.	`xx??` $\mapsto$ `xx--`
4.	`??xx` $\mapsto$ {`01uu`,`10nn`}
5.	`--xx` $\mapsto$ {`01uu`, `10nn`}
6.	`xxxx` $\mapsto$ {`unnu`,`nuun`}

Table 6. Substitution rules: adding information to the forward path (left) and backward path (right)

Once the probability of contradiction becomes significant, substitute/backtrack strategy according to the Table 6 is applied to the remaining steps. In step (3), apart from propagations on a single path [5], quartet and quartet addition propagations (explained in Section 4) are applied. The heuristic ends when the full specification of two paths (containing only {`-`,`u`,`n`}) intersects on the number of words equal to the number of registers in the compression function inner state, as is the case in Table 8.

When new constraint information is to be added at a particular bit position, one can either add information to the forward path or to the backward path. Here, a clarification is necessary regarding the fact that in Table 8, four paths are shown, whereas the heuristic searches for a pair of paths (forward and backward path). This is due to the fact that the paths on the opposite faces of the boomerang are equal (up to 0 and 1 symbols) and thus one can consider a pair of paths. Nonetheless, the inner state of the search algorithm keeps all the four paths explicitly.

The substitutions provided in Table 6 represent generalizations of the substitutions used in [5]. The choice whether the information will be added to the forward or the backward path is made randomly each time. The left-hand and the right-hand tables correspond to adding constraints to the forward and the backward path, respectively. Consider for example rule `xx--` $\mapsto$ {`uu10`,`nn01`}. In this notation, the symbols `xx--` describe a bit position for which $\delta[A_i^j, B_i^j] =$ `x`, $\delta[D_i^j, C_i^j] =$ `x`, $\delta[B_i^j, C_i^j] =$ `-`, $\delta[A_i^j, D_i^j] =$ `-`. The rule simply substitutes the '`x`' symbol on the forward path by '`u`' or '`n`', while at the same time applying the immediate propagation of the '`-`' symbols to '`0`' and '`1`', respectively. This rule represents a generalization of the `x` $\mapsto$ {`u`,`n`} rule used in [5]. Other rules can be explained in a similar manner.

One possible variation of the general heuristic above is as follows. Once the two paths are sufficiently specified so that the contradictions are likely to occur, instead of adding new constraints randomly, a beneficial strategy is to introduce some graduality while extending the two paths. For example, one can choose a parameter k and extend both paths by only k steps. If the heuristic succeeds in extending the paths by k steps, reporting that there is no contradiction in the system, more steps can be attempted. If in the intermediate steps of the search, the path was in fact contradictory and this was not reported by 1-bit conditions, further attempts to extend or find the messages satisfying the paths will fail.

3.2 Application to HAS-160

In this section, we describe how the above heuristic can be applied in the case of HAS-160. First, we fix the position of the active bit in the backward differential to $b_1 = 31$. The following sequence of steps randomizes steps in the light-gray area in Table 7:

- Randomize the position of the forward message difference active bit b_2.
- With the message difference fully specified by b_1, b_2, sample narrow paths in the inner state words in steps denoted by light-gray in Table 7.
- Propagate conditions w.r.t. the three propagation types explained in Section 4. This step is applied repeatedly until none of the three propagation types can be applied on any of the bit positions.

Here, the path sampling is performed simply by initializing randomly the two instances of the path at the given step, calculating the recurrence over the required number of steps and extracting the path. If the Hamming weight of the path is greater than some pre-specified threshold, it is discarded and a new path is sampled. Using the above sampling of partial solution to the paths, the following procedure aims to find the full solution:

(1) Randomize steps in the light-gray area according to the procedure above (steps 43-49 and 34-37 in the forward and backward paths, respectively).
(2) Randomly choose (i, j), $0 \leq i \leq 31$, $38 \leq j \leq 42$, a position within the steps denoted by dark-grey in Table 7. If applicable, apply the substitution specified by Table 6. If not, choose another position. In case there is none, return the state.
(3) Propagate conditions and backtrack in case of contradiction. After a contradiction was reached a sufficient number of times, go to step (1).

