7 SHA-0 and SHA-1

Contents

7.1 Overview

Chapter 7 covers all results on SHA-0 and SHA-1. First we give a formal definition of SHA-0 and SHA-1 in Section 7.2 and provide a treatment on published collision attacks and techniques in Section 7.3. The remaining sections cover our contributions.

Similar to MD5, we apply and improve the differential path analysis and algorithms of Chapter 5 for SHA-0 and SHA-1 in Section 7.4. However in contrast to MD5, this differential path construction is only to be used in the first round, i.e., the first 20 steps.

For the remaining three rounds, i.e., the last 60 steps, the technique of combining *local collisions* is used. *Disturbance vectors* are used to describe combinations of local collisions that may be used for near-collision attacks. So far, for various reasons, it is assumed that the local collisions behave independently in the literature on the analysis of disturbance vectors. This assumption has been shown to be flawed by Stéphane Manuel [Man11].

In Section 7.5, we present a method to analyze the success probability of disturbance vectors over the last three rounds that does not assume independence of local collisions. Our method is based on constructing differential paths that follow the prescribed local collision differences and summing the success probabilities of these differential paths. Since the number of possible differential paths can grow exponentially in the number of steps, various techniques are used to reduce this growth. This method also allows one to divert from the prescribed local collision differences at the beginning of the second round and the last few steps in order to obtain higher success probabilities. Furthermore, it provides an easy way to determine message expansion conditions that are sufficient to obtain the (near-)optimal success probability, a topic which so far has only been treated thoroughly on individual local collisions instead of combinations thereof.

We have constructed and implemented a near-collision attack against full SHA-1 using the above mentioned tools. It has an estimated complexity equivalent to $2^{57.5}$ SHA-1 compressions which can be used directly in an identical-prefix collision attack. This improves upon the near-collision attack with complexity of about 2^{68} SHA-1 compressions presented by Wang et al. [WYY05b]. The construction of our near-collision attack is presented in Section 7.6. This near-collision attack is practically

achievable within a year using about 1300 pc-cores[22]. As no actual near-collision block have been found yet using this near-collision attack, we discuss the verification of both the correctness and the complexity of our near-collision attack in Section 7.6.9.

This near-collision attack results in an identical-prefix collision attack against SHA-1 with an average complexity between $2^{60.3}$ and $2^{65.3}$ calls to the compression function of SHA-1 as explained in Section 7.6.1. Finally, based on this near-collision attack, we present a chosen-prefix collision attack against SHA-1 with an average complexity of about $2^{77.1}$ SHA-1 compressions in Section 7.7.

Although we have not constructed any attacks against SHA-0 or its compression function, the differential path construction algorithm in Section 7.4 and the differential cryptanalysis in Section 7.5 can be used directly in the construction of collision attacks against SHA-0. In particular, the differential cryptanalysis may allow an improvement over current collision attacks due to a better selection of the disturbance vector, the target values for δIHV_{diff} and the message word bitrelations, since the exact joint success probabilities of local collisions can be used instead of approximations based on the individual success probabilities of local collisions.

7.2 Description of SHA-0 and SHA-1

7.2.1 Overview

SHA-0 and SHA-1 work as follows on a given bit string M of arbitrary bit length, cf. [NIS95]:

1. *Padding.* Pad the message: first append a '1'-bit, next the least number of '0' bits to make the resulting bit length equal to 448 mod 512, and finally the bit length of the original unpadded message M as a 64-bit big-endian[23] integer. As a result the total bit length of the padded message $\widehat{M}$ is $512N$ for a positive integer N.

2. *Partitioning.* Partition the padded message $\widehat{M}$ into N consecutive 512-bit blocks M_0, $M_1, \ldots, M_{N-1}$.

3. *Processing.* To hash a message consisting of N blocks, SHA-0 and SHA-1 go through $N+1$ states IHV_i, for $0 \leq i \leq N$, called the *intermediate hash values*. Each intermediate hash value IHV_i is a tuple of five 32-bit words (a, b, c, d, e). For $i = 0$ it has a fixed public value called the *initial value* (IV):

$$IV = (\texttt{67452301}_{16}, \texttt{efcdab89}_{16}, \texttt{98badcfe}_{16}, \texttt{10325476}_{16}, \texttt{c3d2e1f0}_{16}).$$

   For $i = 1, 2, \ldots, N$ intermediate hash value IHV_i is computed using the respective SHA-0 or SHA-1 compression function described below:

$$IHV_i = \text{SHA0Compress}(IHV_{i-1}, M_{i-1}) \quad \text{for SHA-0;}$$

22. Measured on a single core of a Intel Core2 Q9550 2.83Ghz processor.
23. SHA-0/SHA-1 uses big-endian to convert between words and bit strings, whereas MD5 uses little-endian.

$$IHV_i = \text{SHA1Compress}(IHV_{i-1}, M_{i-1}) \quad \text{for SHA-1.}$$

4. *Output.* The resulting hash value is the last intermediate hash value IHV_N, expressed as the concatenation of the hexadecimal byte strings of the five words a, b, c, d, e of $IHV_N = (a, b, c, d, e)$, converted back from their big-endian representation. As an example the IV would be expressed as

$$\texttt{67452301efcdab8998badcfe10325476c3d2e1f0}_{16}.$$

7.2.2 SHA-0 compression function

The input for the compression function SHA0Compress(IHV, B) consists of an intermediate hash value $IHV_{\text{in}} = (a, b, c, d, e)$ and a 512-bit message block B. The compression function consists of 80 *steps* (numbered 0 to 79), split into four consecutive *rounds* of 20 steps each. Each step t uses modular additions, left rotations, and a non-linear function f_t, and involves an *Addition Constant* AC_t. The addition constants are defined per round as follows:

$$AC_t = \begin{cases} \texttt{5a827999}_{16} & \text{for } 0 \le t < 20, \\ \texttt{6ed9eba1}_{16} & \text{for } 20 \le t < 40, \\ \texttt{8f1bbcdc}_{16} & \text{for } 40 \le t < 60, \\ \texttt{ca62c1d6}_{16} & \text{for } 60 \le t < 80. \end{cases}$$

The non-linear function f_t also depends on the round:

$$f_t(X, Y, Z) = \begin{cases} F(X, Y, Z) = (X \wedge Y) \oplus (\overline{X} \wedge Z) & \text{for } 0 \le t < 20, \\ G(X, Y, Z) = X \oplus Y \oplus Z & \text{for } 20 \le t < 40, \\ H(X, Y, Z) = (X \wedge Y) \vee (Z \wedge (X \vee Y)) & \text{for } 40 \le t < 60, \\ I(X, Y, Z) = X \oplus Y \oplus Z & \text{for } 60 \le t < 80. \end{cases} \tag{7.1}$$

The 512-bit message block B is partitioned into sixteen consecutive 32-bit strings which are then interpreted as 32-bit words m_0, $m_1, \ldots, m_{15}$ (with big-endian byte ordering), and expanded to 80 words W_t, for $0 \le t < 80$,

$$W_t = \begin{cases} m_t & \text{for } 0 \le t < 16, \\ W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16} & \text{for } 16 \le t < 80. \end{cases} \tag{7.2}$$

Note that message expansion relation is reversible:

$$W_{t-16} = W_t \oplus W_{t-3} \oplus W_{t-8} \oplus W_{t-14}, \quad \text{for } 16 \le t < 80. \tag{7.3}$$

Similar to MD5 we describe SHA-0's compression function SHA0Compress in an 'unrolled' version such that SHA0Compress $\in \mathcal{F}_{\text{md4cf}}$. For each step t the compression function algorithm uses a working state consisting of five 32-bit words Q_t, Q_{t-1}, Q_{t-2},

Q_{t-3} and Q_{t-4} and calculates a new state word Q_{t+1}. The working state is initialized as

$$(Q_0, Q_{-1}, Q_{-2}, Q_{-3}, Q_{-4}) = (a, b, RR(c, 30), RR(d, 30), RR(e, 30)). \tag{7.4}$$

For $t = 0, 1, \ldots, 79$ in succession, Q_{t+1} is calculated as follows:

$$\begin{aligned} F_t &= f_t(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30)), \\ Q_{t+1} &= F_t + AC_t + W_t + RL(Q_t, 5) + RL(Q_{t-4}, 30). \end{aligned} \tag{7.5}$$

After all steps are computed, the resulting state words are added to the input intermediate hash value and returned as output:

$$\begin{aligned} &\text{SHA0Compress}(IHV_{\text{in}}, B) = \\ &\quad (a + Q_{80},\ b + Q_{79},\ c + RL(Q_{78}, 30),\ d + RL(Q_{77}, 30),\ e + RL(Q_{76}, 30)). \end{aligned} \tag{7.6}$$

7.2.3 SHA-1 compression function

The compression function SHA1Compress of SHA-1 is defined identically to the compression function SHA0Compress of SHA-0 except for the message expansion where a bitwise rotation is added:

$$W_t = \begin{cases} m_t & \text{for } 0 \le t < 16, \\ RL(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}, 1) & \text{for } 16 \le t < 80. \end{cases} \tag{7.7}$$

Note that SHA1Compress $\in \mathcal{F}_{\text{md4cf}}$ and also that SHA-1's message expansion relation is reversible:

$$W_{t-16} = RR(W_t, 1) \oplus W_{t-3} \oplus W_{t-8} \oplus W_{t-14}, \quad \text{for } 16 \le t < 80. \tag{7.8}$$

7.2.4 Expanded messages

A sequence $(W_t)_{t=0}^{79}$ is called an *expanded message* when it is the result of the message expansion from $W_0, \ldots, W_{15}$. An expanded message can be seen as an element of $\mathbb{F}_2^{80\times 32}$. Note that this "definition" differs between the contexts of SHA-0 and SHA-1 and has the following properties:

- for both SHA-0 and SHA-1, the message expansion is linear with respect to $\oplus$: $(W_t \oplus V_t)_{t=0}^{79}$ is an expanded message if $(W_t)_{t=0}^{79}$ and $(V_t)_{t=0}^{79}$ are expanded messages;
- any 16 consecutive words $W_t, \ldots, W_{t+15}$ uniquely determine the entire expanded message $W_0, \ldots, W_{79}$, since both SHA-0's and SHA-1's message expansion relations are reversible;
- the left rotation $(RL(W_t, r))_{t=0}^{79}$ of any expanded message $(W_t)_{t=0}^{79}$ by r bits is also an expanded message;

- the forward and backward shifts $(W_t)_{t=1}^{80}$ and $(W_t)_{t=-1}^{78}$ of any expanded message $(W_t)_{t=0}^{79}$ are also expanded messages, where for SHA-x (using Eq. 7.2 and 7.7):

$$\begin{aligned} W_{80} &= RL(W_{80-3} \oplus W_{80-8} \oplus W_{80-14} \oplus W_{80-16}, x); \\ W_{-1} &= RR(W_{15}, x) \oplus W_{15-3} \oplus W_{15-8} \oplus W_{15-14}. \end{aligned}$$

7.3 Techniques towards collision attacks

SHA-1 and its predecessor SHA-0 have a more complex message expansion compared to MD5. Changing any bit in the first 16 words $W_0, \ldots, W_{15}$ leads to many bit differences in the succeeding words $W_{16}, \ldots, W_{79}$. Constructing a collision attack requires thus a different approach to find a good differential path over the last three rounds.

