Finding Collisions for a 45-Step Simplified HAS-V*

Nicky Mouha[1,2,**], Christophe De Cannière[1,2,***], Sebastiaan Indesteege[1,2,†], and Bart Preneel[1,2]

[1] Department of Electrical Engineering ESAT/SCD-COSIC, Katholieke Universiteit Leuven. Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
[2] Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.

Abstract. Recent attacks on hash functions start by constructing a differential characteristic. By finding message pairs that satisfy this characteristic, a collision can be found. This paper describes the method of De Cannière and Rechberger to construct generalized characteristics for SHA-1 in more detail. This method is further generalized and applied to a simplified variant of the HAS-V hash function. Using these techniques, a characteristic for 45 steps is found, requiring an effort of about 2^{46} compression function evaluations to find a colliding message pair. A lot of the message bits can still be freely chosen when using this characteristic, greatly increasing its usefulness.

Keywords: Cryptanalysis, hash function, HAS-V, collision.

1 Introduction

Hash functions are an important building block in cryptography. As described in [1], these are functions h that convert an input m of arbitrary length in a deterministic way to a fixed-length output $h(m)$. It is crucial that a number of security properties are satisfied, one of which is the infeasibility of finding collisions (collision resistance). A collision consists of two input values m, m' where $m \neq m'$ for which $h(m) = h(m')$.

Recent attacks by Wang et al. on the widely used hash functions MD4 [2], MD5 [3], RIPEMD [2] and SHA-1 [4], as well as other hash functions, show that it is possible to find collisions for these hash functions much faster than expected by the birthday paradox [1]. In response to these attacks, NIST has launched a competition to find a new hash function standard [5]. Although a lot of cryptanalysis effort is directed to these submissions, we feel that it is still very important to analyze existing hash function standards.

These recent collisions have lead to several practical attacks on network applications. For POP3 [6], it is is possible to mount a password recovery attack. Together with IMAP [7], POP3 [8] is one of the most used protocols for retrieving e-mail. Very recently, a rogue CA certificate was created using a collision for MD5 [9,10]. This certificate allows an attacker to impersonate any website on the Internet secured by HTTPS [11], including websites for banking and e-commerce.

The hash function HAS-V [12] is similar in structure to these hash functions. HAS-V fulfills the need of the KCDSA [13], the Korea Certificate-based Digital Signature Algorithm, to use a

* This work was supported in part by the IAP Program P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT program under contract ICT-2007-216676 ECRYPT II.

** This author is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

*** Postdoctoral Fellow of the Research Foundation – Flanders (FWO)

† F.W.O. Research Assistant, Fund for Scientific Research – Flanders (Belgium).

hash function with a variable digest size. The only cryptanalytic results on HAS-V known to us are described in [14]. Results using the recent attacks on hash functions have not been published before. The cryptanalysis of a simplified variant of HAS-V is the subject of this paper.

Recent attacks on hash functions focus on the construction of a differential characteristic, that allows collisions to be found with a good probability by finding messages m, m' that satisfy this characteristic. Characteristics are often constructed in an ad hoc way, which does not give any insight into the application of these attacks to other hash functions. This emphasizes the need for automated methods.

One such method, introduced in [15], is further generalized and applied to the simplified HAS-V. Using this method, we found a characteristic for a 45-step collision with an expected work factor of $2^{75.84}$ step function evaluations, which is given in Table 12. Further improvements lead to the better characteristic shown in Table 13, which has a work factor of $2^{51.53}$, making a collision finding attack feasible. If the cost of one step function evaluation is about 2^{-5} compression function evaluations, these work factors are equivalent to about 2^{71} and 2^{46} compression function evaluations, respectively. Note that a lot of bits in the message words can still be freely chosen.

Notation is defined in Table 1. In Sect. 2, a description of a simplified variant of HAS-V is given. An alternative, cyclic description of this hash function is provided as well. The technique for finding NL-characteristics of [15] is further explained, generalized and applied to HAS-V in Sect. 3. Techniques for improving NL-characteristics are laid out in Sect. 4, where good NL-characteristics for a 45-step simplified HAS-V are obtained as well. A conclusion and suggestions for future work are given in Sect. 5.

Appendix A lists the NL-characteristics we obtained. To assist the reader in understanding the more abstract explanation of the graph method in this paper, a simple example is given in Appendix B. Although this method is extensively used to attack SHA-1 in [15], this paper is the first to fully explain it.

2 A Simplified HAS-V

The hash function HAS-V [12] splits a 1024-bit message block into two 512-bit message blocks, which are then processed in two streams. The rounds of each stream alternately use message words of the first and the second 512-bit block. In our simplified variant of HAS-V, the right stream is omitted, as well as rounds in the left stream that depend on message words of the second 512-bit message block. As recent collision finding attacks are applied to hash functions with only one stream, simplifying the hash function in this way allows us to focus more easily on the main concepts of these recent attacks. For the same reason, the optional output tailoring is not applied. All other properties of HAS-V are left intact. A description of this simplified HAS-V is now given.

2.1 Description

The input message is padded and split into 512-bit message blocks. A 3-round compression function with 20 steps per round is applied to each of these 512-bit message blocks. This compression function $g(m, h)$ uses a 160-bit chaining input h and a 512-bit message block m as its inputs. The chaining input h_{n+1} of the next call of the compression function is calculated as $h_n + g(m, h_n)$. Here, the addition is done in blocks of 32-bit words, using a total of five adders modulo 2^{32}. The chaining variables for the first compression function call are set to fixed values, referred to as the IV. They are shown in Table 2. The last chaining input h represents the hash value.

Given a 512-bit message block m, consisting of 16 32-bit message words m_i, four extra message words, referred to as XOR-words, are derived from these message words for every round, as specified in Table 3. The extended message words w_i consist of the message words m_i followed by the four XOR-words.

Table 4 shows how the expanded message words W_t are derived as a reordering of the extended message words w_i for every round.

Figure 1 gives a schematic representation of the HAS-V step function, which is also described by

$$\begin{cases} A_{t+1} \leftarrow (A_t \lll S_t) + f_j(B_t, C_t, D_t, E_t) + W_t + K_t \ , \\ B_{t+1} \leftarrow A_t \ , \\ C_{t+1} \leftarrow B_t \ggg 2 \ , \\ D_{t+1} \leftarrow C_t \ , \\ E_{t+1} \leftarrow D_t \ . \end{cases} \tag{1}$$

Here, f_j represents a Boolean function, different for every round j:

$$\begin{aligned} f_1(B,C,D,E) &\triangleq (B \wedge C) \oplus (\neg B \wedge D) \oplus (C \wedge E) \oplus (D \wedge E) \ , \\ f_2(B,C,D,E) &\triangleq (B \wedge C) \oplus (\neg B \wedge E) \oplus D \ , \\ f_3(B,C,D,E) &\triangleq (\neg B \wedge C) \oplus (B \wedge D) \oplus (C \wedge E) \oplus (D \wedge E) \ . \end{aligned} \tag{2}$$

In every step a constant K_t, different for every round, is added. These are listed in Table 5. The rotation value S_t is different for every step of a round. They are given in Table 6.