After reducing the number of steps on which the two differentials meet from 5 to 3 (i.e., putting $k = 4$, where it should be noted that after the propagation the number of unconstrained bits will be relatively small), we received several paths reported as non-contradictory. At that point, there are two possible routes to verify the actual correctness of the intermediate result. One is to switch from 1-bit conditions to multi-bit conditions (such as 1.5-bit or 2.5-bit conditions [13]) that capture more information. ARXtools [13] can readily be used for this purpose. Each 2.5-bit verification using ARXtools for checking the compatibility of two paths took around 3-5 minutes. Another option is to continue with the search heuristic towards extending the specification of the paths to more steps, restarting always from the saved intermediate path state. As the knowledge in the system grows, the propagations turns a high proportion of bits into 0 and 1, which diminishes the possibility of contradiction. If the solution cannot be found after some time threshold t, the path can be abandoned. We experimented with both options above and concluded that both approaches are successful.

3.3 Full complexity of finding the HAS-160 second order collision

Our implementation of the heuristic found a correct pair of compatible paths in less than 5 days of execution on an 8-core Intel i7 CPU running at 2.67GHz. In more detail, as explained in Section 3.2, we ran the heuristic to search for paths that meet on 3 instead on 5 steps. It should be noted that due to many propagations, after the search stops, the resulting paths in fact have a small number of remaining unspecified bits in steps 38-42 (less than 32). The heuristic yielded around 8 solutions per day and among 40 returned path pairs, one turned out to be compatible and was successfully extended by one step more, as shown in Table 8.

The conditions for the two paths that are not explicitly given as `u,n,0,1` bits in Table 8 are provided in Tables 9 and 10. To find the quartet of message words and inner states that follow the two differentials in steps 34 to 49, inner state registers in step 42 are chosen to follow the conditions specified by Tables 9,10 and Table 8 and then the usual message modification procedure is applied backward and forward.

Once the middle steps of the quartet structure $n_2 = 34$ to $n_3 = 53$ are satisfied, the second order collision property extends to steps $n_1 = 8$ to $n_4 = 78$ with probability 1 (see Fig. 1 (b)). To cover all of the compression function steps, the middle steps are kept constant and the remaining ones are randomized until the second order collision property is satisfied. In particular, if m_6 and m_{15} are randomized while $m_6 \oplus m_{15}$ is kept constant, according to the message expansion specification, the inner state will be randomized for $54 \leq i \leq 80$ and $0 \leq i \leq 35$. Similarly, if m_6 and m_4 are randomized where $m_6 \oplus m_4$ is kept constant, the randomization will happen for $52 \leq i \leq 79$ and $0 \leq i \leq 34$. Here, a small penalty in probability is paid due to the fact that the paths may be corrupted towards the start/end points. The two mentioned randomizations provide around 64 bits of freedom.

The probability that one randomization explained above yields a second order collision can be bounded from below by p^2q^2, where p and q are the probabilities of two selected sparse differentials in steps $0 \leq i \leq n_1$ and $n_4 \leq i < 80$, respectively. By counting the number of conditions in sparse paths that happened in the quartet in Table 4, we obtain $p = 2^{-22}$ and $q = 2^{-3}$ and the probability lower bound $p^2q^2 = 2^{-50}$. The actual time of execution on the above mentioned PC was less than two days, due to the additional differential paths which contribute to the exact probability of achieving the second order collision property (previously named amplified probability [3, 13]).

4 Details on condition propagation

The heuristic keeps track of the current state of the system by keeping the following information in memory:

- Four differential path tables keeping the current state of bit-conditions
- $4 \times r$ carry graphs [22] (one carry graph for each of four paths consisting of r steps)

In our implementation, we used $r = 16$, keeping the information about steps 33-48. The carry graphs model the carry transitions allowed by the knowledge present in the system. Below, the three types of knowledge propagation are described. The propagations are applied as long as the system is not fully propagated with respect to all three types below.

4.1 Single-path propagations

An explicit example of a single-path propagation [5] (see also [22, 24]) is provided below. The constraints and the corresponding carry graphs for at a particular bit position are all explicitly shown. The new propagated constraints as well as the removed carry graph edges are indicated.

Throughout the compression function execution specified by (4), for any $1 \leq i \leq 80$ and $0 \leq j \leq 31$, bit A_i^j is computed based on the 5 input bits in A_{i-j}, $1 \leq j \leq 5$, the

...