7.3.1 Local collisions

In 1998, Chabaud and Joux [CJ98] constructed a collision attack against SHA-0 based on local collisions. The idea of a local collision is simple: in some step t a disturbance is created by some message word difference $\delta W_t = 2^b$ resulting in $\delta Q_{t+1} = 2^b$. This disturbance is corrected over the next five steps, so that after those five steps no differences occur in the five working state words.

Obvious corrections for step $t+1$ and $t+5$ are $\delta W_{t+1} = -2^{(b+5 \bmod 32)}$ and $\delta W_{t+5} = -2^{(b+30 \bmod 32)}$, since both corrections occur with probability at least $1/2$ and for many values of b this probability is close to 1. In steps $t+2$, $t+3$ and $t+4$, the disturbance $\delta Q_{t+1} = 2^b$ might cause δF_{t+2}, δF_{t+3} and δF_{t+4} to be non-zero, which can be corrected with $\delta W_{t+k} = -\delta F_{t+k}$ for $k \in \{2, 3, 4\}$. Possible corrections for steps $t+2$, $t+3$ and $t+4$ vary per round. Common non-zero values for δF_{t+2}, δF_{t+3} and δF_{t+4} are $\pm 2^b$, $\pm 2^{(b+30 \bmod 32)}$ and $\pm 2^{(b+30 \bmod 32)}$, respectively.

7.3.2 Disturbance vector

Due to the properties of the message expansion, Chabaud and Joux [CJ98] were able to interleave many of these local collisions such that the message word signed bit differences $(\Delta W_t)_{t=0}^{79}$ conform to the message expansion. For more convenient analysis, they consider the *disturbance vector* which is a non-zero expanded message $(DV_t)_{t=0}^{79}$ where every '1'-bit $DV_t[b]$ marks the start of a local collision based on the disturbance $\delta W_t[b] = \pm 1$.

We use $(DW_t)_{t=0}^{79}$ to denote all message word bit differences without sign: $W'_t = W_t \oplus DW_t$. Note that the vector $(DW_t)_{t=0}^{79}$ must be an expanded message, since $(W_t)_{t=0}^{79}$ and $(W'_t)_{t=0}^{79}$ are expanded messages. Chabaud and Joux use the same relative message word bit differences for all local collisions, as this implies that $(DW_t)_{t=0}^{79}$ forms an XOR($\oplus$) over rotated shifts of the disturbance vector. Hence, $(DW_t)_{t=0}^{79}$ is an expanded message.

Let $\mathcal{R} \subset \{0,\ldots,5\} \times \{0,\ldots,31\}$ describe the relative indexes of the changed bits over the six steps of a local collision, then $(DW_t)_{t=0}^{79}$ can be determined as:

$$DW_t = \bigoplus_{(i,r)\in\mathcal{R}} RL(DV_{t-i}, r), \quad t \in \{0,\ldots,79\}. \tag{7.9}$$

Here $DV_{-5},\ldots,DV_{-1}$ are the words resulting from shifting $(DV_t)_{t=0}^{79}$ forward five times for SHA-x:

$$DV_i = RR(DV_{i+16}, x) \oplus DV_{i+13} \oplus DV_{i+8} \oplus DV_{i+2}, \quad i = -1,\ldots,-5.$$

Table 7-1 provides a list of all high probability local collisions with a single bit disturbance in ΔQ_{t+1} (no carries) and for which steps these local collisions are valid. Although for some steps the local collision with the fewest differences is

$$(\delta W_i)_{i=t}^{t+5} = (2^b, -2^{b+5 \bmod 32}, 0, 0, 0, -2^{b+30 \bmod 32}),$$

thus $\mathcal{R} = \{(0,0),(1,5),(5,30)\}$, this local collision cannot be used in rounds two and four due to their boolean function.

The only local collision that is valid for all steps (not allowing carries in ΔQ_{t+1}) is

$$(\delta W_i)_{i=t}^{t+5} = (2^b, -2^{b+5 \bmod 32}, \pm 2^b, \pm 2^{b+30 \bmod 32}, \pm 2^{b+30 \bmod 32}, -2^{b+30 \bmod 32}),$$

thus $\mathcal{R} = \{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}$ which is used for all published attacks against either SHA-0 or SHA-1 and is used throughout the remainder of this thesis.

To show that there are no other possibilities, we show that the set

$$\mathcal{R} = \{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}$$

is the only set of relative bit differences that is valid for $t = 30$ and $b = 31$. Since there can be no carry, i.e., ΔQ_{t+1} is either $\{31\}$ or $\{\overline{31}\}$, it follows that here any local collision can only consist of relative bit differences that are in the set $\{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}$. For all such local collisions the first, second and sixth relative bit differences in this set are unavoidable. Whether the remaining relative bit differences $\{(2,0),(3,30),(4,30)\}$ either must, must not or may be used depends on the boolean function. For $t = 30$, this is the XOR boolean function that always has an output bit difference if there is a difference in only one of the corresponding input bits. This implies that all three remaining relative bit differences must be used. Hence, for $t = 30$ and $b = 31$ the only possible local collisions are described by the relative bit differences

$$\mathcal{R} = \{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}.$$

In the case of SHA-0, since its message expansion does not use bitwise rotation, all '1'-bits in the disturbance vector can be limited to a single bit position. Chabaud

Table 7-1: *Possible local collisions for SHA-0 and SHA-1*

$\mathcal{R}$	t
$\{(0,0),(1,5),(2,0),(3,30),(4,30),(5,30)\}$	$0-79$
$\{(0,0),(1,5),(5,30)\}$	$0-15, 38-55, 78-79$
$\{(0,0),(1,5),(4,30),(5,30)\}$	$0-16, 38-56, 78-79$
$\{(0,0),(1,5),(3,30),(5,30)\}$	$0-15, 38-55, 78-79$
$\{(0,0),(1,5),(3,30),(4,30),(5,30)\}$	$0-17, 38-57, 78-79$
$\{(0,0),(1,5),(2,0),(5,30)\}$	$0-15, 37-55, 77-79$
$\{(0,0),(1,5),(2,0),(4,30),(5,30)\}$	$0-16, 37-56, 77-79$
$\{(0,0),(1,5),(2,0),(3,30),(5,30)\}$	$0-15, 36-55, 76-79$

Note: See Section 7.3.2. Listed combinations of local collision relative bit differences $\mathcal{R}$ and starting step t together with any starting bit $b \in \{0,\ldots,31\}$ forms an exhaustive list of all local collisions with a single bit disturbance $\Delta Q_{t+1}[b] = \pm 1$ (no carries) with a success probability of at least 2^{-5}.

and Joux took advantage of this fact by placing all '1'-bits on bit position 1 where local collisions generally have a higher probability than at other bit positions. Due to the added bitwise rotation in the message expansion of SHA-1, disturbance vectors always have '1'-bits at different bit positions and generally more '1'-bits compared to those for SHA-0.

Consecutive '1'-bits within a word DV_t can be compressed to a single local collision [WYY05c], i.e., $DV_t[0] = DV_t[1] = 1$ is used as $\Delta W_t[0] = -1$ and $\Delta W_t[1] = +1$ which leads to the single bit disturbance $\delta W_t = 2^1 - 2^0 = +2^0$. Due to the rotations in the step function, the bits at bit positions 1 and 2 are not considered consecutive in this regard as they result in the non-consecutive bits at positions 31 and 0 after bitwise left rotation by 30. The same holds for the bit position pair (26,27) due to the bitwise left rotation by five.

7.3.3 Disturbance vector restrictions

Initially for the most straightforward collision attack, namely a single-block collision attack, three restrictions were placed on disturbance vectors:

1. $DV_{-5} = DV_{-4} = DV_{-3} = DV_{-2} = DV_{-1} = 0$: δIHV_{in} must be zero $(0,0,0,0,0)$ for a single-block collision attack. Using the disturbance vector for all steps, the '1'-bits in the words $DV_{-5},\ldots,DV_{-1}$ mark disturbances in $\delta Q_{-4},\ldots,\delta Q_0$ and thus in δIHV_{in}. Since $\delta IHV_{\text{in}} = (0,0,0,0,0)$, it follows that DV_{-5}, DV_{-4}, DV_{-3}, DV_{-2} and DV_{-1} must be zero as well;

2. $DV_{75} = DV_{76} = DV_{77} = DV_{78} = DV_{79} = 0$: this restriction is necessary to enforce that $\delta IHV_{\text{out}} = \delta IHV_{\text{in}}$. This implies that if $\delta IHV_{\text{in}} = (0,0,0,0,0)$ then also $\delta IHV_{\text{out}} = (0,0,0,0,0)$;

3. at most one of $DV_i[b]$ and $DV_{i+1}[b]$ is non-zero, for $b = 0,\ldots,31$ and $i = 0,\ldots,15$: the boolean function in the first round prevents having two consecutive local collisions in the same bit position starting in the first 16 steps.

The first and third restrictions can be alleviated by diverting from local collisions as prescribed by the disturbance vector in the first round. So instead one can construct a differential path over the first round and use local collisions for the remaining rounds for use in a (near-)collision attack. Wang et al. [WYY05b] were the first to do this.

Finally, given the use of differential paths in the first round, one can also alleviate the second restriction and construct a two-block collision attack [WYY05b]. Without these three restrictions significantly better disturbance vectors were found, thus a two-block collision attack can be more efficient than a single-block collision attack. Since the complexity of each near-collision attack contributes to the overall collision attack complexity, using more than two blocks does not offer a further advantage.

7.3.4 Disturbance vector selection

To choose the best disturbance vector, several cost functions of disturbance vectors can be used where the cost function is only applied over the last, say, 60 words that form the last three rounds.

- **Hamming weight** (e.g., [BC04, PRR05, RO05, MP05, JP05]): counts the total number of local collisions over the specified steps, since a lower number of local collisions is expected to yield a higher overall probability;
- **Bitcondition count** (e.g., [WYY05c, YIN+08]): sum of the number of bitconditions for each local collision independently (not allowing carries);
- **Probability** (e.g., [MPRR06, Man11]): product of the probabilities of all local collisions independently where carries are allowed;
- **Joint probability** (Section 7.5): the probability of fulfilling all local collisions simultaneously.

Stéphane Manuel [Man08, Man11] noticed that all interesting disturbance vectors, including all disturbance vectors used in attacks in the literature, belong to the two classes shown in Table 7-2. Within each class all disturbance vectors are forward or backward shifts and/or rotations of each other. Since a disturbance vector is an expanded message, it is uniquely determined by any 16 consecutive words. The first class named 'type I' consists of disturbance vectors $(DV_t)_{t=0}^{79}$ in which there are 15 consecutive zero words directly followed by a word DV_i with only a single bit position b set to '1', thus $DV_i = 2^b$. Such a disturbance vector is identified as disturbance vector $\mathrm{I}(i-15, b)$. The second class named 'type II' consists of disturbance vectors $(DV_t)_{t=0}^{79}$ identified as $\mathrm{II}(i, b)$ such that

$$DV_j = \begin{cases} 2^{b+31 \bmod 32} & j \in \{i+1, i+3\}; \\ 2^b & j = i+15; \\ 0 & j \in \{i, i+2, i+4, i+5, \ldots, i+14\}. \end{cases}$$

In the literature, a number of disturbance vectors reported as (near-)optimal with respect to some cost function are:

- [WYY05b]: DV I(49,2);
- [RO05]: DVs I(52,31) and DVs I(45,1), I(49,1), I(51,1)[24];
- [JP05]: DVs I(51,0), I(52,0), II(52,0);
- [PRR05]: DV I(50,2);
- [YIN+08]: DV II(56,2);
- [Man11]: DV II(52,0) (DV II(49,0) in an earlier version [Man08]).