2.2 Cyclic Description

Using the step function of the simplified HAS-V, five internal variables A_t, B_t, C_t, D_t and E_t are obtained from five previous internal variables, A_{t-1}, B_{t-1}, C_{t-1}, D_{t-1} and E_{t-1}. As $B_t \equiv A_{t-1}$, $C_t \equiv A_{t-2} \ggg 2$, $D_t \equiv A_{t-3} \ggg 2$ and $E_t \equiv A_{t-4} \ggg 2$, it is sufficient to keep track of only A_t when calculating the step functions. These A_t, preceded by the IV values, are denoted by Q_t. A similar cyclic formulation was proposed for MD5 in [16], however there Q_t refer to values of B_t. The compression function can then be formulated alternatively for $t = 0, \ldots, 59$ as:

$$Q_{t+1} \leftarrow (Q_t \lll S_i) + f_j(Q_{t-1}, Q_{t-2} \ggg 2, Q_{t-3} \ggg 2, Q_{t-4} \ggg 2) + W_t + K_t \ . \tag{3}$$

The values of Q_t for $t = -4, \ldots 0$ are derived from the IV:

$$(Q_{-4}, Q_{-3}, Q_{-2}, Q_{-1}, Q_0) \leftarrow (E \lll 2, D \lll 2, C \lll 2, B, A) \ . \tag{4}$$

It is this cyclic formulation of the simplified HAS-V that will be used from now on in this text.

3 NL-characteristics

For collision attacks, non-linear characteristics (NL-characteristics) start with chaining input and output difference zero. Differences are introduced via the message input, which then cancel themselves out with a sufficiently high probability by following the characteristic. High-probability characteristics are crucial for building fast collision-finding attacks, yet not much is known about their

Table 1. Notation.

notation	description
$x \parallel y$	concatenation of the binary strings x and y
$x \wedge y$	bitwise AND of x and y
$x \vee y$	bitwise OR of x and y
$x \oplus y$	bitwise XOR of x and y
$\neg x$	bitwise NOT of x
$x \lll s$	rotation of x to the left by s positions
$x \ggg s$	rotation of x to the right by s positions
$x + y$	addition of x and y modulo 2^{32} (in text)
$x \boxplus y$	addition of x and y modulo 2^{32} (in figures)
$x[i]$	bit selection: 0 if $(x \wedge 2^i) \equiv 0$, 1 otherwise

Table 2. The IV values for the simplified HAS-V.

	A	B	C	D	E
IV	0x67452301	0xEFCDAB89	0x98BADCFE	0x10325476	0xC3D2E1F0

Table 3. Calculation of the XOR-words for the simplified HAS-V.

	w_{16}	w_{17}	w_{18}	w_{19}
Round 1	$w_0 \oplus w_1 \oplus w_2 \oplus w_3$	$w_4 \oplus w_5 \oplus w_6 \oplus w_7$	$w_8 \oplus w_9 \oplus w_{10} \oplus w_{11}$	$w_{12} \oplus w_{13} \oplus w_{14} \oplus w_{15}$
Round 2	$w_3 \oplus w_6 \oplus w_9 \oplus w_{12}$	$w_{15} \oplus w_2 \oplus w_5 \oplus w_8$	$w_{11} \oplus w_{14} \oplus w_1 \oplus w_4$	$w_7 \oplus w_{10} \oplus w_{13} \oplus w_0$
Round 3	$w_{12} \oplus w_5 \oplus w_{14} \oplus w_7$	$w_0 \oplus w_9 \oplus w_2 \oplus w_{11}$	$w_4 \oplus w_{13} \oplus w_6 \oplus w_{15}$	$w_8 \oplus w_1 \oplus w_{10} \oplus w_3$
Round 4	$w_7 \oplus w_2 \oplus w_{13} \oplus w_8$	$w_3 \oplus w_{14} \oplus w_9 \oplus w_4$	$w_{15} \oplus w_{10} \oplus w_5 \oplus w_0$	$w_{11} \oplus w_6 \oplus w_1 \oplus w_{12}$
Round 5	$w_{15} \oplus w_9 \oplus w_5 \oplus w_3$	$w_{12} \oplus w_8 \oplus w_6 \oplus w_2$	$w_{13} \oplus w_{11} \oplus w_7 \oplus w_1$	$w_{14} \oplus w_{10} \oplus w_4 \oplus w_0$

Table 4. The message expansion for the simplified HAS-V.

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

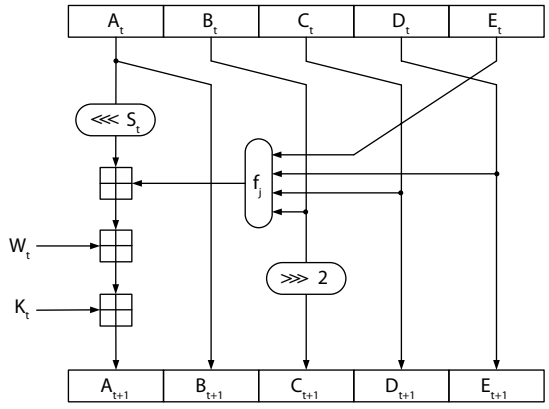

Fig. 1. The HAS-V step function.

Table 5. Constant K_t for the simplified HAS-V.

	Round 1	Round 2	Round 3
K_t	0x00000000	0x6ED9EBA1	0xA953FD4E

Table 6. Rotation value S_t for the simplified HAS-V.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
S_t	5	11	7	13	15	6	13	9	5	11	7	12	8	15	13	8	15	6	7	14

construction. They are often generated manually, using a great deal of intuition and experience. This paper further improves the results from [15], where an automated method is described for constructing these NL-characteristics.

In this paper, NL-characteristics are applied to the simplified HAS-V. An NL-characteristic is a set of conditions $\nabla Q_{-4}, \ldots \nabla Q_T$ and $\nabla W_0 \ldots \nabla W_{T-1}$. Each $\nabla X[i]$ represents a set of possible combinations for $(X[i], X[i]')$, as shown in Table 7. The number of steps T is left variable to be able to study step-reduced versions of this simplified HAS-V. In this paper, results will be obtained for $T = 45$.

3.1 Representation of Conditions on One Bit $\nabla Q_{t+1}[i]$

The step function (3) can be written as follows for every bit, for $0 \leq t < T$ and $0 \leq i < 32$. Indices i are calculated modulo 32. The carry input of the addition is denoted by $C_{t,i}$, the carry output by $C_{t+1,i+1}$.

$$\begin{aligned} C_{t+1,i+1} \parallel Q_{t+1}[i] \leftarrow & Q_t[i - S_t] + f_j(Q_{t-1}[i], Q_{t-2}[i+2], Q_{t-3}[i+2], \\ & Q_{t-4}[i+2]) + W_t[i] + K_t[i] + C_{t,i} \ . \end{aligned} \tag{5}$$

To calculate $Q_{t+1}[i]$, the bit positions of the previous state words Q_{t-k} $(0 \leq k < 5)$ are schematically represented in Fig. 2.

In the step function for every bit (5), a single, large addition is used with a carry input and output. The resulting carry states $(C_{t+1,i+1}, C'_{t+1,i+1})$ are then the only way in which adjacent bits of the same message word pair (W_t, W'_t) or internal states (Q_{t+1}, Q'_{t+1}) interact. If a particular carry state $(C_{t+1,i+1}, C'_{t+1,i+1})$ cannot occur as the output for the calculation of this bit $(Q_{t+1,i}, Q'_{t+1,i})$, nor as the input of the calculation of the next bit $(Q_{t+2,i+1}, Q'_{t+2,i+1})$, this combination of carries is said to be invalid.

For the calculation of every $\nabla Q_{t+1}[i]$, all valid combinations of $(C_{t,i}, C'_{t,i})$, (Q_{t-k}, Q'_{t-k}) for $0 \leq k < 5$ and (W_t, W'_t) are represented by an edge in Fig. 3. Imposing new conditions on $\nabla Q_{t+1}[i]$ will lead to the elimination of some of these edges.

The Boolean function f_j and the constant bit $K_t[i]$ are fixed for one bit position. Therefore, they are not included on the edges in Fig. 3. Each of the 2^{16} input bits of the edges then completely determines the six output bits $(Q_{t+1}[i], Q'_{t+1}[i])$ and $(C_{t,i+1}, C'_{t,i+1})$.