δK	`01101110110110011110101110100001`
$\delta[W_{B,41}, W_{C,41}]$	`--------0-----0-----------------`
$\delta[B_{37}, C_{37}]$	`11-0-u---00u-10n---10-0n1un-----`
$\delta[B_{38}, C_{38}]$	`0u1000-100111uu011-0n11u0000-u1-`
$\delta[B_{39}, C_{39}]$	`01un-n0u010u-0-u00u-1---n0u-un00`
$\delta[B_{40}, C_{40}]$	`1-nu001uu01-0n--u01-1-0-u0--11-1`
$\delta[B_{41}, C_{41}]$	`1--n--00--0-u1--u--u1--001-0---1`
$\delta[B_{42}, C_{42}]$	`1---n-u1uun-n1u--00n0nn--0n0---n`

δK	`01101110110110011110101110100001`
$\delta[W_{B,41}, W_{C,41}]$	`--------0-----0-----------------`
$\delta[B_{37}, C_{37}]$	`11-0-u---00u-10n---10-0n1un-----`
$\delta[B_{38}, C_{38}]$	`0u1000-100111uu011-0n11u0000-u1-`
$\delta[B_{39}, C_{39}]$	`01un-n0u010u-0-u00u-1---n0u-un00`
$\delta[B_{40}, C_{40}]$	`1-nu001uu01-0n--u01-1-0-u0--11-1`
$\delta[B_{41}, C_{41}]$	`1--n--00--0-u1--u--u1--001-0---1`
$\delta[B_{42}, C_{42}]$	`1---n-u1uun-n1u--00n0nn--0n0---n`

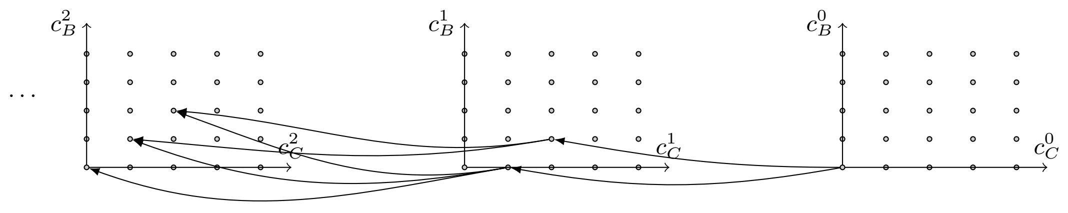

Fig. 3. Extract of single-path path constraints

message word bit as well as a particular constant bit. Moreover, bit A_i^j depends on the carries coming from the computations at bit positions $j < k \leq 0$.

In Fig. 3, an extract of the path is provided, borrowed from the $\Delta[B, C]$ path in Table 8. The bit positions treated in this case are $\delta[B_{42}^1, C_{42}^1]$ (left) and $\delta[B_{42}^0, C_{42}^0]$ (right). The shaded bits are the bit positions participating in the computation of the two bits. As for the carry graph, it consists of 32 subgraphs, each comprising of 5×5 nodes. In Fig. 3, only the subgraphs corresponding to bit positions 1 (left) and 0 (right) are shown. Each subgraph node represents a particular carry configuration at the particular bit position. Due to the fact that there is 5 summands in (4), the carry value is limited to $\{0, \ldots 4\}$ and thus each subgraph contains 5×5 nodes. The edges in the graphs represent possible carry configuration transitions from bit position i to $i + 1$.

Next, the edges connecting subgraphs for bit positions $i = 0$ to $i = 1$ in Fig. 3 are explained. The shown edges and the corresponding bit-conditions are aligned in the sense that there is no possible propagations at the particular positions neither from the bit-conditions to graphs nor vice-versa. According to the bit-conditions on position 0, we have

$$c_B^1 | B_{42}^0 = c_B^1 | 1 = 1 + W_{B,41}^0 + B_{37}^{15} + f_2(1, 1, 1) + 0 = 1 + W_{B,41}^0 + B_{37}^{15}$$
$$c_C^1 | C_{42}^0 = c_C^1 | 0 = 1 + W_{C,41}^0 + C_{37}^{15} + f_2(1, 0, 1) + 0 = 1 + W_{C,41}^0 + C_{37}^{15} + 1$$

From the above two equalities, it follows that $W_{B,41}^0 = B_{37}^{15}$ and $W_{C,41}^0 = C_{37}^{15}$. Since $\delta[W_{B,41}^0, W_{C,41}^0]$ and $\delta[B_{37}^{15}, C_{37}^{15}]$ are set to `-`, the possible carry configurations are $(c_B^1, c_C^1) \in \{(0, 1), (1, 2)\}$, which corresponds to the two edges between the two subgraphs.