Most of these disturbance vectors have many disturbances on bit position 31 and/or 1 as the local collisions starting at those bit positions generally have higher success probabilities. The disturbance vector used in the near-collision attack presented in Section 7.6 and [Ste10] is DV II(52,0). This choice is not based on the results in the above publications, but is entirely based on preliminary results of those presented in Section 7.5.11. In the course of writing this thesis, Manuel published an updated version [Man11] of [Man08] which supports our choice.

7.3.5 Differential path construction

As mentioned before, Wang et al. [WYY05b] were the first to construct a hand-made differential path. In 2006, De Cannière and Rechberger[CR06] introduced a more algorithmic solution to construct differential paths for SHA-0 and SHA-1 which, instead of working from two directions to each other, works similar to a probabilistic algorithm from coding theory that searches for low weight code words. Yajima et al. [YSN+07] also present a differential path construction algorithm which is similar to, but less efficient than, the one we present in Section 7.4 below.

7.3.6 History of published attacks

SHA-0 Attacks The first theoretical collision attack against SHA-0 was published by Chabaud and Joux [CJ98] with estimated attack complexity of 2^{61} SHA-0 compressions. Their results were achieved by composing local collisions such that message differences conform to the message expansion. Their work forms the basis for all subsequent attacks against SHA-0 and SHA-1.

The first practical attack against SHA-0 is the near-collision attack by Eli Biham and Rafi Chen [BC04] with an estimated complexity of 2^{40} SHA-0 compressions, which uses a message modification technique dubbed *neutral bits* by the authors. This is a technique that, given IHV, IHV' and messages M and M' that follow a differential path up to some step t, flips one or more message bit positions in M and M' simultaneously such that the altered M and M' also follow the

24. These results were obtained using a Hamming weight based cost function, thus rotated versions are considered equal. These specific rotations were chosen to avoid a situation where their analysis would always be incorrect, see section 6 of their paper.

Table 7-2: *SHA-1 disturbance vectors of type I and type II*

disturbance vector I$(K, 0)$, $K \in \mathbb{Z}$

i	DV_{K+i}	DW_{K+i}
...	...	...
-18	31	28, 31
-17	30, 31	4, 28, 29, 30, 31
-16	–	3, 4, 28, 31
-15	31	29, 30
-14	31	4, 28, 31
-13	–	4, 28, 31
-12	–	28, 31
-11	31	31
-10	–	4
-9	–	29, 31
-8	–	29
-7	31	29, 31
-6	–	4, 29
-5	31	–
-4	–	4, 29
-3	31	29
-2	–	4
-1	31	29
0	–	4
1	–	29, 31
2	–	–
3	–	29
4	–	29
$5 - 14$	–	–
15	0	0
16	–	5
17	–	0
18	1	1, 30
19	–	6, 30
20	–	1, 30
21	2	2, 31
22	–	7, 31
23	1	1, 2, 31
24	3	0, 3, 6
25	–	0, 1, 8
26	–	0, 3, 31
27	4	1, 4, 31
...	...	...

disturbance vector II$(K, 0)$, $K \in \mathbb{Z}$

i	DV_{K+i}	DW_{K+i}
...	...	...
-20	–	29
-19	31	31
-18	–	4
-17	31	–
-16	–	4, 29
-15	31	29
-14	–	4
-13	30, 31	29, 30
-12	–	3, 4
-11	–	29, 30, 31
-10	31	28, 31
-9	–	4, 28, 29
-8	–	28, 29, 31
-7	–	29
-6	–	29
-5	31	29, 31
-4	–	4
-3	–	31
-2	–	29
-1	–	29
0	–	29
1	31	31
2	–	4
3	31	–
4	–	4, 29
5	–	29, 31
6	–	–
7	–	29
8	–	29
$9 - 14$	–	–
15	0	0
16	–	5
17	–	0
18	1	1, 30
19	0	0, 6, 30
20	–	1, 5, 30
21	2	0, 2, 31
...	...	...

Note: we describe the bit-positions of all '1'-bits of the 32-bit words DV_{K+i} and DW_{K+i}. The SHA-1 (reverse) message expansion relation is used to extend the above tables forward (backward). Disturbance vectors I(K, b) and II(K, b) for $b \in \{0, \ldots, 31\}$ are obtained by left rotating all 80 words of disturbance vectors I$(K, 0)$ and II$(K, 0)$, respectively, by b bit positions [Man11].

differential path up to step t. In essence tunnels and neutral bits are very similar message modification techniques. Using their techniques they are able to find collisions for SHA-0 reduced to 65 steps.

The first (identical-prefix) collisions for full SHA-0 [BCJ+05][25], constructed using four near-collision blocks[26], were found by Biham et al. with an estimated complexity of 2^{51} SHA-0 compressions. Wang et al. [WYY05c] improved this to a collision attack consisting of two near-collision blocks with an estimated total complexity of 2^{39} SHA-0 compressions. This attack was further improved by Naito et al. [NSS+06] to approximately 2^{36} SHA-0 compressions and later by Manuel and Peyrin [MP08] to approximately $2^{33.6}$ SHA-0 compressions. These three improvements also all provide example collisions.

So far no chosen-prefix collision attacks against SHA-0 have been published.

SHA-1 Attacks The first theoretical (identical-prefix) collision attack against full SHA-1 was published by Wang et al. [WYY05b] with an estimated complexity of 2^{69} SHA-1 compressions. In their paper also a practical collision attack against SHA-1 reduced to 58 steps is presented with an example collision. They claimed have improved their theoretical attack against full SHA-1 to an estimated complexity of 2^{63} SHA-1 compressions [WYY05a, Wan06]. No publication has followed so far however, instead [Coc07] has reconstructed and confirmed parts of the analysis of their attack given the details presented at the CRYPTO2005 rump session.

The paper [BCJ+05], besides collision attacks against full SHA-0, also presented collisions for 40-step SHA-1 with an attack complexity of approximately 2^{57} SHA-1 compressions. De Cannière and Rechberger[CR06] were able to construct a practical collision attack against 64-step SHA-1 and provide example collisions. De Cannière et al. [CMR07] were able to find collisions for 70-step SHA-1 in 2007. Collisions for 73-step SHA-1 were presented in [Gre10]. So far no collisions have been presented for a larger number of steps of SHA-1. In particular, though anticipated since 2006, no actual collisions for SHA-1 have been found.

At the rump session of CRYPTO2007, Mendel et al. [MRR07] claimed to have a collision attack against full SHA-1 with an estimated complexity of $2^{60.x}$ SHA-1 compressions and started a distributed computing project. No further publication has followed and their distributed computing project was stopped mid 2009.

Rafael Chen claims an identical-prefix collision attack against full SHA-1 in his PhD thesis [Che11] with an estimated complexity of 2^{58}. However, in his complexity estimation he extends a single 72-step SHA-1 near-collision attack

25. Announced at the rump session of Crypto'04

26. It would have been more efficient to use only two near-collision blocks. However, lacking differential path construction algorithms, the authors used a linearized model of SHA-1 to deal with the first few steps of differential paths. At least four near-collision blocks were needed for a collision attack within this model and using an additional technique.

of complexity $2^{53.1}$ to a two-block identical-prefix collision attack against full SHA-1 with an additional factor of only $2^{4.9}$. It can be easily verified that the highest success probability over the last 8 steps that can be achieved for the second near-collision block (that targets one specific δIHV) is about $2^{-8.356}$, thus there is an error in the complexity estimation of at least a factor $2^{3.5}$.[27]

There are two other papers that have to be mentioned here, namely [MHP09] by McDonald et al. in which the authors claimed to have found a differential path leading to a collision attack against full SHA-1 with an estimated complexity of 2^{52} SHA-1 compressions. Their result was directly based on the disturbance vector analysis and claimed possible collision attack complexities of Stéphane Manuel [Man08]. McDonald et al. decided to withdraw their paper when later analysis indicated that the claimed possible collision attack complexities in [Man08] were inaccurate. In the journal version [Man11] of [Man08] a more detailed analysis of disturbance vectors is made using a more conservative cost function and any claims towards possible collision attack complexities have been removed.

The literature on SHA-1 does not represent the state-of-the-art cryptanalytic methods as several claims have not been substantiated by publications. Moreover, due to lack of details it is hard if not impossible to verify the correctness and accurateness of the above claimed attack complexities and/or compare them, thus it is unclear which attack should be considered as the best correct collision attack against SHA-1. RFC6194 [PCTH11] considers the first collision attack by Wang et al. [WYY05b] with estimated complexity 2^{69} SHA-1 compressions as the best (identical-prefix) collision attack against full SHA-1. This attack is based on two near-collisions attacks with a complexity of about 2^{68} SHA-1 compressions each.

So far no chosen-prefix collision attacks against SHA-1 have been published. No implementations of (near-)collision attacks against SHA-1 have been published to date, except the implementation of the near-collision attack described in this thesis [HC].[28]

27. Exact probability can be determined using the methods presented in Section 7.5, estimations can easily be obtained using a Monte Carlo simulation. The given error factor of $2^{3.5}$ does not take into account the complexity of the first near-collision block and the fact that the second near-collision block is slightly harder than $2^{53.1}$.

28. Necessary for public verification of correctness and the actual runtime complexity. Also aids in further understanding and allows further improvements by the cryptographic community.

7.4 Differential path construction

In this section we present differential path construction algorithms for SHA-0 and SHA-1 based on the algorithms presented in Chapter 5. Similar to MD5, the SHA-0 and SHA-1 differential path construction algorithms are improved versions that make use of bitconditions (see Section 6.2.2). Also, the final connect algorithm operates in a bit-wise manner similar to the algorithm in Section 6.2.5 instead of a word-wise manner as in Chapter 5.

Differential path construction algorithms for SHA-0 and SHA-1 have already been proposed by De Cannière and Rechberger [CR06] and Yajima et al. [YSN+07] each using a different methodology. The first uses an approach based on a probabilistic algorithm from coding theory for finding low weight codewords. The second is very similar to the algorithms proposed in this section where from two ends partial differential paths are constructed towards each other. Our algorithm improves over that of [YSN+07] on efficiency: our connection algorithm operates in a bit-wise manner and therefore it is able to stop earlier in case of impossible forward and backward differential path pairs.

Compared to MD5, the rotations of Q_i that are given as input to the boolean function in the step update $f_t(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30))$ (see Eq. 7.5) complicate matters, since for both SHA-0 and SHA-1 indirect bitconditions on $Q_t[i]$ can involve one of the following bits: $Q_{t-1}[i]$, $Q_{t+1}[i]$, $Q_{t-1}[i+2 \bmod 32]$, $Q_{t-2}[i+2 \bmod 32]$, $Q_{t+1}[i-2 \bmod 32]$ and $Q_{t+2}[i-2 \bmod 32]$.