3.2 Propagation of Conditions for Every Word ∇Q_{t+1}

Initially, all input conditions are allowed for all bits. As this implies that all outputs are allowed for every bit, this is a self-consistent state. However, as soon as some restrictions are imposed on a bit, this may affect other bits. We refer to this mechanism as the propagation of conditions.

Table 7. All possible conditions for $(X[i], X'[i])$.

	(0,0)	(1,0)	(0,1)	(1,1)
?	✓	✓	✓	✓
-	✓	-	-	✓
x	-	✓	✓	-
0	✓	-	-	-
u	-	✓	-	-
n	-	-	✓	-
1	-	-	-	✓
#	-	-	-	-

	(0,0)	(1,0)	(0,1)	(1,1)
3	✓	✓	-	-
5	✓	-	✓	-
7	✓	✓	✓	-
A	-	✓	-	✓
B	✓	✓	-	✓
C	-	-	✓	✓
D	✓	-	✓	✓
E	-	✓	✓	✓

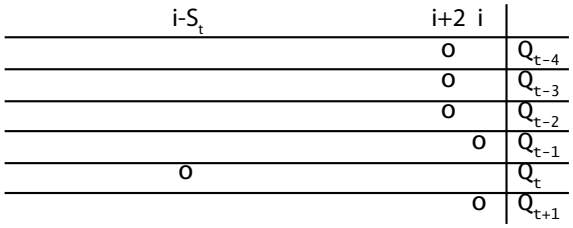

Fig. 2. Calculation of $Q_{t+1}[i]$ from $Q_t[i - S_t]$, $Q_{t-1}[i]$, $Q_{t-1}[i + 2]$, $Q_{t-1}[i + 2]$ and $Q_{t-1}[i + 2]$.

To calculate the possible conditions for every word ∇Q_{t+1} for $0 \leq t < T$, it is necessary to do both a forward and a backward propagation over all conditions $\nabla Q_{t+1}[i]$ for $0 \leq i < 32$. In Fig. 3, every possible input combination is shown as an edge connecting the input carries $(C_{t,i}, C'_{t,i})$ to the output carries $(C_{t,i+1}, C'_{t,i+1})$. Note that there can be multiple edges between two nodes.

In Fig. 4, a forward propagation (for $i = 0, 1, \ldots, 31$) is done where edges are removed if they start at an impossible input carry. In Fig. 5, a backward propagation is performed (for $i = 31, 30, \ldots, 0$) where edges are removed if the output carry is invalid. If necessary, the input conditions $\nabla Q_t[i - S_t]$, $\nabla Q_{t-1}[i]$, $\nabla Q_{t-2}[i + 2]$, $\nabla Q_{t-3}[i + 2]$,$\nabla Q_{t-4}[i + 2]$ and $\nabla W_t[i]$, as well as the output condition $\nabla Q_{t+1}[i]$ in the NL-characteristic are updated. In this way, one word can affect the conditions of another word.

This step is repeated for every word ∇Q_{t+1} for $0 \leq t < T$, until further propagation would not remove additional edges or until at least one condition is inconsistent. Every time a message bit $W_t[i]$ is assigned a new value, message bits $W_{t'}[i]$ that are related by the message expansion, are updated as well if necessary. The remaining valid paths for one word are then shown in Fig. 6.

3.3 Double conditions

Conditions that do not involve one pair of bits, but two pairs of bits, are referred to as "double conditions". The use of these is new to this paper. They are similar to Table 7, except that double conditions apply to four bits instead of two. Thus, there are 2^{16} possible double conditions, instead of 2^4.

For the simplified HAS-V, double conditions can be used in three locations for the calculation of one bit of ∇Q_{t+1}. These are shown in Fig. 7, as the result of the only possibilities of creating an overlap of at least two bits of Fig. 2 with a translated version of this pattern.

The use of the first double condition is explained, the other two cases are analogous. During the calculation of $\nabla Q_{t'+1}[2]$, a double condition is used to represent the possibilities of the joint

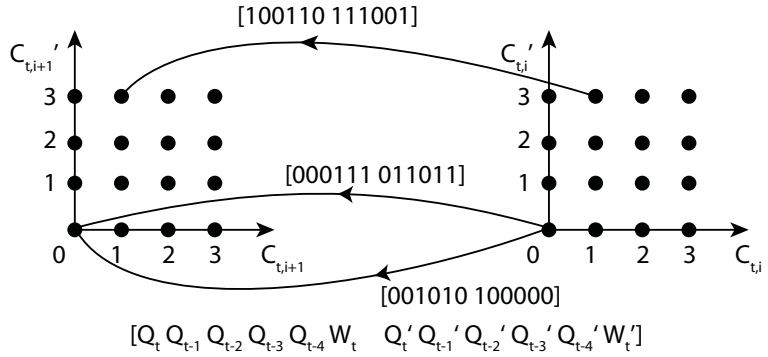

Fig. 3. Explanation of the edges in the graph. When adding four bits (and the carry input), the carry output $C_{t,i}$ can be 0, 1, 2 or 3. The addition of the corresponding four bits of the second message of the collision pair results in the carry $C'_{t,i}$.

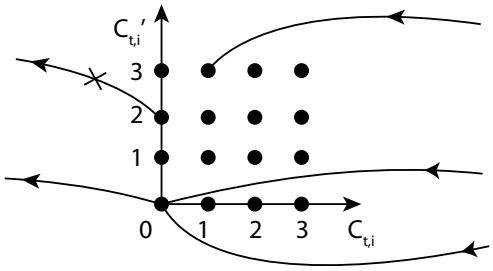

Fig. 4. Removing edges through forward propagation.

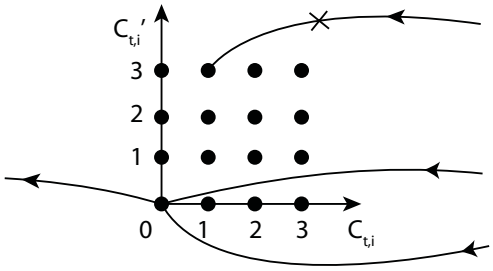

Fig. 5. Removing edges through backward propagation.

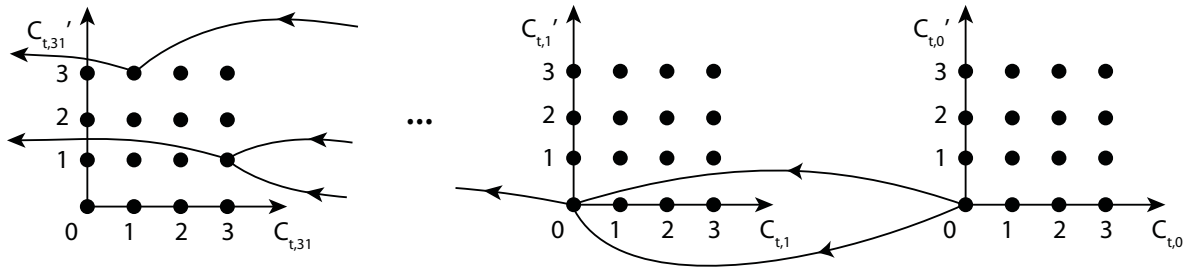

Fig. 6. Remaining valid paths for one word ∇Q_{t+1}.