Whenever it is possible to deduce new information from what is already present in the system, propagations need to be carried out until no new information can be derived. Continuing with the setting in Fig. 3, assume that during the heuristic, the symbol `-` at position $\delta[W_{B,41}^0, W_{C,41}^0]$ is substituted by `0`. Then, the propagation at this bit consists of substituting `-` at position $\delta[B_{37}^{15}, C_{37}^{15}]$ by `0` and deleting the $(0, 0) \mapsto (1, 2)$ graph edge. The edge deletion continues to the left and to the right. In case of Fig. 3, this amounts to deleting the edges coming out of node $(1, 2)$ and continuing in the same manner throughout the rest of the subgraphs. Next, all of the influenced bit positions, either through carry graphs or through bit-conditions, need to be repropagated similarly to the process described above.

4.2 Quartet propagations

This type of propagations is the simplest of all three types presented in this section, since it does not involve the carry graphs. An example of this type of propagation is as follows. Let (i, j) denote a specific bit position in the range of the considered steps. Let the bit-conditions $\delta[A_i^j, B_i^j]$, $\delta[D_i^j, C_i^j]$, $\delta[B_i^j, C_i^j]$, $\delta[A_i^j, D_i^j]$ in the four paths be equal to `u`, `x`, `-`, and `?`, respectively. It follows that $A_i^j = 0$, $B_i^j = 1$, $C_i^j = 1$ and $D_i^j = 0$ and thus the quartet can be readily substituted by a new one

$$\texttt{(ux-?)} \mapsto \texttt{(uu10)}$$

Given a quartet of conditions, the substitution quartet is found by going through all the bit value quartets that satisfy the given condition quartet. The new quartet consists of the symbols from Table 2 that represent minimal sets contain the valid bit value pairs.

4.3 Quartet addition propagations

In this subsection, the following terminology is adopted: carry subgraphs as shown in Fig. 3 are called *2-graphs*. Nodes with at least one input/output edge in the 2-graphs are called *active* nodes. During the execution of the heuristic, each active 2-graph node corresponds to a possible carry configuration that has not yet been ruled out by the heuristic.

Quartet addition propagation is illustrated in Fig. 4. The four graphs in the top part represent a particular case of the 2-graphs that correspond to a single bit position (i, j) on paths $[A, B]$, $[B, C]$, $[D, C]$, $[A, D]$, respectively from left to right. The active nodes are circled and the information about the number of input/output edges is abstracted from the picture. The quartet addition propagation is based on the fact that the four different 2-graphs may impose incompatible constraint on the carry configurations at the considered bit position. For instance, according to the 2-graph corresponding to the path $[D, C]$ (third graph from the left in Fig. 4), since node $(c_D, c_C) = (3, 2)$ is active, it follows that having a carry equal to 3 at this bit position in the branch D is not ruled out. However, since there is no active nodes in the third column of the (c_A, c_D) graph, the node $(c_D, c_C) = (3, 2)$ should be deactivated.

For the purpose of deciding which 2-carry graph nodes should be deactivated, it is convenient to introduce another type of carry graphs that will be called *4-carry* graphs. For each bit-position covered by the heuristic, the four 2-carry graphs are represented as one 4-carry graph, as shown in the bottom part of Fig. 4. The 4-carry graphs abstract the information about active nodes in the 2-carry graphs.