However, this is simpler in the first round as the first round boolean function of both SHA-0 and SHA-1 (ignoring the input rotations) is identical to MD5's first round boolean function. As can be seen in Table C-1, indirect bitconditions are only applied to the second input involving the third input or vice versa. This implies for SHA-0 and SHA-1 that in the first round any indirect bitcondition on $Q_t[i]$ can involve only $Q_{t-1}[i]$ or $Q_{t+1}[i]$, thus never bit positions other than i. Therefore the improved algorithm for MD5 in Section 6.2.5 can be more easily adapted to SHA-0 and SHA-1 over the first round. This limitation to the first round does not pose a problem for constructing a near-collision attack, since for the remaining rounds the differential path is the result of interleaving local collisions.

7.4.1 Differential paths

Similar to MD5, a differential path for SHA-0 or SHA-1 is described using bitconditions $\mathfrak{q}_t = (\mathfrak{q}_t[i])_{i=0}^{31}$ on (Q_t, Q'_t), where each bitcondition $\mathfrak{q}_t[i]$ specifies a restriction on the bits $Q_t[i]$ and $Q'_t[i]$ possibly including values of other bits $Q_l[i]$ for $l \in \{t-1, t+1\}$. For the first round only the following bitconditions are necessary and used: '`.`', '`+`', '`-`', '`0`', '`1`', '`^`', '`v`', '`!`' and '`y`' (see Tables 6-1 and 6-2), where the last two bitconditions are only used by message modification techniques (see Section 7.6.8). Since the disturbance vector determines all bit differences $(DW_t)_{t=0}^{79}$ only up to their sign, the actual chosen differences $(\Delta\widehat{W}_t)_{t=0}^{19}$ are maintained besides the bitconditions in the differential path.

A partial differential path for SHA-0 or SHA-1 over steps $t = t_b, \ldots, t_e$ can be seen as a $(t_e - t_b + 6) \times 32$ matrix $(\mathfrak{q}_t)_{t=t_b-4}^{t_e+1}$ of bitconditions paired with message difference vector $(\Delta\widehat{W}_t)_{t=t_b}^{t_e}$. The bitconditions are used to specify the values of ΔQ_t and ΔF_t. As for each step t only $\Delta Q_{t-3}, \Delta Q_{t-2}, \Delta Q_{t-1}, \Delta Q_t$ are required in a differential step, in such a partial differential path $\mathfrak{q}_{t-4}$ and $\mathfrak{q}_{t_e+1}$ are used only to represent $\delta RL(Q_{t_b-4}, 30)$ and δQ_{t_e+1} instead of BSDRs.

7.4.2 Forward

Suppose we have a partial differential path consisting of at least bitconditions $\mathfrak{q}_{t-3}$, $\mathfrak{q}_{t-2}$ and the differences ΔQ_{t-4}, ΔQ_{t-1} and δQ_t are known. We want to extend this partial differential path forward with step t resulting in the differences δQ_{t+1}, ΔW_t, ΔQ_t, bitconditions $\mathfrak{q}_{t-1}$ and additional bitconditions $\mathfrak{q}_{t-2}$ and $\mathfrak{q}_{t-3}$. We use an adaptation of the algorithm from Section 5.6.1 to perform such a forward extension using bitconditions (see also Section 6.2.2).

If the BSDR ΔQ_{t-1} is only used in previous steps to determine δQ_{t-1} and $\delta RL(Q_{t-1}, 5)$ then one can replace ΔQ_{t-1} by any low weight BSDR $\delta\widehat{Q}_{t-1}$ of δQ_{t-1} such that $\delta RL(Q_{t-1}, 5) = \sigma(RL(\delta\widehat{Q}_{t-1}, 5))$. Otherwise $\Delta\widehat{Q}_{t-1} = \Delta Q_{t-1}$. The BSDR $\Delta\widehat{Q}_{t-1}$ directly translates to bitconditions $\mathfrak{q}_{t-1}$ as in Table 6-1.

We select ΔQ_t based on the value of δQ_t. Since upcoming steps can use the remaining freedom in ΔQ_t, we choose for each $Z \in dRL(\delta Q_t, 5)$ (see Lemma 5.4) at most one ΔQ_t such that $\sigma(RL(\Delta Q_t, 5)) = Z$. We continue with any one of these ΔQ_t, preferably one with low weight. We choose a ΔW_t such that $\Delta W_t[i] \in \{-DW_t[i], +DW_t[i]\}$ for $i = 0, \ldots, 31$.

We assume that all indirect bitconditions in $\mathfrak{q}_{t-3}$ are forward and involve only bits of Q_{t-2} and that $\mathfrak{q}_{t-2}$ consists of only direct bitconditions.[29] To determine the differences $\Delta F_t = (g_i)_{i=0}^{31}$ we proceed as follows. For $i = 0, \ldots, 31$ we assume that we have valid bitconditions

$$(\mathfrak{a}, \mathfrak{b}, \mathfrak{c}) = (\mathfrak{q}_{t-1}[i],\ \mathfrak{q}_{t-2}[i + 2 \bmod 32],\ \mathfrak{q}_{t-3}[i + 2 \bmod 32]),$$

where only $\mathfrak{c}$ can be indirect and if so involves $Q_{t-2}[i + 2 \bmod 32]$ associated with $\mathfrak{b}$. Hence, in the notation of Section 6.2.2: $(\mathfrak{a}, \mathfrak{b}, \mathfrak{c}) \in \mathcal{L}$. If $|V_{t,\mathfrak{abc}}| = 1$ then there is no ambiguity and we set $g_i = V_{t,\mathfrak{abc}}$ and $(\widehat{\mathfrak{a}}, \widehat{\mathfrak{b}}, \widehat{\mathfrak{c}}) = (\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$. Otherwise, if $|V_{t,\mathfrak{abc}}| > 1$, then we choose g_i arbitrarily from $V_{t,\mathfrak{abc}}$ and we resolve the ambiguity in $\Delta F_t[i]$ by replacing bitconditions $(\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$ by $(\widehat{\mathfrak{a}}, \widehat{\mathfrak{b}}, \widehat{\mathfrak{c}}) = FC(t, \mathfrak{abc}, g_i)$. Note that in the next step $t + 1$ our assumptions hold again as both $\widehat{\mathfrak{a}}$ and $\widehat{\mathfrak{b}}$ are direct bitconditions.

29. This assumption is valid for the results of this algorithm for the previous step $t - 1$. However this algorithm requires valid inputs for the first step, e.g., $t = 0$. E.g, we use the values $\Delta Q_{-4} = \Delta Q_{-1} = (0)_{i=0}^{31}$, $\mathfrak{q}_{-3} = \mathfrak{q}_{-2} = (\text{'}\cdot\text{'})_{i=0}^{31}$ and $\delta Q_0 = 0$ to represent identical but unknown intermediate hash values $IHV_{\text{in}} = IHV'_{\text{in}}$. Another possibility is to use the values $\mathfrak{q}_{-4}, \ldots, \mathfrak{q}_0$ consisting of direct bitconditions '0', '1', '+', '-' that represent given values IHV_{in} and IHV'_{in}. In this case, the forward construction algorithm can skip choosing BSDRs ΔQ_{-1} and ΔQ_0 and translating them to $\mathfrak{q}_{-1}$ and $\mathfrak{q}_0$ in steps $t = 0$ and $t = 1$, respectively.

Once all g_i and thus ΔF_t have been determined, δQ_{t+1} is determined as

$$\delta Q_{t+1} = \sigma(\Delta F_t) + \sigma(\Delta W_t) + \sigma(RL(\Delta Q_t, 5)) + \sigma(RL(\Delta Q_{t-4}, 5)).$$

7.4.3 Backward

Similar to the forward extension, we now consider the backward extension of a partial differential path. Suppose we have a partial differential path consisting of at least bitconditions $\mathfrak{q}_{t-2}, \mathfrak{q}_{t-1}$ and differences $\delta RL(Q_{t-3}, 30)$, ΔQ_t and δQ_{t+1} are known. We want to extend this partial differential path backward with step t resulting in the differences $\delta RL(Q_{t-4}, 30)$, ΔW_t, bitconditions $\mathfrak{q}_{t-3}$ and additional bitconditions $\mathfrak{q}_{t-1}$ and $\mathfrak{q}_{t-2}$. We use an adaptation of the algorithm from Section 5.6.2 to perform such a backward extension using bitconditions.

We choose a ΔW_t such that $W_t[i] \in \{-DW_t[i], +DW_t[i]\}$ for $i = 0, \ldots, 31$. We select ΔQ_{t-3} based on the value of $\delta RL(Q_{t-3}, 30)$. We choose any low weight BSDR Z of $\delta RL(Q_{t-3}, 30)$, so that $\Delta Q_{t-3} = RR(Z, 30)$ which then translates into a possible $\mathfrak{q}_{t-3}$ as in Table 6-1.

The differences $\Delta F_t = (g_i)_{i=0}^{31}$ are determined by assuming for $i = 0, \ldots, 31$ that we have valid bitconditions

$$(\mathfrak{a}, \mathfrak{b}, \mathfrak{c}) = (\mathfrak{q}_{t-1}[i],\ \mathfrak{q}_{t-2}[i + 2 \bmod 32],\ \mathfrak{q}_{t-3}[i + 2 \bmod 32]),$$

where only $\mathfrak{a}$ can be indirect and if so it involves $Q_{t-2}[i]$.[30] Note that $Q_{t-2}[i]$ is *not* associated with $\mathfrak{b}$. To deal with this issue, we first ignore such indirect bitconditions and reapply them later on.

We set $\widetilde{\mathfrak{a}} = \mathfrak{a}$ if $\mathfrak{a}$ is a direct bitcondition, otherwise $\widetilde{\mathfrak{a}} =$ '.'. It follows that $(\widetilde{\mathfrak{a}}, \mathfrak{b}, \mathfrak{c}) \in \mathcal{L}$. If $|V_{t,\widetilde{\mathfrak{a}}\mathfrak{b}\mathfrak{c}}| = 1$ then there is no ambiguity and we set $\{g_i\} = V_{t,\widetilde{\mathfrak{a}}\mathfrak{b}\mathfrak{c}}$ and $(\widehat{\mathfrak{a}}, \widehat{\mathfrak{b}}, \widehat{\mathfrak{c}}) = (\widetilde{\mathfrak{a}}, \mathfrak{b}, \mathfrak{c})$. Otherwise, if $|V_{t,\widetilde{\mathfrak{a}}\mathfrak{b}\mathfrak{c}}| > 1$, then we choose g_i arbitrarily from $V_{t,\widetilde{\mathfrak{a}}\mathfrak{b}\mathfrak{c}}$ and we resolve the ambiguity by replacing bitconditions $(\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$ by $(\widehat{\mathfrak{a}}, \widehat{\mathfrak{b}}, \widehat{\mathfrak{c}}) = BC(t, \widetilde{\mathfrak{a}}\mathfrak{b}\mathfrak{c}, g_i)$. Note that in the next step $t - 1$ our assumptions hold again as $\widehat{\mathfrak{c}}$ is a direct bitcondition and $\widehat{\mathfrak{b}}$ is either a direct bitcondition or an indirect backward bitcondition involving $\widehat{\mathfrak{c}}$.

Finally, for all i such that $\mathfrak{q}_{t-1}[i]$ was an indirect bitcondition, we reapply this bitcondition. This means that if the new $\widehat{\mathfrak{q}}_{t-1}[i]$='.' then we revert it to the old value of $\mathfrak{q}_{t-1}[i]$. Otherwise, it must be either '0' or '1', since $\widehat{\mathfrak{a}}$ cannot be an indirect bitcondition (see Table C-1). If $\widehat{\mathfrak{q}}_{t-2}[i] \in \{\text{'.'}, \widehat{\mathfrak{q}}_{t-1}[i]\}$ then we replace it by $\widehat{\mathfrak{q}}_{t-1}[i]$, otherwise a contradiction has arisen and other choices have to be tried.