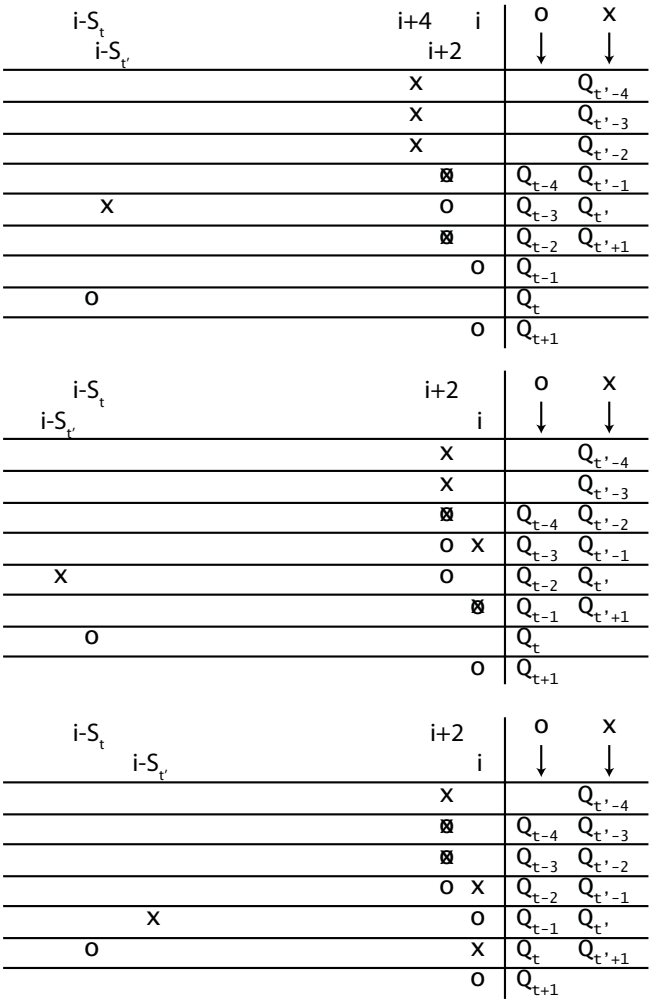

Fig. 7. Double conditions for the HAS-V step function, obtained as the only possible overlaps of at least two bits in Fig. 2 with a translated version of this pattern.

occurrence of $\nabla Q_{t'+1}[2]$ and $\nabla Q_{t'-1}[2]$. When $\nabla Q_{t+1}[0]$ is calculated, it can be seen that the same double condition now also applies to the joint occurrence of $\nabla Q_{t-2}[2]$ and $\nabla Q_{t-4}[2]$. It is possible that this information leads to the removal of additional edges. If this is the case, the number of iterations needed to construct an NL-characteristic is lowered, and inconsistencies can be found sooner. In our implementation of the search for NL-characteristics, double conditions can be implemented with minimal overhead.

3.4 Work factor

The work factor N_{w} of an NL-characteristic indicates the expected number of step function evaluations required to find a collision using this characteristic. When building NL-characteristics, the collision search is optimized by lowering the work factor. This concept was introduced in [15].

Message freedom $F_{\mathrm{W}}(t)$ "Single-message modification" [3] (also known as "single-step modification" [2]) can be used during the search process, as there is still freedom left in the choice of several expanded message words W_t. Due to the constraints imposed by the XOR-words, this is not possible for each of the 20 message word pairs (W_t, W'_t) of the first round. Of the five message word pairs involved in the calculation of each XOR-word, only the first four can be chosen. The last message word pair cannot be freely chosen, but must equal the XOR of the four others.

The message freedom $F_{\mathrm{W}}(t)$ of a characteristic at step t is the number of ways to choose (W_t, W'_t), without violating any (linear) condition imposed by the message expansion, given fixed values of (W_j, W'_j) for $0 \leq j < t$.

The description of the simplified HAS-V indicates that $F_{\mathrm{W}}(t)$ is always 1 for $t = 10, 14, 15, 19$ and $t \geq 20$. For the other values of t, $F_{\mathrm{W}}(t)$ is the product of the number of possibilities for conditions $\nabla W_t[i]$ for $0 \leq i < 32$. This number of possibilities equals the number of checkmarks (✓) for the respective conditions in Table 7.

Uncontrolled probability $P_{\mathrm{u}}(t)$ The uncontrolled probability $P_{\mathrm{u}}(t)$ of a characteristic at step t is the probability that the output (Q_{t+1}, Q'_{t+1}) follows the characteristic, given that all input pairs (Q_{t-k}, Q'_{t-k}) for $0 \leq k < 5$ and message word pairs (W_t, W'_t) follow this characteristic as well:

$$P_{\mathrm{u}}(t) = P\left((Q_{t+1}, Q'_{t+1}) \in \nabla Q_{t+1} \mid (Q_{t-k}, Q'_{t-k}) \in \nabla Q_{t-k} \text{ for } 0 \leq k < 5, \text{ and } (W_t, W'_t) \in \nabla W_t\right) . \tag{6}$$

This probability can be calculated as the number of remaining paths of Fig. 6, divided by the number of paths for which only the input pairs (Q_{t-k}, Q'_{t-k}) for $0 \leq k < 5$ and the message word pairs (W_t, W'_t) follow the characteristic, but not necessarily the output pair (Q_{t+1}, Q'_{t+1}).

Controlled probability $P_{\mathrm{c}}(t)$ The controlled probability $P_{\mathrm{c}}(t)$ of a characteristic at step t is the probability that there exists at least one pair of message words (W_t, W'_t) following the characteristic, such that the output (Q_{t+1}, Q'_{t+1}) follows the characteristic, given that all input pairs (Q_{t-k}, Q'_{t-k}) for $0 \leq k < 5$ do as well:

$$P_{\mathrm{c}}(t) = P\left(\exists (W_t, W'_t) \in \nabla W_t : \ (Q_{t+1}, Q'_{t+1}) \in \nabla Q_{t+1} \mid (Q_{t-k}, Q'_{t-k}) \in \nabla Q_{t-k} \text{ for } 0 \leq k < 5\right) . \tag{7}$$

A graph is made for every bit i for the calculation of $(Q_{t+1}[i], Q'_{t+1}[i])$ to determine this probability. Each node of the graph is a carry mask, indicating which of the 16 possible values of $(C_{t,i}, C'_{t,i})$ can occur. Thus, a carry mask can have 2^{16} possible values. Note the analogy with Table 7, where the possible combinations of $(X[i], X[i]')$ are shown.

Let n be the number of possibilities for $(Q_{t-k}[i], Q'_{t-k}[i]) \in \nabla Q_{t-k}[i]$ for $0 \leq k < 5$. For each possibility, we run through all carries $(C_{t,i}, C'_{t,i})$ and all message bit pairs $(W_t[i], W'_t[i])$. A binary 16×16 matrix indicates which transition possibilities from $(C_{t,i}, C'_{t,i})$ to $(C_{t,i+1}, C'_{t,i+1})$ can occur.

Using this 16×16 transition matrix, we can calculate the possible carry masks for bit $i + 1$ using the carry mask of bit i. For the least significant bit ($i = 0$), only one carry mask is possible: the carry is $(0, 0)$ with probability 1. Each of the edges in the graph has probability $1/n$. Unlike in Fig. 6, there is never more than one edge between two nodes. This step is repeated for every $(Q_{t-k}[i], Q'_{t-k}[i]) \in \nabla Q_{t-k}[i]$ for $0 \leq k < 5$.

Table 8. Lowest Hamming weights found for L-characteristics, not taking the weight of ∇Q_{t+1} for $0 \leq t < 20$ into account.

	collision	near-collision	pseudo-collision
40 steps	30	26	27
45 steps	75	68	65

This calculation is performed for bits $i = 0 \ldots 31$. We now consider the most significant bit ($i = 31$). One carry mask indicates that none of the carries $(C_{t,31}, C'_{t,31})$ are valid. $P_c(t)$ then equals the sum of all the other carry masks.