As shown in Fig. 4, the 4-carry graph has four groups of nodes that simply represent the carry values c_A, c_B, c_C and c_D, respectively. The edges in the 4-carry graph are constructed simply by mapping the active nodes in the corresponding 2-carry graphs to the edges between the corresponding node groups. This mapping is specified by an example as follows. The active nodes in the (c_A, c_D) 2-carry graph are $(0, 0)$ and $(2, 1)$. This is translated to

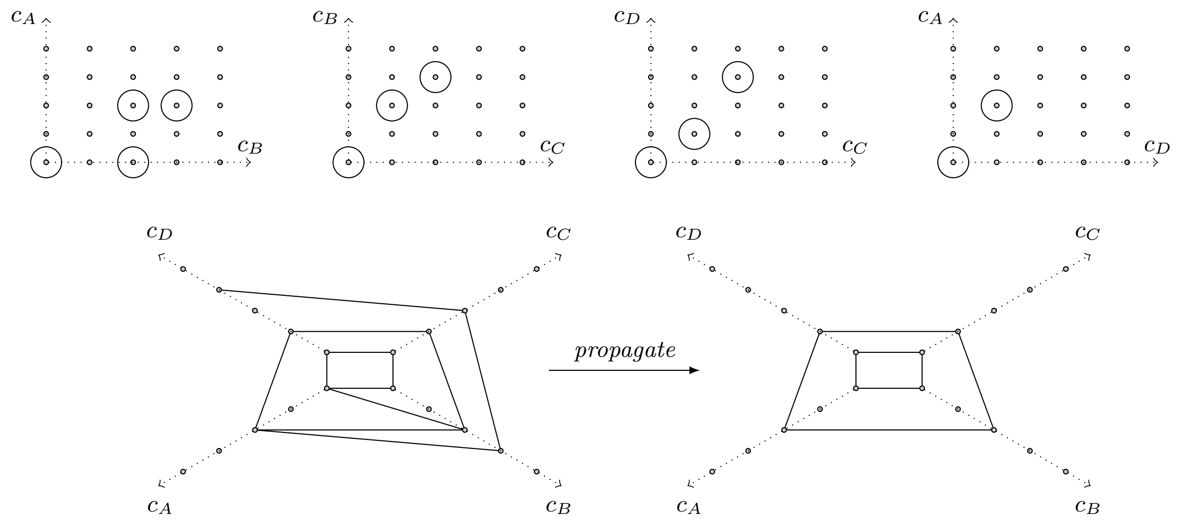

Fig. 4. Example: 2-carry graphs and the corresponding 4-carry graph before and after propagation

the edges $(0, 0)$ and $(2, 1)$ between the c_A and c_D branches in the 4-carry graph. The other three 2-carry graph active nodes are mapped to the edges analogously.

The 4-carry graph representation allows expressing the quartet addition propagation rules in a natural way. For that purpose, let a *cycle* denote a closed path connecting four nodes, where no two nodes are members of the same node group in the 4-graph. The propagation rules are then as follows:

(R1) Remove all "dead-end" edges, i.e., the ones with an end node of degree 1
(R2) Remove all edges that do not participate in any cycle

In the case of the propagation given in Fig. 4, the quartet addition propagation consisted of three applications of (R1) and one application of (R2). Since each 4-graph edge corresponds to a node in the corresponding 2-graph, the edge removal according to rules (R1) and (R2) amounts to deactivating the corresponding nodes in the 2-graph. The node deactivation is done by deleting all input and output edges for the corresponding 2-graph node. In the case of our HAS-160 search, implementing only rule (R1) turned out to be sufficient.

5 Conclusion

We proposed a heuristic for searching for compatible differential paths and applied it to HAS-160. Instead of working with 0/1 bit values, we used the reasoning on sets of bits described by 1-bit constraints. The three types of propagations used during the search (single-path propagations, quartet propagations and quartet addition propagations) are explained through particular examples. Using the 1-bit constraints along with these propagations yielded an acceptable rate of false positives and the second order collision was successfully found. One possible future research direction is to evaluate the performance of the proposed heuristic in case of SHA-2 with a goal of improving the attack [3] and to assess the impact of high rate of contradictory paths reported in [18] in this context.

Acknowledgments. The authors would like to thank Gaëtan Leurent for his help related to ARXtools and the discussions on the topic.