Once all g_i and thus ΔF_t are determined, $\delta RL(Q_{t-4}, 30)$ is determined as

$$\delta RL(Q_{t-4}, 30) = \delta Q_{t+1} - \sigma(\Delta F_t) - \sigma(\Delta W_t) - \sigma(RL(\Delta Q_t, 5)).$$

30. Again this assumption is met by the previous step $t + 1$ of the algorithm. Nevertheless the first call to the algorithm for, e.g., $t = 19$ requires valid inputs.

7.4.4 Connect

Construction of a full differential path can be done as follows. Assume that the forward construction has been carried out up to some step t. Furthermore, assume that the backward construction has been carried out down to step $t+6$. For our near-collision attack we used a low value for t as explained in Section 7.6.4 (p. 168). For each combination of forward and backward partial differential paths thus found, this leads to bitconditions $\ldots$, $\mathfrak{q}_{t-3}$, $\mathfrak{q}_{t-2}$, $\mathfrak{q}_{t-1}$, and $\mathfrak{q}_{t+3}$, $\mathfrak{q}_{t+4}$, $\mathfrak{q}_{t+5}$, $\mathfrak{q}_{t+6}, \ldots$ and differences ΔQ_t, δQ_{t+1}, $\delta RL(Q_{t+2}, 30)$. As more thoroughly explained at the end of this section, we replace all backward indirect bitconditions '^' by '.' to ensure correctness. Later on, we reapply all such removed backward indirect bitconditions.

It remains to try and glue together each of these combinations by finishing steps $t+1, \ldots, t+5$ until a full differential path is found. We use an adaptation of the algorithm in Section 5.6.3 which uses bitconditions and operates in a bit-wise manner instead of a word-wise manner. Due to the bitwise left-rotations over 30 bit positions in the step function, the description of the algorithm for SHA-0 and SHA-1 is more complicated compared to MD5. We first present a sketch of our algorithm that deals with the core principles, followed by a more precise definition.

Similar to MD5, all values for δQ_i are known and the goal is to find bitconditions and differences $\delta W_{t+1}, \ldots, \delta W_{t+5}$ such that some target values $\delta F_{t+1}, \ldots, \delta F_{t+5}$ are obtained:

$$\delta Q_{i+1} = \sigma(RL(\Delta Q_i, 5)) + \sigma(RL(\Delta Q_{i-4}, 30)) + \delta F_i + \delta W_i, \quad i \in \{t+1, \ldots, t+5\}. \tag{7.10}$$

We have some amount of freedom in choosing ΔQ_t, ΔQ_{t+1} and ΔQ_{t+2} as long as they remain compatible with the known values of δQ_{t+i}, $\delta RL(Q_{t+i}, 5)$ and $\delta RL(Q_{t+i}, 30)$ as described later on.

Due to the bitwise left-rotation over 30 bit position it follows that bit position 0 of step i depends on bitcondition $\mathfrak{q}_t[2]$ which is treated at bit position 2 of step $i-1$ for $i \in \{t+2, \ldots, t+5\}$. This issue is dealt with by using 40 imaginary bit positions $b \in \{0, \ldots, 39\}$ and the connect algorithm first searches for correct values at bit position $b = 0$ and then iteratively extends to higher bit positions. For each successful extension to the last bit position $b = 39$, one finds at least one valid full differential path.

At each bit position $b \in \{0, \ldots, 39\}$, the algorithm considers step $t+1+j$ at bit position $b-2j$ for $j \in \{0, \ldots, 4\}$ if and only if $b-2j \in \{0, \ldots, 31\}$. Whenever the algorithm considers step $t+1+j$ for $j \in \{0, \ldots, 4\}$ at bit position $b-2j \in \{0, \ldots, 31\}$, it does the following:

1. If $j \in \{0, 1, 2\}$ then it first selects a value for $\Delta Q_{t+j}[b-2j]$ that is compatible with the three known differences δQ_{t+j}, $\delta RL(Q_{t+j}, 5)$ and $\delta RL(Q_{t+j}, 30)$.

2. Next (if $j \in \{0, \ldots, 4\}$), it searches for a value for $\Delta W_{t+j+1}[b-2j]$ and bitconditions $\mathfrak{q}_{t+j}[b-2j]$, $\mathfrak{q}_{t+j-1}[b-2j+2 \bmod 32]$ and $\mathfrak{q}_{t+j-2}[b-2j+2 \bmod 32]$. These bitconditions must be compatible with all bitconditions known up to this point.

Furthermore, they must unambiguously lead to some value $\Delta F_{t+j+1}[b-2j]$ that is 'compatible' with Equation 7.10 (using $i = t + j + 1$).

3. For each such resulting tuple of values $\Delta Q_{t+j}[b-2j]$, $\Delta W_{t+j+1}[b-2j]$, $\mathfrak{q}_{t+j}[b-2j]$, $\mathfrak{q}_{t+j-1}[b-2j+2 \bmod 32]$ and $\mathfrak{q}_{t+j-2}[b-2j+2 \bmod 32]$, the algorithm continues with the next step $t+2+j$ at bit position $b-2j-2$ if $j < 4$ and $b-2j-2 \in \{0,\ldots,31\}$.

Now we give a more precise definition of the connection algorithm. First, we choose a low weight BSDR $\Delta\widetilde{Q}_{t+1}$ of δQ_{t+1} and a low weight BSDR $\Delta\widetilde{Q}_{t+2}$ such that $\sigma(RL(\Delta\widetilde{Q}_{t+2}, 30)) = \delta RL(Q_{t+2}, 30)$. Then we determine target values for FW_k which can be seen as the target value for $\delta F_k + \delta W_k$ for $k = t+1,\ldots,t+5$:

$$FW_k = \sigma(\Delta\widetilde{Q}_{k+1}) - \sigma(RL(\Delta\widetilde{Q}_k, 5)) - \sigma(RL(\Delta\widetilde{Q}_{k-4}, 30)). \tag{7.11}$$

So far we can choose any ΔQ_t, ΔQ_{t+1} and ΔQ_{t+2} under the following requirements so that Equation 7.11 holds:

$$\sum_{b=b_l}^{b_h} 2^b \Delta Q_k[b] = \sum_{b=b_l}^{b_h} 2^b \Delta\widetilde{Q}_k[b] \quad (\text{in } \mathbb{Z}),$$

for $(b_l, b_h) \in \{(0,1),(2,26),(27,31)\}$ and $k \in \{t, t+1, t+2\}$.

We aim to complete the differential path by searching for new bitconditions $\mathfrak{q}_{t-3},\ldots,\mathfrak{q}_{t+6}$ that are compatible with the differential steps from the forward and backward construction, and by finding message word differences $\Delta W_{t+1},\ldots,\Delta W_{t+5}$ such that the following equation holds for $k = t+1,\ldots,t+5$:

$$\delta Q_{k+1} - \sigma(RL(\Delta Q_k, 5)) - \sigma(RL(\Delta Q_{k-4}, 30)) = FW_k = \delta F_k + \delta W_k.$$

An efficient way to find these new bitconditions is to first test if they exist, and if so to backtrack to actually construct them. For $i = 0, 1, \ldots, 40$ we attempt to construct a set $\mathcal{U}_i$ consisting of all tuples

$$(q_0, q_1, q_2, fw_1, fw_2, fw_3, fw_4, fw_5, (\mathfrak{q}_j[b])_{(j,b)\in A_i}),$$

where $q_0, q_1, q_2 \in \mathbb{Z}_{2^{32}}$ and $fw_1, fw_2, fw_3, fw_4, fw_5 \in \mathbb{Z}_{2^{32}}$ and A_i is a later to be defined constant set, such that:

1. $q_j \equiv 0 \bmod 2^{\min(32,\max(0,i-2j))}$ and $fw_j \equiv 0 \bmod 2^{\min(32,\max(0,i-2j-2))}$;

2. there exist bitconditions, compatible with the forward and backward differential paths and the bitconditions $(\mathfrak{q}_j[b])_{(j,b)\in A_i}$, that uniquely determine the $\Delta Q_j[b]$ and $\Delta F_j[b]$ below and BSDRs ΔW_k for which $W_k[i] \in \{-DW_k[i], +DW_k[i]\}$ for $k = t+1,\ldots,t+5$ and $i = 0,\ldots,31$ such that

$$\delta Q_{t+j} = q_j + \sum_{\ell=0}^{\theta(i-2j)} 2^\ell \Delta Q_{t+j}[\ell], \qquad j \in \{0,1,2\}; \tag{7.12}$$

$$FW_{t+j} = fw_j + \sum_{\ell=0}^{\theta(i-2j-2)} 2^\ell (\Delta F_{t+j}[\ell] + \Delta W_{t+j}[\ell]), \qquad j \in \{1,2,3,4,5\}; \tag{7.13}$$

where $\theta(j) = \min(32, \max(0, j))$.

The set A_i informally consists of all indices (j, b) for which $\mathfrak{q}_j[b]$ may have been modified by the construction of previous $\mathcal{U}_\ell$ for $\ell = 0, \ldots, i-1$ and for which the construction of upcoming $\mathcal{U}_\ell$ for $\ell = i+1, \ldots, 40$ depends on $\mathfrak{q}_j[b]$. This implies $A_0 = A_{40} = \emptyset$. The sets $A_1, \ldots, A_{39}$ are defined for $i = 1, \ldots, 39$ as:

$$\begin{aligned}
A_i &= \bigcup_{j \in \{1,2,3,4\}} \left(A_{i,j}^{(1)} \cup A_{i,j}^{(2)} \cup A_{i,j}^{(3)} \cup A_{i,j}^{(4)} \right), \\
A_{i,j}^{(1)} &= \left\{ (t+j-1, 0) \;\middle|\; i - 2j + 1 \in \{0, \ldots, 31\} \right\}, \\
A_{i,j}^{(2)} &= \left\{ (t+j-1, 1) \;\middle|\; i - 2j \in \{0, \ldots, 31\} \right\}, \\
A_{i,j}^{(3)} &= \left\{ (t+j-2, \ell + 2 \bmod 32) \;\middle|\; \ell = i - 2j + 1 \in \{0, \ldots, 31\} \right\}, \\
A_{i,j}^{(4)} &= \left\{ (t+j-2, \ell + 2 \bmod 32) \;\middle|\; \ell = i - 2j \in \{0, \ldots, 31\} \right\}.
\end{aligned}$$

From these conditions it follows that $\mathcal{U}_0$ must be chosen as

$$\{(\widetilde{q}_0, \widetilde{q}_1, \widetilde{q}_2, FW_{t+1}, FW_{t+2}, FW_{t+3}, FW_{t+4}, FW_{t+5}, \emptyset)\}, \tag{7.14}$$

where $\widetilde{q}_0 = \sigma(\Delta Q_t)$, $\widetilde{q}_1 = \sigma(\Delta Q_{t+1})$ and $\widetilde{q}_2 = \sigma(\Delta Q_{t+2})$. Algorithm 7-1 (p. 136–139) informally does the following to construct $\mathcal{U}_{i+1}$:

7-1. If $0 \le i < 32$ then step $t+1$ at bit i is processed (otherwise proceed directly to 7-1-a.): First a valid value for $\Delta W_{t+1}[i]$ and a valid differential bitcondition ('-','.' or '+') for $Q_t[i]$ are chosen such that Equation 7.12 holds. Next all possible boolean function differences $\Delta F_{t+1}[i]$ using $\mathfrak{q}_t[i]$, $\mathfrak{q}_{t-1}[i + 2 \bmod 32]$ and $\mathfrak{q}_{t-2}[i + 2 \bmod 32]$ such that Equation 7.13 holds are considered. Perform subroutine 7-1-a below for each possible choice.