Total work factor N_w In the collision search tree, the average number of children of a node at step t is $F_W(t) \cdot P_u(t)$. Only a fraction $P_c(t)$ of the nodes at step t have children at all. The search stops as the last step $T-1$ of the compression function is reached. We can thus obtain the following recursive relation for the expected number of nodes $N_s(t)$ at every step of the compression function:

$$N_s(t) = \begin{cases} 1 & \text{for } t = T-1 \ , \\ \max\left(N_s(t+1) \cdot F_W^{-1}(t) \cdot P_u^{-1}(t), P_c^{-1}(t)\right) & \text{for } 0 \leq t < T-1 \ . \end{cases} \tag{8}$$

The total work factor is then given by

$$N_w = \sum_{t=0}^{T-1} N_s(t) \ . \tag{9}$$

In tables, the base 2 logarithms of $F_W(t)$, $P_u(t)$, $P_c(t)$, $N_s(t)$ and $N_w(t)$ are shown.

A difference with [15], is that in this work, the double conditions of Sect. 3.3 are also taken into account in the calculation of the work factor. These are assumed to be included in the definitions of $P_u(t)$ and $P_c(t)$. This is because double conditions are used in the actual collision search as well. Implementing this is possible with minimal overhead, and can only improve N_w. Experimental results of using these double conditions will be given in Sect. 4.

4 Finding NL-characteristics for 45 Steps

To obtain a good NL-characteristic, Stage 1 of [15] consists of obtaining a sparse L-characteristic to use as a starting point. As can be seen in Table 8, no suitable L-characteristic could be found for 45 steps of the simplified HAS-V. The weight of the ∇Q_{t+1} for the first round is not taken into account, assuming for simplicity that these can all be satisfied by single-message modification.

To overcome this problem, we looked for message differences that are localized at a small number of steps of the internal states ∇Q_{t+1}. The Boolean functions f_j are particularly well suited to allow for NL-characteristics consisting of very short collision regions. It can be seen that both f_1 and f_3 allow any input difference to be either passed on or canceled out at the output. For $f_2(B, C, D, E)$, this is also the case for every input difference, except for an input difference at D, which will always lead to an output difference. The HAS-V specification [12] reveals that this is by design, in an attempt to satisfy the "Strict Avalanche Criterion (SAC)" [17]. As the attacker can choose both messages m, m' of a collision pair, he can control the output differences of the f-function at certain positions (either probabilistically, or by single- or multi-message modification). This allows

Table 9. The work factor N_w (in base 2 logarithm) after each of the four stages.

	Stage 1	Stage 2	Stage 3	Stage 4
without double conditions	143.87	89.30	81.84	59.92
with double conditions	143.87	81.28	75.84	51.53

for more freedom in the construction of NL-characteristics, while still keeping the probability of the characteristic high.

Differences in the message words m_i are only introduced in $m_{12}[0]$ and $m_{14}[0]$. Due to the message expansion, these differences can be found in $W_{16}[0]$, $W_{18}[0]$, $W_{21}[0]$, $W_{23}[0]$. Before and after this collision region, equality is imposed on the internal state words.

In the short collision region, all conditions for $\nabla Q_{t+1}[i]$ are still unrestricted ("?") at Stage 1.

Stages 2 and 3 are the same as in [15]. In Stage 2, unrestricted conditions ("?") are randomly chosen, and the requirement that they are equal ("-") is imposed. This stage is repeated several times, until a characteristic with a sufficiently low work factor is obtained. Further in Stage 2, conditions ("x") start to appear, which are replaced by either ("u") or ("n") when selected. In Stage 3, local optimizations are performed by going over all "-" conditions, and replacing them by "0" or "1" if this improves the work factor. By repeating Stage 3 several times, the work factor gradually decreases. The end result after Stage 3 is shown in Table 12, with corresponding work factor $N_w = 2^{75.84}$.

After Stage 3, adding a single extra condition will never decrease the work factor. It is possible, however, to reduce the work factor even further. This is done in an additional stage, Stage 4, not described in [15]. In Stage 4, not one, but several conditions are added locally, as long as they do not worsen the work factor. If adding multiple conditions improves the work factor, a minimal set of conditions is derived from these, that still lowers the work factor. This set is obtained by relaxing the additional conditions again, one by one, to see if they had any impact on the global work factor. Only the conditions of this minimal set are kept. Experiments show that it is even possible, that relaxing conditions decreases the work factor of the NL-characteristic. This fourth stage is also repeated several times. The end result is shown in Table 13, where a work factor N_w of $2^{51.53}$ is obtained. After the Stage 4, it is not possible to decrease the work factor by adding or relaxing a single condition.

Note that the characteristics obtained after every stage are not necessarily the best possible. Every stage can thus be performed several times, until a characteristic is found that is good enough.

Experimental results indicating the impact of these double conditions on N_w after each of the four stages, are shown in Table 9.

Although time limits did not allow us to find a colliding message pair, we have verified for reduced versions that the complexity estimates accurately reflect the actual search cost, both with and without the inclusion of double conditions.

5 Conclusion and Future Work

This paper shows how techniques developed for SHA-1 in [15] can be further improved and generalized for a simplified variant of the hash function HAS-V. This simplified variant consists of only a single stream.

For 45 steps of this simplified HAS-V, an NL-characteristic is constructed, requiring about $2^{51.53}$ step function evaluations, or about 2^{46} compression function evaluations, to find a collision. A lot of the message bits can still be freely chosen when using this characteristic.

Stage 1 of method of De Cannière and Rechberger [15], the search for a good L-characteristic, is replaced by the requirement that collisions occur in a very short region. As the method described in this paper can be applied without finding good L-characteristics first, it might be used for hash functions such as RIPEMD-160 [18], for which also no good L-characteristics were found [19].

"Double conditions" are introduced as conditions for two pair of bits. They can be used to speed up the actual collision search.

An extra stage, Stage 4, is introduced to further improve the work factor for finding a collision. It is shown how this additional stage can reduce the work factor from $2^{75.84}$ step function evaluations, or about 2^{71} compression function evaluations, in Table 12, to $2^{51.53}$, or about 2^{46} compression function evaluations, in Table 13.

6 Acknowledgments

The authors would like to thank their colleagues at COSIC, and the symmetric cryptography subgroup in particular, for their useful comments and suggestions. Special thanks go to Vesselin Velichkov, who greatly helped in improving the clarity of this paper.

Several techniques in this paper were already used in the cryptanalysis of SHA-1 by Christophe De Cannière and Christian Rechberger [15], but had not been explained before. The authors are greatly indebted to Christian Rechberger, not only for his useful comments and suggestions, but also for allowing us build upon his previous work for SHA-1.

The original publication is available at `www.springerlink.com`.

References

1. Preneel, B.: Analysis and design of cryptographic hash functions. PhD thesis, Katholieke Universiteit Leuven (1993)
2. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 1–18
3. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 19–35
4. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In Shoup, V., ed.: CRYPTO. Volume 3621 of Lecture Notes in Computer Science., Springer (2005) 17–36
5. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register **27**(212) (November 2007) 62212–62220 Available: `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf` (2008/10/17).
6. Leurent, G.: Message Freedom in MD4 and MD5 Collisions: Application to APOP. In Biryukov, A., ed.: FSE. Volume 4593 of Lecture Notes in Computer Science., Springer (2007) 309–328
7. Crispin, M.: Internet Message Access Protocol - Version 4rev1. RFC 3501 (Proposed Standard) (March 2003) Updated by RFCs 4466, 4469, 4551, 5032, 5182.
8. Myers, J., Rose, M.: Post Office Protocol - Version 3. RFC 1939 (Standard) (May 1996) Updated by RFCs 1957, 2449.
9. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Naor, M., ed.: EUROCRYPT. Volume 4515 of Lecture Notes in Computer Science., Springer (2007) 1–22
10. Sotirov, A., Stevens, M., Appelbaum, J., Lenstra, A., Molnar, D.A., Osvik, D.A., de Weger, B.: MD5 considered harmful today: Creating a rogue CA certificate (December 2008) 25th Chaos Communications Congress, Berlin, Germany.