References

1. *Telecommunications Technology Association. Hash Function Standard Part 2, Hash Function Algorithm Standard (HAS-160), TTAS.KO-12.0011/R1, 2008.*
2. Jean-Philippe Aumasson. Zero-sum distinguishers, Rump session talk at CHES, 2009. http://131002.net/data/talks/zerosum_rump.pdf.
3. Alex Biryukov, Mario Lamberger, Florian Mendel, and Ivica Nikolic. Second-order differential collisions for reduced SHA-256. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 270–287. Springer, 2011.
4. Alex Biryukov, Ivica Nikolić, and Arnab Roy. Boomerang attacks on BLAKE-32. In Antoine Joux, editor, *FSE*, volume 6733 of *Lecture Notes in Computer Science*, pages 218–237. Springer, 2011.
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 *Lecture Notes in Computer Science*, pages 1–20. Springer, 2006.
6. 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 *Lecture Notes in Computer Science*, pages 286–295. Springer, 2006.
7. Pierre-Alain Fouque, Gaëtan Leurent, and Phong Q. Nguyen. Automatic search of differential path in MD4. *IACR Cryptology ePrint Archive*, 2007:206, 2007.
8. 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.
9. Dmitry Khovratovich. Bicliques for permutations: Collision and preimage attacks in stronger settings. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 544–561. Springer, 2012.
10. Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.
11. Aleksandar Kircanski, Yanzhao Shen, Gaoli Wang, and Amr M. Youssef. Boomerang and slide-rotational analysis of the SM3 hash function. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 304–320. Springer, 2012.
12. Mario Lamberger and Florian Mendel. Higher-order differential attack on reduced SHA-256. *IACR Cryptology ePrint Archive*, 2011:37, 2011.
13. Gaëtan Leurent. Analysis of differential attacks in ARX constructions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2012.
14. Gaëtan Leurent. Construction of differential characteristics in ARX designs - application to Skein. *IACR Cryptology ePrint Archive*, 2012:668, 2012.
15. Gaëtan Leurent and Arnab Roy. Boomerang attacks on hash function using auxiliary differentials. In Orr Dunkelman, editor, *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 2012.
16. Florian Mendel and Tomislav Nad. Boomerang distinguisher for the SIMD-512 compression function. In Daniel J. Bernstein and Sanjit Chatterjee, editors, *INDOCRYPT*, volume 7107 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2011.
17. Florian Mendel, Tomislav Nad, and Martin Schläffer. Cryptanalysis of round-reduced HAS-160. In Howon Kim, editor, *ICISC*, volume 7259 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2011.
18. 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*, volume 7073 of *Lecture Notes in Computer Science*, pages 288–307. Springer, 2011.
19. Florian Mendel, Tomislav Nad, and Martin Schläffer. Collision attacks on the reduced dual-stream hash function RIPEMD-128. In Anne Canteaut, editor, *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2012.
20. Florian Mendel, Tomislav Nad, and Martin Schläffer. Finding collisions for round-reduced SM3. In Ed Dawson, editor, *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 174–188. Springer, 2013.

21. 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 *Lecture Notes in Computer Science*, pages 324–334. Springer, 2007.
22. Nicky Mouha, Christophe De Cannière, Sebastiaan Indesteege, and Bart Preneel. Finding collisions for a 45-step simplified HAS-V. In Heung Youl Youm and Moti Yung, editors, *WISA*, volume 5932 of *Lecture Notes in Computer Science*, pages 206–225. Springer, 2009.
23. Sean Murphy. The return of the cryptographic boomerang. *IEEE Transactions on Information Theory*, 57(4):2517–2521, 2011.
24. Thomas Peyrin. Analyse de fonctions de hachage cryptographes. Ph.D. Thesis, University of Versailles, 2008., `http://www.iacr.org/phds/?p=detail&entry=500`.
25. Yu Sasaki. Boomerang distinguishers on MD4-family: First practical results on full 5-pass haval. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.
26. 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 *Lecture Notes in Computer Science*, pages 302–317. Springer, 2008.
27. Yu Sasaki and Lei Wang. Distinguishers beyond three rounds of the RIPEMD-128/-160 compression functions. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS*, volume 7341 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2012.
28. Yu Sasaki, Lei Wang, Yasuhiro Takasaki, Kazuo Sakiyama, and Kazuo Ohta. Boomerang distinguishers for full HAS-160 compression function. In Goichiro Hanaoka and Toshihiro Yamauchi, editors, *IWSEC*, volume 7631 of *Lecture Notes in Computer Science*, pages 156–169. Springer, 2012.
29. Martin Schläffer and Elisabeth Oswald. Searching for differential paths in MD4. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 242–261. Springer, 2006.
30. Marc Stevens, Arjen K. Lenstra, and Benne de Weger. Chosen-prefix collisions for MD5 and colliding x.509 certificates for different identities. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2007.
31. David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *FSE*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 1999.
32. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.
33. Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.
34. Hongbo Yu, Jiazhe Chen, and Xiaoyun Wang. The boomerang attacks on the round-reduced Skein-512. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2012.
35. 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 *Lecture Notes in Computer Science*, pages 146–155. Springer, 2005.