7-1-a. If $0 \le i - 2 < 32$ then step $t+2$ at bit $i-2$ is processed (otherwise directly proceed to 7-1-b.): First a valid value for $\Delta W_{t+2}[i-2]$ and a valid differential bitcondition ('-','.' or '+') for $Q_{t+1}[i-2]$ are chosen such that Equation 7.12 holds. Next all possible boolean function differences $\Delta F_{t+2}[i-2]$ using $\mathfrak{q}_{t+1}[i-2]$, $\mathfrak{q}_t[i \bmod 32]$ and $\mathfrak{q}_{t-1}[i \bmod 32]$ such that Equation 7.13 holds are considered. Perform subroutine 7-1-b below for each possible choice.

7-1-b. If $0 \le i-4 < 32$ then step $t+3$ at bit $i-4$ is processed (otherwise directly proceed to 7-1-c.): First a valid value for $\Delta W_{t+3}[i-4]$ and a valid differential bitcondition ('-','.' or '+') for $Q_{t+2}[i-4]$ are chosen such that Eq. 7.12 holds. Next all possible boolean function differences $\Delta F_{t+3}[i-4]$ using $\mathfrak{q}_{t+2}[i-4]$, $\mathfrak{q}_{t+1}[i-2 \bmod 32]$

and $\mathfrak{q}_t[i-2 \bmod 32]$ such that Equation 7.13 holds are considered. Perform subroutine 7-1-c below for each possible choice.

7-1-c. If $0 \leq i-6 < 32$ then step $t+4$ at bit $i-6$ is processed (otherwise directly proceed to 7-1-d.): First a valid value for $\Delta W_{t+4}[i-6]$ is chosen. Next all possible boolean function differences $\Delta F_{t+4}[i-6]$ using $\mathfrak{q}_{t+3}[i-6]$, $\mathfrak{q}_{t+2}[i-4 \bmod 32]$ and $\mathfrak{q}_{t+1}[i-4 \bmod 32]$ such that Equation 7.13 holds are considered. Perform subroutine 7-1-d below for each possible choice.

7-1-d. If $0 \leq i-8 < 32$ then step $t+5$ at bit $i-8$ is processed (otherwise save the resulting new tuple in $\mathcal{U}_{i+1}$): First a valid value for $\Delta W_{t+5}[i-8]$ is chosen. Next all possible boolean function differences $\Delta F_{t+5}[i-8]$ using $\mathfrak{q}_{t+4}[i-8]$, $\mathfrak{q}_{t+3}[i-6 \bmod 32]$ and $\mathfrak{q}_{t+2}[i-6 \bmod 32]$ such that Equation 7.13 holds are considered and for each saves the resulting new tuple in $\mathcal{U}_{i+1}$.

For $i = 1, \ldots, 40$, we use Algorithm 7-1 (pp. 136–139) to construct $\mathcal{U}_i$ based on $\mathcal{U}_{i-1}$. As soon as we encounter an i for which $\mathcal{U}_i = \emptyset$, we know that the desired differential path cannot be constructed from this combination of forward and backward partial differential paths, and that we should try another combination. If, however, we find $\mathcal{U}_{40} \neq \emptyset$ then it must be the case that $\mathcal{U}_{40} = (0,0,0,0,0,0,0,0,\emptyset)$. Furthermore, in that case, every set of bitconditions that leads to this non-empty $\mathcal{U}_{40}$ gives rise to a full differential path. Thus if $\mathcal{U}_{40} \neq \emptyset$, there exists at least one valid trail $u_0, \ldots, u_{40}$ with $u_i \in \mathcal{U}_i$ and where u_{i+1} is a tuple resulting from u_i in Algorithm 7-1. For each valid trail, the desired new bitconditions $\mathfrak{q}_{t-2}, \ldots, \mathfrak{q}_{t+4}$ can be found as $\mathfrak{g}', \mathfrak{f}'_0, \mathfrak{e}'_0, \mathfrak{d}'_0, \mathfrak{c}'_0, \mathfrak{b}''_2, \mathfrak{a}'$ for bits $i+2, i, i-2, i-4, i-6, i-6, i-8 \pmod{32}$, respectively and if applicable, for $i = 0, \ldots, 39$ in Algorithm 7-1.[31]

Finally, there remains an issue that has not been dealt with so far, namely that $(\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$ must be in $\mathcal{L}$ (see Section 6.2.2) for every occurrence of the form $FC(j, \mathfrak{abc}, z)$ in Algorithm 7-1. The connection algorithm works forward, which implies in the same way as in Section 7.4.2 that $\mathfrak{b}$ is a direct bitcondition and $\mathfrak{c}$ is either a direct bitcondition or a forward indirect bitcondition involving $\mathfrak{b}$. If the bitcondition $\mathfrak{a}$ is indirect then $(\mathfrak{a}, \mathfrak{b}, \mathfrak{c})$ cannot be in $\mathcal{L}$, since $\mathfrak{a}$ involves bitcondition other than $\mathfrak{b}$ and $\mathfrak{c}$. Thus $\mathfrak{a}$ must be a direct bitcondition. However, $\mathfrak{a}$ may come from the backward partial differential path and thus may be a backward indirect bitcondition. To resolve this issue, we replace all backward indirect bitconditions '^' by '.' before running Algorithm 7-1. We reapply all such removed backward indirect bitconditions '^' on $Q_j[b]$ to the differential paths resulting from the above procedure in the following manner. Note that $\mathfrak{q}_j[b]$ must be either '.', '0' or '1', since the only step that could have resulted in $\mathfrak{q}_j[b] =$'v' comes after the connection steps $t+1, \ldots, t+5$. If $\mathfrak{q}_j[b] =$'.' then we set $\mathfrak{q}_j[b] =$'^' to reapply the backward bitcondition. If $\mathfrak{q}_j[b] \in \{$'0','1'$\}$ and $\mathfrak{q}_{j-1}[b] \in \{$'.','v',$\mathfrak{q}_j[b]\}$ then we set $\mathfrak{q}_{j-1}[b] = \mathfrak{q}_j[b]$ to reapply the backward

31. Note that step $t+5$ at bit position $i-8$ determines three of these final bitconditions, namely $\mathfrak{q}_{t+2}[i-6 \bmod 32]$, $\mathfrak{q}_{t+3}[i-6 \bmod 32]$ and $\mathfrak{q}_{t+4}[i-8 \bmod 32]$. Furthermore, the lower steps $t+1$, $t+2$, $t+3$ and $t+4$ at bit position j determine only one final bitcondition, namely $\mathfrak{q}_{t-2}[j+2 \bmod 32]$, $\mathfrak{q}_{t-1}[j+2 \bmod 32]$, $\mathfrak{q}_t[j+2 \bmod 32]$ and $\mathfrak{q}_{t+1}[j+2 \bmod 32]$, respectively. This explains the double $i-6$.

bitcondition. If both options above do not hold then a contradiction has arisen and the full differential path cannot be valid.

For an example full differential path constructed with the above algorithm see Table 7-6 (p. 169).

7.4.5 Complexity

The complexity to construct valid differential paths for SHA-1 depends on many factors as is also explained in Sections 5.6.4 and 6.2.6. In the case of SHA-1, the complexity of the connection algorithm also depends on the number of possible message word differences on those five steps. However, the choice of which five steps to use for the connection algorithm depends mostly on the particular choice of the disturbance vector so as to leave maximal freedom for message modification techniques.

A rough approximation for the complexity to construct the differential path for our near-collision attack in Section 7.6 is the equivalent of 2^{43} SHA-1 compression function calls. This is significantly larger than the differential path construction for MD5. Nevertheless, it is also significantly smaller than the lowest complexity claimed (and withdrawn) so far for a SHA-1 collision attack. This complexity is based on our choices for the disturbance vector II(52,0), the five connecting steps, amount of freedom left for message modification techniques and the maximum number of bitconditions in the first round. It should be clear that for other choices the complexity of constructing differential paths can be smaller or larger.

Our implementations of our differential path construction algorithms for SHA-1 are published as part of project HashClash [HC].

Algorithm 7-1 Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$ for SHA-0 and SHA-1.

Assume $\mathcal{U}_i$ is constructed inductively by means of this algorithm. For each tuple $(q_0, q_1, q_2, fw_1, fw_2, fw_3, fw_4, fw_5, (\mathfrak{q}_j[b])_{(j,b)\in A_i}) \in \mathcal{U}_i$ do the following:†

1. Let $\mathcal{U}_{i+1} = \emptyset$ and $\widehat{\mathfrak{q}}_j[b] = \mathfrak{q}_j[b]$ for $(j, b) \in A_i$.‡
2. If $i \geq 32$ then
3. Let $\mathfrak{e}'_2 = \mathfrak{q}_{t+1-1}[i]$, $\widehat{q}_0 = q_0$ and $\widehat{fw}_1 = fw_1$.
4. Proceed with subroutine step2 (Algorithm 7-1-a, p. 137)
5. Else
6. For each different $q'_0 \in \{-q_0[i], +q_0[i]\}$ do
7. Let $\widehat{q}_0 = q_0 - 2^i q'_0$.
8. If $i \in \{1, 26, 31\}$ and $\widehat{q}_0 \neq \sum_{b=i+1}^{31} 2^b \Delta\widetilde{Q}_t[b]$ then skip steps 9-16.
9. Let $\mathfrak{e}_2 =$ '$\mathtt{-}$', '$\mathtt{.}$' or '$\mathtt{+}$' based on whether $q'_0 = -1$, 0 or $+1$.
10. Let $\mathfrak{f}_2 = \mathfrak{q}_{t+1-2}[i + 2 \bmod 32]$ and $\mathfrak{g} = \mathfrak{q}_{t+1-3}[i + 2 \bmod 32]$.
11. For each different $w'_1 \in \{-DW_{t+1}[i], +DW_{t+1}[i]\}$ do
12. Let $Z_1 = fw_1 - 2^i w'_1$
13. For each different $z'_1 \in \{-Z_1[i], Z_1[i]\} \cap V_{t+1,\mathfrak{e}_2\mathfrak{f}_2\mathfrak{g}}$ do
14. Let $(\mathfrak{e}'_2, \mathfrak{f}'_2, \mathfrak{g}') = FC(t + 1, \mathfrak{e}_2\mathfrak{f}_2\mathfrak{g}, z'_1)$ and $\widehat{fw}_1 = fw_1 - 2^i(w'_1 + z'_1)$
15. Let $\widehat{\mathfrak{q}}_{t+1-1}[i] = \mathfrak{e}'_2$ and $\widehat{\mathfrak{q}}_{t+1-2}[i + 2 \bmod 32] = \mathfrak{f}'_2$.
16. Proceed with subroutine step2 (Algorithm 7-1-a, p. 137)
17. End if
18. Return $\mathcal{U}_{i+1}$

† For any $\mathfrak{q}_j[b]$ above: if $(j, b) \in A_i$ this bitcondition is retrieved from the current tuple in $\mathcal{U}_i$, otherwise it is retrieved from the forward or backward differential path depending on j.
‡ This line provides default (previous) values $\widehat{\mathfrak{q}}_j[b]$ for Algorithm 7-1-d line 10.