11. Rescorla, E.: HTTP Over TLS. RFC 2818 (Informational) (May 2000)
12. Park, N.K., Hwang, J.H., Lee, P.J.: HAS-V: A New Hash Function with Variable Output Length. In Stinson, D.R., Tavares, S.E., eds.: Selected Areas in Cryptography. Volume 2012 of Lecture Notes in Computer Science., Springer (2000) 202–216
13. Lim, C.H., Lee, P.J.: A Study on the Proposed Korean Digital Signature Algorithm. In Ohta, K., Pei, D., eds.: ASIACRYPT. Volume 1514 of Lecture Notes in Computer Science., Springer (1998) 175–186
14. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V Compression Function. In Nam, K.H., Rhee, G., eds.: ICISC. Volume 4817 of Lecture Notes in Computer Science., Springer (2007) 335–345
15. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In Lai, X., Chen, K., eds.: ASIACRYPT. Volume 4284 of Lecture Notes in Computer Science., Springer (2006) 1–20
16. Hawkes, P., Paddon, M., Rose, G.G.: Musings on the Wang et al. MD5 Collision. Cryptology ePrint Archive, Report 2004/264 (2004) `http://eprint.iacr.org/`.
17. Webster, A.F., Tavares, S.E.: On the Design of S-Boxes. In Williams, H.C., ed.: CRYPTO. Volume 218 of Lecture Notes in Computer Science., Springer (1985) 523–534
18. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In Gollmann, D., ed.: FSE. Volume 1039 of Lecture Notes in Computer Science., Springer (1996) 71–82
19. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In Katsikas, S.K., Lopez, J., Backes, M., Gritzalis, S., Preneel, B., eds.: ISC. Volume 4176 of Lecture Notes in Computer Science., Springer (2006) 101–116
20. Lipmaa, H., Moriai, S.: Efficient Algorithms for Computing Differential Properties of Addition. In Matsui, M., ed.: FSE. Volume 2355 of Lecture Notes in Computer Science., Springer (2001) 336–350
21. Lipmaa, H., Wallén, J., Dumas, P.: On the Additive Differential Probability of Exclusive-Or. In Roy, B.K., Meier, W., eds.: FSE. Volume 3017 of Lecture Notes in Computer Science., Springer (2004) 317–331
22. Indesteege, S., Preneel, B.: Practical Collisions for EnRUPT. In Dunkelman, O., ed.: FSE. Lecture Notes in Computer Science, Springer (2009) 246–259

A NL-characteristics

The NL-characteristics obtained after Stage 3 of Sect. 4 are shown in Table 12. After Stage 4, Table 13 is obtained. The work factor N_{w} improves from $2^{75.84}$ to $2^{51.53}$.

B A two-bit example

B.1 Introduction

Let n denote the word size in bits. We will write the differential probability of addition modulo 2^n as $\mathrm{xdp}^+(\alpha, \beta \rightarrow \gamma)$, where α, β and γ are bitstrings, most significant bit first. The best known method to find xdp^+ was an exponential-in-n calculation, before Lipmaa and Moriai introduced their algorithm in [20]. In [21], it was shown how xdp^+ can be calculated as a series of matrix multiplications in linear time in n.

In this section, we will calculate the $\mathrm{xdp}^+(11, 01 \rightarrow 10)$ by representing the addition as a graph and applying dynamic programming. We then show the relation of this graph method with [21], as both algorithms can be implemented in $\mathcal{O}(n)$ by using matrix multiplications. Afterwards, we mention several improvements and extensions to the graph method. Although this two-bit example may seem contrived, we found a fully worked-out example to be very useful to help understand the more abstract explanation of Fig. 3-6 in Sect. 3. There, an extension of the graph method is used to represent the step update function of HAS-V.

B.2 Visualizing $\mathrm{xdp}^+(11, 01 \rightarrow 10)$ in a graph

For $\mathrm{xdp}^+(\alpha_1 \parallel \alpha_0, \beta_1 \parallel \beta_0 \rightarrow \gamma_1 \parallel \gamma_0) \triangleq \mathrm{xdp}^+(11, 01 \rightarrow 10)$, we consider two additions, $z = x + y$ and $z' = x' + y'$, as shown in Fig. 8. For this particular xdp^+, we define the input differences for the least significant bits ($\alpha_0 = x_0 \oplus x'_0 = 1$ and $\beta_0 = y_0 \oplus y'_0 = 1$), and for the most significant bits ($\alpha_1 = x_1 \oplus x'_1 = 1$ and $\beta_1 = y_1 \oplus y'_1 = 0$). We assume that all valid inputs x, x' and y, y' are uniformly distributed. We then find $\mathrm{xdp}^+(11, 01 \rightarrow 10)$ as the probability that the output has difference $\gamma_0 = z_0 \oplus z'_0 = 0$ and $\gamma_1 = z_1 \oplus z'_1 = 1$.

The calculation for the least significant bits is shown in Table 10. As there is no carry input for the least significant bits, we only consider $C_0 = C'_0 = 0$. We list all values that satisfy the input conditions (α_0 and β_0) for the least significant bit. Note that the output condition ($\gamma_0 = z_0 \oplus z'_0 = 0$) is satisfied as well for all valid inputs $(C_0, C'_0, x_0, y_0, x'_0, y'_0)$.

Table 10. The summation for the least significant bits (z_0, z'_0), where $\alpha_0 = x_0 \oplus x'_0 = 1$ and $\beta_0 = y_0 \oplus y'_0 = 1$.

C_0	C'_0	x_0	y_0	x'_0	y'_0	C_1	C'_1	z_0	z'_0	α_0	β_0	γ_0
0	0	0	0	1	1	0	1	0	0	1	1	0
0	0	0	1	1	0	0	0	1	1	1	1	0
0	0	1	0	0	1	0	0	1	1	1	1	0
0	0	1	1	0	0	1	0	0	0	1	1	0

We then draw each of these input values as the four rightmost edges in the graph of Fig. 9. Every edge is labeled with the input conditions $[x_0 \;\; y_0 \;\; x'_0 \;\; y'_0]$, and starts at (C_0, C_0). Together,

these uniquely determine (z_0, z'_0) and (C_1, C_1). For now, the reader can ignore that some lines are dashed.

Next, we do the calculation for the most significant bits, as shown in Table 11. We again list all values that satisfy the input conditions (α_1 and β_1). Note that now, several carry inputs (C_1, C'_1) are possible. The output condition ($\gamma_1 = z_1 \oplus z'_1 = 1$) is not always satisfied, implying that $\text{xdp}^+(11, 01 \to 10) < 1$.

Table 11. The summation for the most significant bits (z_1, z'_1), where $\alpha_1 = x_1 \oplus x'_1 = 1$ and $\beta_1 = y_1 \oplus y'_1 = 0$.

C_1	C'_1	x_1	y_1	x'_1	y'_1	C_2	C'_2	z_1	z'_1	α_1	β_1	γ_1
0	0	0	0	1	0	0	0	0	1	1	0	1
0	0	0	1	1	1	0	1	1	0	1	0	1
0	0	1	0	0	0	0	0	1	0	1	0	1
0	0	1	1	0	1	1	0	0	1	1	0	1
1	0	0	0	1	0	0	0	1	1	1	0	0
1	0	0	1	1	1	1	1	0	0	1	0	0
1	0	1	0	0	0	1	0	0	0	1	0	0
1	0	1	1	0	1	1	0	1	1	1	0	0
0	1	0	0	1	0	0	1	0	0	1	0	0
0	1	0	1	1	1	0	1	1	1	1	0	0
0	1	1	0	0	0	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	0	0	1	0	0

Again, we draw each of the inputs of Table 11 as edges in Fig. 9. To improve the readability, we use three separate coordinate systems on top of each other on the left of the figure. These should in fact overlap: the same nodes are represented three times. For example, four edges end in $(C_2, C'_2) = (0, 0)$. If the output pairs are valid ($\gamma_1 = z_1 \oplus z'_1 = 1$), we use full lines, and if they are invalid ($\gamma_1 = z_1 \oplus z'_1 = 0$), dashed lines are used.