step	$\Delta[A,B]$	$\Delta[D,C]$	$\Delta[B,C]$	$\Delta[A,D]$	step
9	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	$\textcircled{9}$
10	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	10
11	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	11
12	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	12
13	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	13
⋮	⋮	⋮	⋮ [NO DIFFERENCE]	⋮	⋮
29	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	29
30	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	30
31	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	31
32	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	32
33	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	$\textcircled{33}$
34	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{34}$
35	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	35
36	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{36}$
37	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	37
38	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	38
39	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{39}$
$\boxed{40}$	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{40}$
$\boxed{41}$	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	41
42	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	42
43	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	43
44	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{44}$
45	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{45}$
46	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	$\textcircled{46}$
47	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	47
48	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	48
49	????????????????????????????????	????????????????????????????????	????????????????????????????????	????????????????????????????????	49
$\boxed{50}$	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	$\textcircled{50}$
51	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	$\textcircled{51}$
52	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	52
$\boxed{53}$	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	53
54	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	54
⋮	⋮ [NO DIFFERENCE]	⋮	⋮	⋮	⋮
76	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	76
$\boxed{77}$	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	77

Table 7. Input for the search heuristic

step	$\Delta[A,B]$	$\Delta[D,C]$	$\Delta[B,C]$	$\Delta[A,D]$	step
29	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	29
30	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	30
31	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	31
32	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	32
33	????????????????????????????????	????????????????????????????????	--------------------------------	--------------------------------	$\textcircled{33}$
34	0???????????????????????????????	1???????????????????????????????	u-------------------------------	u-------------------------------	$\textcircled{34}$
35	0??????????u???????x0???x-0?????	1??????????u???????x0???x-1?????	u----------1--------0-----u-----	u----------0--------0-----u-----	35
36	1x????????xu?-01B?--0Bx--u0D????	0x????????xu?-11B?--1Bx--u0D????	n----------1--u1----u----10-----	n----------0--u1----u----00-----	$\textcircled{36}$
37	11-0D0B??0n0?101-x-10-01u01C???x	11-0D1B??0n1?100-x-10-00u10C???x	11-0-u---00u-10n---10-0n1un-----	11-0-u---01u-10n---10-0n0un1----	37
38	00u0nn-1n01uu000uu-011u00nnn-01-	01u0nn-1n01uu110uu-001u10nnn-11-	0u1000-100111uu011-0n11u0000-u1-	0u0011-110100uu000-0n10u0111-u1-	38
39	n101-1000100-0-0000-1---100-010n	n110-0010101-0-1001-1---001-100n	01un-n0u010u-0-u00u-1---n0u-un00	11un-n0u010u-0-u00u-1---n0u-un01	$\textcircled{39}$
$\boxed{40}$	1-100010001-01--0n1-u-0-00--11-1	1-010011101-00--1n1-u-0-10--11-1	1-nu001uu01-0n--u01-1-0-u0--11-1	1-nu001uu01-0n--u11-0-0-u0--11-1	$\textcircled{40}$
$\boxed{41}$	u--1--00--0-01--0--0u--001-0---1	u--0--00--0-11--1--1u--001-0---1	1--n--00--0-u1--u--u1--001-0---1	0--n--00--0-u1--u--u0--001-0---1	41
42	u---1-01001-110--n01011--n10---1	u---0-11110-011--n00000--n00---0	1---n-u1uun-n1u--00n0nn--0n0---n	0---n-u1uun-n1u--10n0nn--1n0---n	42
43	n------01----0------u------00-un	n------00----0------u------01-un	0????--0nD???0x?????1x??x--0u-10	1????--0nD???0x?????0x??x--0u-01	43
44	0-----10----------------1u------	0------0----------------1u------	0?????C0????????????????11?????x	0?????C0????????????????10?????x	$\textcircled{44}$
45	------00------------u------1----	------00------------u------1----	??????00????????????1??????1????	??????00????????????0??????1????	$\textcircled{45}$
46	u-------------------------------	u-------------------------------	1???????????????????????????????	0???????????????????????????????	$\textcircled{46}$
47	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	47
48	-------u------------------------	-------u------------------------	???????1????????????????????????	???????0????????????????????????	48
49	-------n------------------------	-------n------------------------	???????0????????????????????????	???????1????????????????????????	49
$\boxed{50}$	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	$\textcircled{50}$
51	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	$\textcircled{51}$
52	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	52
$\boxed{53}$	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	53
54	--------------------------------	--------------------------------	????????????????????????????????	????????????????????????????????	54