Algorithm 7-1-a Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$ for SHA-0 and SHA-1 (continued).

Subroutine step2

1. If $i < 2$ or $i \geq 34$ then
2. Let $\mathfrak{d}'_2 = \mathfrak{q}_{t+2-1}[i-2]$, $\widehat{q}_1 = q_1$, $\widehat{fw}_2 = fw_2$.
3. Proceed with subroutine step3 (Algorithm 7-1-b, p. 138)
4. Else
5. For each different $q'_1 \in \{-q_1[i-2], +q_1[i-2]\}$ do
6. Let $\widehat{q}_1 = q_1 - 2^{i-2}q'_1$.
7. If $i-2 \in \{1, 26, 31\}$ and $\widehat{q}_1 \neq \sum_{b=i-2+1}^{31} 2^b \Delta\widetilde{Q}_t[b]$ then skip steps 8-15.
8. Let $\mathfrak{d}_2 =$ '-', '.' or '+' based on whether $q'_1 = -1$, 0 or $+1$.
9. Let $\mathfrak{f}_0 = \mathfrak{q}_{t+2-3}[i \bmod 32]$.
10. For each different $w'_2 \in \{-DW_{t+2}[i-2], +DW_{t+2}[i-2]\}$ do
11. Let $Z_2 = fw_2 - 2^{i-2}w'_2$
12. For each different $z'_2 \in \{-Z_2[i-2], Z_2[i-2]\} \cap V_{t+2,\mathfrak{d}_2\mathfrak{e}'_2\mathfrak{f}_0}$ do
13. Let $(\mathfrak{d}'_2, \mathfrak{e}''_2, \mathfrak{f}'_0) = FC(t+2, \mathfrak{d}_2\mathfrak{e}'_2\mathfrak{f}_0, z'_2)$ and $\widehat{fw}_2 = fw_2 - 2^{i-2}(w'_2 + z'_2)$
14. Let $\widehat{\mathfrak{q}}_{t+2-1}[i-2] = \mathfrak{d}'_2$ and $\widehat{\mathfrak{q}}_{t+2-2}[i \bmod 32] = \mathfrak{e}''_2$.
15. Proceed with subroutine step3 (Algorithm 7-1-b, p. 138)
16. End if
17. Return to main routine

Algorithm 7-1-b Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$ for SHA-0 and SHA-1 (continued).

Subroutine step3

1. If $i < 4$ or $i \geq 36$ then
2. Let $\mathfrak{c}'_2 = \mathfrak{q}_{t+3-1}[i-4]$, $\widehat{q_2} = q_2$, $\widehat{fw_3} = fw_3$.
3. Proceed with subroutine step4 (Algorithm 7-1-c, p. 139)
4. Else
5. For each different $q'_2 \in \{-q_2[i-4], +q_2[i-4]\}$ do
6. Let $\widehat{q_2} = q_2 - 2^{i-4}q'_2$.
7. If $i-4 \in \{1, 26, 31\}$ and $\widehat{q_2} \neq \sum_{b=i-4+1}^{31} 2^b \Delta\widetilde{Q}_t[b]$ then skip steps 8-15.
8. Let $\mathfrak{c}_2 =$ '$\mathtt{-}$', '$\mathtt{.}$' or '$\mathtt{+}$' based on whether $q'_2 = -1$, 0 or $+1$.
9. Let $\mathfrak{e}_0 = \mathfrak{q}_{t+3-3}[i - 2 \bmod 32]$.
10. For each different $w'_3 \in \{-DW_{t+3}[i-4], +DW_{t+3}[i-4]\}$ do
11. Let $Z_3 = fw_3 - 2^{i-4}w'_3$
12. For each different $z'_3 \in \{-Z_3[i-4], Z_3[i-4]\} \cap V_{t+3,\mathfrak{c}_2\mathfrak{d}'_2\mathfrak{e}_0}$ do
13. Let $(\mathfrak{c}'_2, \mathfrak{d}''_2, \mathfrak{e}'_0) = FC(t+3, \mathfrak{c}_2\mathfrak{d}'_2\mathfrak{e}_0, z'_3)$ and $\widehat{fw_3} = fw_3 - 2^{i-4}(w'_3 + z'_3)$
14. Let $\widehat{\mathfrak{q}}_{t+3-1}[i-4] = \mathfrak{c}'_2$ and $\widehat{\mathfrak{q}}_{t+3-2}[i - 2 \bmod 32] = \mathfrak{d}''_2$.
15. Proceed with subroutine step4 (Algorithm 7-1-c, p. 139)
16. End if
17. Return to subroutine step2

Algorithm 7-1-c Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$ for SHA-0 and SHA-1 (continued).
Subroutine step4

1. If $i < 6$ or $i \geq 38$ then
2. Let $\mathfrak{b}'_2 = \mathfrak{q}_{t+4-1}[i-6]$, $\widehat{fw}_4 = fw_4$.
3. Proceed with subroutine step5 (Algorithm 7-1-d, p. 139)
4. Else
5. Let $\mathfrak{b}_2 = \mathfrak{q}_{t+4-1}[i-6]$ and $\mathfrak{d}_0 = \mathfrak{q}_{t+4-3}[i-4 \bmod 32]$.
6. For each different $w'_4 \in \{-DW_{t+4}[i-6], +DW_{t+4}[i-6]\}$ do
7. Let $Z_4 = fw_4 - 2^{i-6}w'_4$
8. For each different $z'_4 \in \{-Z_4[i-6], Z_4[i-6]\} \cap V_{t+4,\mathfrak{b}_2\mathfrak{c}'_2\mathfrak{d}_0}$ do
9. Let $(\mathfrak{b}'_2, \mathfrak{c}''_2, \mathfrak{d}'_0) = FC(t+4, \mathfrak{b}_2\mathfrak{c}'_2\mathfrak{d}_0, z'_4)$ and $\widehat{fw}_4 = fw_4 - 2^{i-6}(w'_4 + z'_4)$
10. Let $\widehat{\mathfrak{q}}_{t+4-1}[i-6] = \mathfrak{b}'_2$ and $\widehat{\mathfrak{q}}_{t+4-2}[i-4 \bmod 32] = \mathfrak{c}''_2$.
11. Proceed with subroutine step5 (Algorithm 7-1-d, p. 139)
12. End if
13. Return to subroutine step3

Algorithm 7-1-d Construction of $\mathcal{U}_{i+1}$ from $\mathcal{U}_i$ for SHA-0 and SHA-1 (continued).
Subroutine step5

1. If $i < 8$ then
2. Let $\widehat{fw}_5 = fw_5$.
3. Insert $(\widehat{q}_0, \widehat{q}_1, \widehat{q}_2, \widehat{fw}_1, \widehat{fw}_2, \widehat{fw}_3, \widehat{fw}_4, \widehat{fw}_5, (\widehat{\mathfrak{q}}_j[b])_{(j,b)\in A_{i+1}})$ in $\mathcal{U}_{i+1}$.
4. Else
5. Let $\mathfrak{a} = \mathfrak{q}_{t+5-1}[i-8]$ and $\mathfrak{c}_0 = \mathfrak{q}_{t+5-3}[i-6 \bmod 32]$.
6. For each different $w'_5 \in \{-DW_{t+5}[i-8], +DW_{t+5}[i-8]\}$ do
7. Let $Z_5 = fw_5 - 2^{i-8}w'_5$
8. For each different $z'_5 \in \{-Z_5[i-8], Z_5[i-8]\} \cap V_{t+5,\mathfrak{a}\mathfrak{b}'_2\mathfrak{c}_0}$ do
9. Let $(\mathfrak{a}', \mathfrak{b}''_2, \mathfrak{c}'_0) = FC(t+5, \mathfrak{a}\mathfrak{b}'_2\mathfrak{c}_0, z'_5)$ and $\widehat{fw}_5 = fw_5 - 2^{i-8}(w'_5 + z'_5)$
10. Insert $(\widehat{q}_0, \widehat{q}_1, \widehat{q}_2, \widehat{fw}_1, \widehat{fw}_2, \widehat{fw}_3, \widehat{fw}_4, \widehat{fw}_5, (\widehat{\mathfrak{q}}_j[b])_{(j,b)\in A_{i+1}})$ in $\mathcal{U}_{i+1}$.
11. End if
12. Return to subroutine step4

7.5 Differential cryptanalysis

As laid out in Section 7.3.4, all publications so far assume independence of local collisions in their analysis of disturbance vectors. However, this assumption is flawed as shown in this section and for instance the updated version [Man11] of [Man08]. So far no disturbance vector cost function that treats the local collisions as dependent has been presented.

The differential cryptanalysis method presented in this section does exactly this, i.e., it allows one to determine the exact success probability of a specific local collision or a specific combination of local collisions. As such it can be used as a cost function to determine (near-)optimal disturbance vectors. It improves upon the cost function that takes the product of the exact success probability of each individual local collision (allowing additional carries of δQ_i) over a specified range of steps, since it determines the exact *joint* success probability over the specified range of steps.

Our results clearly show that the joint probability differs from the product of the individual probabilities. Also, our results show that using dependence between local collisions leads to significantly higher success probabilities *under the correct optimal message conditions* than when assuming independent local collisions. It should be noted that if message conditions are used that are incompatible with the correct optimal message conditions then the average success probability will be lower and in the extreme can be even 0. In particular this may be the case for message conditions derived using previous analysis methods.

Our method also allows a more detailed analysis of the beginning of the second round and the last few steps. At these steps it may be more advantageous to divert from a prescribed combination of local collisions, as even higher success probabilities may be achieved. Moreover, our method allows us to find the smallest set of message expansion conditions which still results in the highest joint probability of success over the last three rounds. However, these conditions may be more limiting than or even incompatible with the message expansion conditions as prescribed by local collisions.

Our method is based on constructing differential paths that follow the prescribed local collision differences and summing the success probabilities of such differential paths that share the same message differences, the first five working state differences and the last five working state differences. The message differences and first five working state differences are preconditions of a differential path, whereas the last five working state differences determine $\delta IHV_{\text{diff}} = \delta IHV_{\text{out}} - \delta IHV_{\text{in}}$. Here, the success probability of a differential path over steps $i, \ldots, j$ is defined informally as the probability that all differential steps are fulfilled simultaneously assuming that $W_i, \ldots, W_j$ and $Q_{i-4}, \ldots, Q_i$ are independent uniform random variables and that both the message differences $\delta W_i, \ldots, \delta W_j$ and the first five working state differences $\delta RL(Q_{i-4}, 30)$, $\delta Q_{i-3}, \ldots, \delta Q_i$ as described in the differential path hold.

To overcome the exponential growth in the number of differential paths and possible message difference vectors $(\delta W_t)_{t=i}^{j}$ over the number of steps considered, our method employs several techniques. We first sketch the two most important techniques and then we present our method in detail.