Due to backward propagation (explained in Fig. 5), the inputs $[x_0 \ y_0 \ \ x'_0 \ y'_0]$ with values [00 11] and [11 00] become dashed lines as well: they will eventually result in an incorrect output difference $\gamma_1 = 0$. The probability $\text{xdp}^+(11, 01 \to 10)$ is then equal to the number of paths in the graph with valid inputs (x, x', y, y') and outputs z, z' (full lines), divided by the number of paths that have valid inputs $(x, x'y, y')$ (full or dashed lines). This ratio is equal to 8/16, or 1/2.

Note that storing this graph does not require a lot of memory. For every bit in the n-bit addition, we need to store 2^6 bits. Each of these 2^6 bits is either set to 1 if the input $(C_0, C'_0, x_0, y_0, x'_0, y'_0)$ is valid, and 0 otherwise. For the entire n-bit addition, we thus need to store only $2^6 n$ bits; the memory requirement is $\mathcal{O}(n)$.

B.3 Calculating $\text{xdp}^+(11, 01 \to 10)$ using matrix multiplications

Similar to [21], we can calculate xdp^+ as a series of matrix multiplications. The graph of Fig. 9 can be seen as a first-order Markov chain. We have $[1 \ 0 \ 0 \ 0]^T$ as the initial distribution, as the input carry of the addition is $C_0 = C'_0 = 0$ with probability 1. All three other input carries (C_0, C'_0) have probability 0.

As the input conditions for $[x_0 \ y_0 \ \ x'_0 \ y'_0]$ are given, these specify the transition matrix of the Markov chain. Every column contains the transition probabilities for one carry input (C_0, C'_0) to

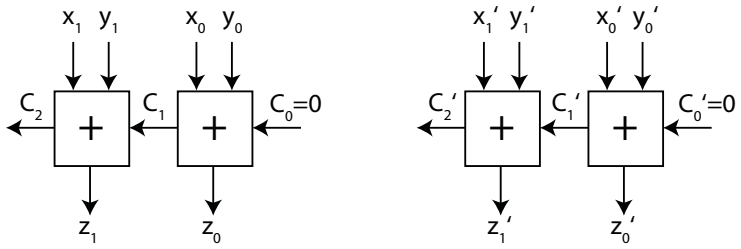

Fig. 8. Calculating $z = x + y$ and $z' = x' + y'$. All variables with subscripts represent one bit.

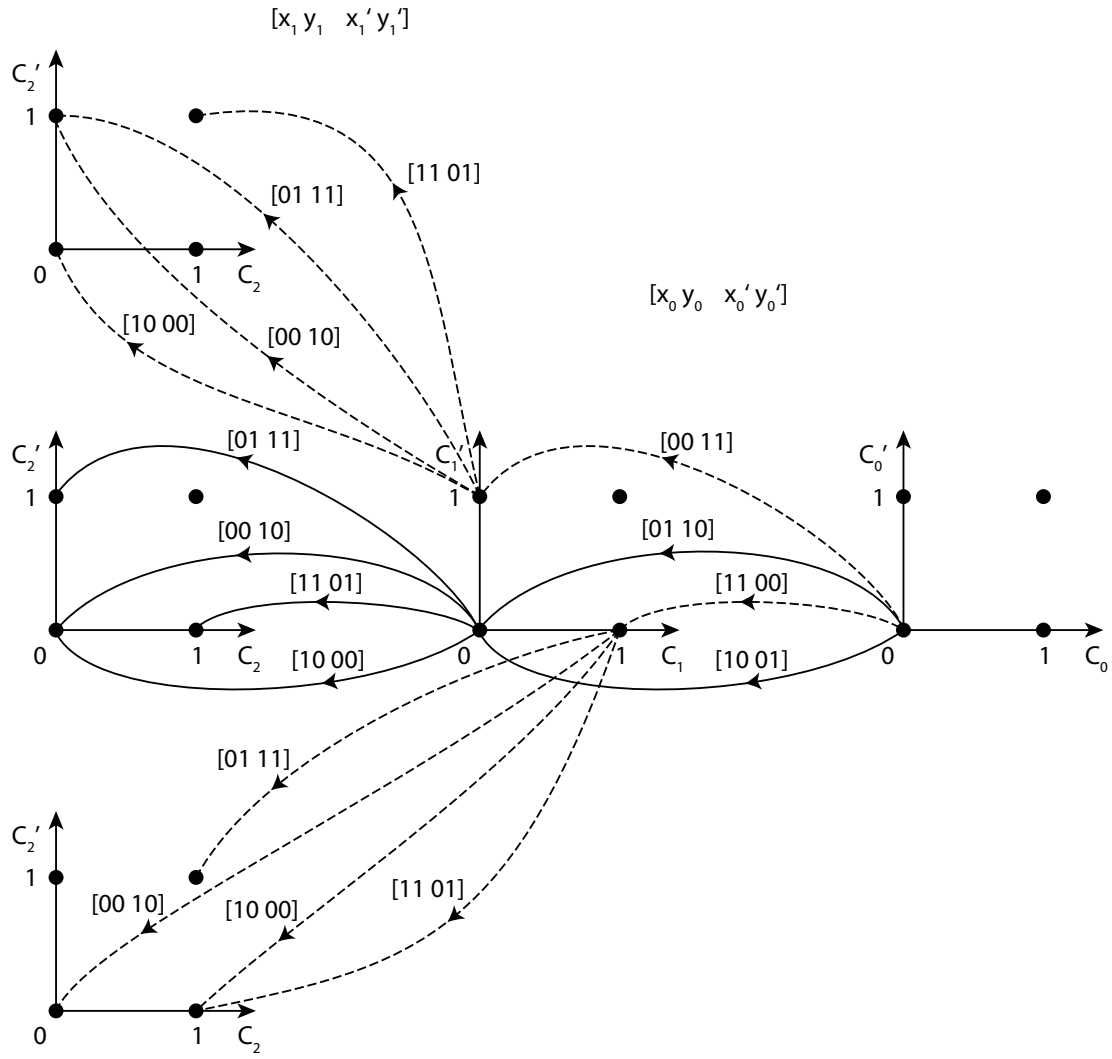

Fig. 9. Graph representation to calculate $\mathrm{xdp}^+(11, 01 \to 10)$. Only valid input pairs are shown. Full lines are used for the eight paths that have valid output pairs ($\gamma_1 = z_1 \oplus z_1' = 1$), and dashed lines are used for paths with invalid output pairs ($\gamma_1 = z_1 \oplus z_1' = 0$). As there are eight of each, the ratio gives $\mathrm{xdp}^+(11, 01 \to 10) = 8/16 = 1/2$. The three coordinate systems on top of each other on the left represent the same nodes three times. This makes the drawing more readable, however note that, for example, four edges end in $(C_2, C_2') = (0, 0)$.

every carry output (C_1, C'_1). We left-multiply the initial distribution by this transition matrix. We do the same for every subsequent bit of the n-bit addition.

Lastly, we sum all probabilities (by left-multiplying by [1 1 1 1]): the carry outputs (C_n, C'_n) are not used, so all of them are valid. This gives us the total probability of xdp^+.

B.4 Extending the graph method

As the reader may have noticed, the matrices of the previous section are larger than those in [21]. This is because we do not take the symmetry into account: although the value of $C_i \oplus C'_i$ would be sufficient, we keep track of the values of (C_i, C'_i). This symmetry exists because we restrict the input differences α, β and the output differences γ to exclusive-or differences.

The graph based method of the previous section, however, can also support the signed differences that were used for the cryptanalysis of MD5 [3], and as well as all the other generalized conditions of Table 7.