Table 8. Output of the heuristic: compatible paths for HAS-160

Step	Conditions
33	$A_{33,14} \neq A_{32,14}$
34	$A_{34,20} = A_{33,20}$
35	$A_{35,0} \neq A_{34,0}$, $A_{35,16} \neq A_{33,31}$, $A_{35,26} \neq A_{34,26}$
36	$A_{36,3} = A_{35,3}$, $A_{36,9} \neq A_{35,9}$, $A_{36,21} = A_{35,21}$, $A_{36,22} = A_{34,5}$, $A_{36,23} = A_{35,23}$
37	$A_{37,0} = A_{36,0}$, $A_{37,1} = A_{36,1}$, $A_{37,2} \neq A_{35,17}$, $A_{37,13} = A_{36,13}$, $A_{37,23} \neq A_{36,23}$
38	$A_{38,25} = A_{36,8}$
39	$A_{39,19} \vee \overline{A}_{37,2} = 1$
40	$A_{40,17} \vee \overline{A}_{38,0} = 1$, $A_{40,30} \vee \overline{A}_{38,13} = 1$
41	$A_{41,16} \vee \overline{A}_{39,23} = 1$

Table 9. Backward differential conditions not shown in Table 8

Step	Conditions
37	$A_{37,2} = A_{36,2}$, $A_{37,3} \neq A_{36,3}$, $A_{37,10} \neq A_{36,10}$, $A_{37,13} = A_{36,28}$, $A_{37,15} = 0$, $A_{37,25} = A_{36,8}$, $A_{37,29} = A_{36,12}$
38	$A_{38,0} = 1$
39	$A_{39,4} = 1$, $A_{39,8} = 0$, $A_{39,9} = 1$, $A_{39,12} = 0$, $A_{39,17} = 0$, $A_{39,19} = 1$
40	$A_{40,4} = 0$, $A_{40,5} = 0$, $A_{40,8} = 0$, $A_{40,12} = 1$
41	$A_{41,13} = 0$, $A_{41,14} = 0$
42	$A_{42,7} = 0$,
43	$A_{43,6} = 0$, $A_{43,7} \vee \overline{A}_{41,14} = 1$
44	$A_{44,0} = 0$, $A_{44,1} = 0$, $A_{44,4} \vee \overline{A}_{42,11} = 1$, $A_{44,26} \vee \overline{A}_{42,1} = 1$
45	$A_{45,26} = 0$
46	$A_{46,4} \vee \overline{A}_{44,11} = 1$
47	$A_{47,4} = 1$, $A_{47,24} \vee \overline{A}_{45,31} = 1$, $A_{47,31} = 1$
48	$A_{48,31} = 0$
49	$A_{49,17} = 0$
50	$A_{50,17} = 0$, $A_{50,24} = 1$
51	$A_{51,17} = 0$

Table 10. Forward differential conditions not shown in Table 8

step	$\Delta[W_A, W_B]$	$\Delta[W_D, W_C]$	$\Delta[W_B, W_C]$	$\Delta[W_A, W_D]$
33	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
34	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
35	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
36	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
37	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
38	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
39	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
40	`0-------------u-----------------`	`1-------------u-----------------`	`u-------------1-----------------`	`u-------------0-----------------`
41	`--------0-----u-----------------`	`--------0-----u-----------------`	`--------0-----1-----------------`	`--------0-----0-----------------`
42	`-------------------------------1`	`-------------------------------1`	`-------------------------------1`	`-------------------------------1`
43	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
44	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
45	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
46	`1-------------------------------`	`0-------------------------------`	`n-------------------------------`	`n-------------------------------`
47	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`
48	`--------------------------------`	`--------------------------------`	`--------------------------------`	`--------------------------------`

Table 11. Message differences after propagation