It is straightforward to generalize the graph method to the addition of three or more words. In this case, we extend each of the n adders of Fig. 8 to three or more input bits. This will increase the maximal values of the carry (C_i, C'_i): for example, the addition of four bits (and the carry input) can have a maximal carry output of 3.

In this case, value of the carry (C_i, C'_i) is equal to all output bits of the adder at position i, except the least significant bit. This can be seen as a variant of Fig. 8, where three or more bits are input to every adder.

In fact, the method can be generalized for any combination of additions, exclusive-ors and Boolean functions, as long as no rotations are present (except at the input or output). It is this calculation that was used for every step of SHA-1 in [15], and is also used for every step of HAS-V in this paper. By constructing higher-order Markov chains, the graph method was used in [22] to efficiently calculate the differential probability of a multiplication by 9, given by $\mathrm{xdp}^+(x, x \ll 3)$.

Table 12. NL-characteristic of 45 steps after Stage 3, work factor $N_{\mathrm{w}} = 2^{75.84}$.

t	∇Q_{t+1}	∇W_t	F_{W}	$P_{\mathrm{u}}(t)$	$P_{\mathrm{c}}(t)$	$N_{\mathrm{s}}(t)$
-5	00001111010010111000011111000011					
-4	01000000110010010101000111011000					
-3	01100010111010110111001111111010					
-2	11101111110011011010101110001001					
-1	01100111010001010010001100000001					
0	----------	----------	32	0.00	0.00	0.00
1	----------	----------	32	0.00	0.00	0.00
2	----------	----------	32	0.00	0.00	0.00
3	----------	----------	32	0.00	0.00	0.00
4	----------	----------	32	0.00	0.00	0.00
5	----------	----------	32	0.00	0.00	0.00
6	----------	----------	32	0.00	0.00	0.00
7	----------	----------10	30	0.00	0.00	0.00
8	----------	----------	32	0.00	0.00	0.00
9	----------	----------	32	0.00	0.00	0.00
10	----------	----------	0	0.00	0.00	0.00
11	----------	----------	32	0.00	0.00	0.00
12	----------	----------	32	0.00	0.00	0.00
13	----------0---------	----------	32	-1.00	0.00	22.60
14	----------010001101100-	----------	0	-12.00	0.00	53.60
15	----------001011---1-0	----------	0	-9.00	0.00	41.60
16	---0----------unnnnnnnnnnnn	1----------u	30	-14.00	-1.00	32.60
17	0--1----------0100u111010	----------	32	-13.00	0.00	48.60
18	0--1------1---------uu-uu0001110	----------000-----u	28	-19.77	-7.77	67.60
19	01-unn--nnu----------11000nn10	0----------	0	-19.00	-1.97	75.83
20	n-u0uu001-000----------0-0000-10	----------1-101--00	0	-14.30	-2.30	56.83
21	u-u-0n11----0---------11-010--10	1----------u	0	-18.98	-6.42	42.53
22	--1-110---011----------0-n----11	----------10	0	-10.00	-1.00	23.56
23	--0-0-10----------n-0---0	----------000-----u	0	-7.56	-1.61	13.56
24	--1-1--1----------1----	----------	0	-4.00	0.00	6.00
25	----------0----	----------	0	-1.00	0.00	2.00
26	----------1----	----------	0	-1.00	0.00	1.00
27	----------	----------	0	0.00	0.00	0.00
28	----------	----------	0	0.00	0.00	0.00
29	----------	----------	0	0.00	0.00	0.00
30	----------	----------	0	0.00	0.00	0.00
31	----------	----------	0	0.00	0.00	0.00
32	----------	----------	0	0.00	0.00	0.00
33	----------	----------	0	0.00	0.00	0.00
34	----------	0----------	0	0.00	0.00	0.00
35	----------	----------	0	0.00	0.00	0.00
36	----------	----------	0	0.00	0.00	0.00
37	----------	----------	0	0.00	0.00	0.00
38	----------	----------	0	0.00	0.00	0.00
39	----------	----------	0	0.00	0.00	0.00
40	----------	----------	0	0.00	0.00	0.00
41	----------	0----------	0	0.00	0.00	0.00
42	----------	----------	0	0.00	0.00	0.00
43	----------	----------10	0	0.00	0.00	0.00
44	----------	----------	0	0.00	0.00	0.00

Table 13. NL-characteristic of 45 steps after Stage 4, work factor $N_{\mathrm{w}} = 2^{51.53}$.

t	∇Q_{t+1}	∇W_t	F_{W}	$P_{\mathrm{u}}(t)$	$P_{\mathrm{c}}(t)$	$N_{\mathrm{s}}(t)$
-5	00001111010010111000011111000011					
-4	01000000110010010101000111011000					
-3	01100010111010110111001111111010					
-2	11101111110011011010101110001001					
-1	01100111010001010010001100000001					
0	----------	----------	32	0.00	0.00	0.00
1	----------	----------	32	0.00	0.00	0.00
2	----------	----------	32	0.00	0.00	0.00
3	----------	----------	32	0.00	0.00	0.00
4	----------	----------	32	0.00	0.00	0.00
5	----------	----------	32	0.00	0.00	0.00
6	----------	----------	32	0.00	0.00	0.00
7	----------	----00----------10	28	0.00	0.00	0.00
8	----------	----------	32	0.00	0.00	0.00
9	----------	----------	32	0.00	0.00	0.00
10	----------	----------	0	0.00	0.00	0.00
11	----------0----------	----------	32	-1.00	0.00	0.00
12	----------0-0--------	----------	32	-2.00	0.00	0.00
13	----------1-00-----0--	----------	32	-4.00	0.00	23.48
14	----------0100011011001	----------	0	-13.00	0.00	51.48
15	------00----------0001011011110	----------	0	-17.00	0.00	38.48
16	-1-0--110----------unnnnnnnnnnnn	10----------1-00---0u	25	-17.83	-4.24	21.48
17	01-1--1----------110100u111010	----------0-------	31	-18.00	0.00	28.65
18	00-1-1110011111-----uu1uu0001110	----------01000--1--u	25	-20.00	-1.00	41.65
19	01-unn--nnu1111------1011000nn10	000-00000100000------1-100000000	0	-11.58	-8.59	46.65
20	n-u0uu0010000---------0000000-10	000001011110---1------0101011000	0	-6.10	-4.62	35.07
21	u-u-0n11--110---------11-01---10	10----------1-00---0u	0	-10.03	-1.00	28.96
22	--1-110---011----------0-n----11	----00----------10	0	-7.46	-0.12	18.93
23	--0-0-10----------n-0---0	----------01000--1--u	0	-5.48	0.00	11.48
24	--1-1--1----------1----	----------	0	-4.00	0.00	6.00
25	----------0----	----------	0	-1.00	0.00	2.00
26	----------1----	----------	0	-1.00	0.00	1.00
27	----------	----------	0	0.00	0.00	0.00
28	----------	----------	0	0.00	0.00	0.00
29	----------	----------	0	0.00	0.00	0.00
30	----------	----------	0	0.00	0.00	0.00
31	----------	----------	0	0.00	0.00	0.00
32	----------	----------0-------	0	0.00	0.00	0.00
33	----------	----------	0	0.00	0.00	0.00
34	----------	000-00000100000------1-100000000	0	0.00	0.00	0.00
35	----------	----------	0	0.00	0.00	0.00
36	----------	----------	0	0.00	0.00	0.00
37	----------	----------	0	0.00	0.00	0.00
38	----------	----------	0	0.00	0.00	0.00
39	----------	----------	0	0.00	0.00	0.00
40	----------	----------	0	0.00	0.00	0.00
41	----------	000-00000100000------1-100000000	0	0.00	0.00	0.00
42	----------	----------	0	0.00	0.00	0.00
43	----------	----00----------10	0	0.00	0.00	0.00
44	----------	----------	0	0.00	0.00	0.00