The main technique is differential path reduction which removes all information in the differential path that is not strictly required for a forward or backward extension with a differential step. For instance, consider all differential paths over steps 59 to 66 with a single local collision starting at step 60 and bit position 2. There are 50 different possible values for ΔQ_{61}, since $\sigma(\Delta Q_{61})$ can be either positive or negative and carries can extend from bit position 2 up to bit position 26. The number of possible differential paths can be even greater as there can be multiple values for ΔF_{62}, ΔF_{63} and ΔF_{64} for each value of ΔQ_{61}. Nevertheless, since $\delta Q_{55} = \ldots = \delta Q_{59} = 0$ and $\delta Q_{62} = \ldots = \delta Q_{67} = 0$, the information of this local collision is not strictly required for the forward and backward extension with differential step 67 and 58, respectively. It follows that all such differential paths reduce to the trivial differential path with no differences at all.

Together with each reduced differential path, we maintain intermediary probabilities that accumulate the success probabilities of the removed parts of all differential paths. Since in the definition of the success probability of a differential path we assume that the message differences hold, we maintain a separate intermediary probability for each combination of a reduced differential path and possible message difference vector. This reduction is performed whenever all differential paths have been extended with a certain step t, after which we continue by extending with the next step.

Since the number of possible message difference vectors also grows exponentially in the number of steps, we employ another technique that allows us to 'combine' message difference vectors. Consider for each possible message difference vector $w = (\delta W_t)_{t=i}^{j}$ the function that maps each possible reduced differential path $\mathcal{P}$ to the associated intermediary probability for that w and $\mathcal{P}$. Suppose there are several message difference vectors $w_1, \ldots, w_K$ (over steps $i, \ldots, j$) for which said functions are identical. Then these message difference vectors all lead to identical reduced differential paths and associated intermediary probabilities. Furthermore, any future extension, i.e., future reduced differential paths and intermediary probabilities, depend only on these identical reduced differential paths and intermediary probabilities and not on the message difference vector. This implies that we can combine these message difference vectors in the following manner. We remove all intermediary probabilities associated with $w_2, \ldots, w_K$ and keep only the intermediary probability of a single message difference vector w_1 for future extensions. Furthermore, we create a substitution rule such that in any future extended message difference vector w that has the same differences as w_1 over steps $i, \ldots, j$, we may replace this subvector w_1 of w by any of $w_2, \ldots, w_K$.

7.5.1 Definition

Our method is based on differential paths $\mathcal{P}$ over steps $t = t_b, \ldots, t_e$, which are not described by bitconditions as in Section 7.4, but described by:

$$\mathcal{P} = (\delta RL(Q_{t_b-4}, 30), (\Delta Q_t)_{t=t_b-3}^{t_e}, \delta Q_{t_e+1}, (\Delta F_t)_{t=t_b}^{t_e}, (\delta W_t)_{t=t_b}^{t_e}),$$

under the following restrictions:

- correct differential steps for $t = t_b, \ldots, t_e$:

$$\delta Q_{t+1} = \sigma(RL(\Delta Q_t, 5)) + \delta RL(Q_{t-4}, 30)) + \sigma(\Delta F_t) + \delta W_t, \tag{7.15}$$

  where $\delta Q_{t+1} = \sigma(\Delta Q_{t+1})$ if $t \neq t_e$ and $\delta RL(Q_{t-4}, 30) = \sigma(RL(\Delta Q_{t-4}, 30))$ if $t \neq t_b$;

- for $t = t_b, \ldots, t_e$, both values -1 and $+1$ of $\Delta F_t[31]$ result in the same contribution $2^{31} \in \mathbb{Z}_{2^{32}}$ in $\sigma(\Delta F_t)$. We restrict $\Delta F_t[31]$ to $\{0, 1\}$ and a non-zero value represents $\Delta F_t[31] = \pm 1$;

- each $\Delta F_t[b]$ is individually possible, i.e., $(2^b \Delta F_t[b] \bmod 2^{32}) \in V_{t,b}$, where

$$V_{t,b} = \left\{ \begin{array}{c} (f_t(q'_1,q'_2,q'_3) \wedge 2^b) \\ -(f_t(q_1,q_2,q_3) \wedge 2^b) \end{array} \middle| \begin{array}{c} q_i, q'_i \in \mathbb{Z}_{2^{32}} \text{ for } i=1,2,3, \\ \Delta q_1 = \Delta Q_{t-1} \\ \Delta q_2 = RL(\Delta Q_{t-2}, 30) \\ \Delta q_3 = RL(\Delta Q_{t-3}, 30) \end{array} \right\}; \tag{7.16}$$

The probability $\Pr[\mathcal{P}]$ of such a differential path $\mathcal{P}$ is defined as:[32]

$$\Pr \left[\begin{array}{c} \Delta \widehat{Q}_j = \Delta Q_j \text{ for } j \in \{t_b-3, \ldots, t_e\}, \\ \delta \widehat{Q}_{t_e+1} = \delta Q_{t_e+1}, \\ 2^b \Delta \widehat{F}_i[b] = 2^b \Delta F_i[b] \bmod 2^{32}, \\ \text{for } i \in \{t_b, \ldots, t_e\},\ b \in \{0, \ldots, 31\} \end{array} \middle| \begin{array}{c} \widehat{Q}_{t_b-4} \xleftarrow{R} \mathbb{Z}_{2^{32}}, \\ \widehat{Q}'_{t_b-4} = RR(RL(\widehat{Q}_{t_b-4}, 30) + \delta RL(Q_{t_b-4}, 30), 30), \\ \widehat{Q}_k \xleftarrow{R} \mathbb{Z}_{2^{32}},\ \widehat{Q}'_k = \widehat{Q}_k + \delta Q_k \text{ for } k \in \{t_b-3, \ldots, t_b\}; \\ \widehat{W}_t \xleftarrow{R} \mathbb{Z}_{2^{32}},\ \widehat{W}'_t = \widehat{W}_t + \delta W_t, \\ \widehat{F}_t = f_t(\widehat{Q}_{t-1}, RL(\widehat{Q}_{t-2}, 30), RL(\widehat{Q}_{t-3}, 30)), \\ \widehat{F}'_t = f_t(\widehat{Q}'_{t-1}, RL(\widehat{Q}'_{t-2}, 30), RL(\widehat{Q}'_{t-3}, 30)), \\ \widehat{Q}_{t+1} = RL(\widehat{Q}_t, 5) + RL(\widehat{Q}_{t-4}, 30) + \widehat{F}_t + \widehat{W}_t + AC_t, \\ \widehat{Q}'_{t+1} = RL(\widehat{Q}'_t, 5) + RL(\widehat{Q}'_{t-4}, 30) + \widehat{F}'_t + \widehat{W}'_t + AC_t, \\ \text{for } t \in \{t_b, \ldots, t_e\} \end{array} \right].$$

More informally, this is the success probability of the following experiment:

Experiment 7.1. *This experiment involves partial SHA-1 computations of two messages. For the first message, values for $\widehat{Q}_{t_b-4}, \ldots, \widehat{Q}_{t_b}$ and $\widehat{W}_{t_b}, \ldots, \widehat{W}_{t_e}$ are selected uniformly at random. The remaining values for $\widehat{Q}_{t_b+1}, \ldots, \widehat{Q}_{t_e+1}$ are computed using SHA-0's and SHA-1's step function for $t = t_b, \ldots, t_e$:*

$$\widehat{F}_t = f_t(\widehat{Q}_{t-1}, RL(\widehat{Q}_{t-2}, 30), RL(\widehat{Q}_{t-3}, 30)),$$
$$\widehat{Q}_{t+1} = RL(\widehat{Q}_t, 5) + RL(\widehat{Q}_{t-4}, 30) + \widehat{F}_t + \widehat{W}_t + AC_t.$$

For the second message, we apply the differential path differences to the randomly selected variables:

$$\begin{aligned} \widehat{Q}'_{t_b-4} &= RR(RL(\widehat{Q}_{t_b-4}, 30) + \delta RL(Q_{t_b-4}, 30), 30), \\ \widehat{Q}'_i &= \widehat{Q}_i + \delta Q_i && \text{for } i = t_b - 3, \ldots, t_b, \\ \widehat{W}'_j &= \widehat{W}_j + \delta W_j && \text{for } j = t_b, \ldots, t_e. \end{aligned}$$

32. Note that $2^b \Delta \widehat{F}_j[b] = 2^b \Delta F_j[b] \bmod 2^{32}$ for $b = 0, \ldots, 31$ does not imply $\Delta \widehat{F}_j = \Delta F_j$. At bit position 31, the first case distinguishes only between $\Delta F_j[31]$ being zero or non-zero, whereas the latter case also distinguishes $\Delta F_j[31]$ by sign.

The remaining values $\widehat{Q}'_{t_b+1}, \ldots, \widehat{Q}'_{t_e+1}$ *are computed using SHA-0's and SHA-1's step function for* $t = t_b, \ldots, t_e$*:*

$$\begin{aligned}\widehat{F}'_t &= f_t(\widehat{Q}'_{t-1}, RL(\widehat{Q}'_{t-2}, 30), RL(\widehat{Q}'_{t-3}, 30)),\\ \widehat{Q}'_{t+1} &= RL(\widehat{Q}'_t, 5) + RL(\widehat{Q}'_{t-4}, 30) + \widehat{F}'_t + \widehat{W}'_t + AC_t.\end{aligned}$$

The experiment has succeeded when the above step function computations follow the differential path $\mathcal{P}$*, thus when all the following equations hold:*

$$\begin{aligned}\delta\widehat{Q}_{t_e+1} &= \delta Q_{t_e+1},\\ \Delta\widehat{Q}_i &= \Delta Q_i && \text{for } i = t_b - 3, \ldots, t_e,\\ 2^b \Delta\widehat{F}_j[b] &= 2^b \Delta F_j[b] \bmod 2^{32} && \text{for } j = t_b, \ldots, t_e,\ b = 0, \ldots, 31.\end{aligned}$$

7.5.2 Probability analysis

In this section we present a method to efficiently determine the probability of a differential path $\mathcal{P}$. To this end, consider a slight change in Experiment 7.1:

Experiment 7.2. *This experiment is a modification of Experiment 7.1. Instead of randomly selecting values for* $\widehat{W}_{t_b}, \ldots, \widehat{W}_{t_e}$ *and computing values for* $\widehat{Q}_{t_b+1}, \ldots, \widehat{Q}_{t_e+1}$*, one randomly selects values for* $\widehat{Q}_{t_b+1}, \ldots, \widehat{Q}_{t_e+1}$ *and computes values for* $\widehat{W}_{t_b}, \ldots, \widehat{W}_{t_e}$ *using:*

$$\begin{aligned}\widehat{F}_t &= f_t(\widehat{Q}_{t-1}, RL(\widehat{Q}_{t-2}, 30), RL(\widehat{Q}_{t-3}, 30)),\\ \widehat{W}_t &= \widehat{Q}_{t+1} - RL(\widehat{Q}_t, 5) - RL(\widehat{Q}_{t-4}, 30) - \widehat{F}_t - AC_t.\end{aligned}$$

The success requirement is left unchanged.

Since there is a bijective relation between $(\widehat{W}_t)_{t=t_b}^{t_e}$ and $(\widehat{Q}_{t+1})_{t=t_b}^{t_e}$, this implies that $(\widehat{W}_t)_{t=t_b}^{t_e}$ is also uniformly distributed in Experiment 7.2. Hence, the success probabilities of both experiments are equal. Note that this second experiment is completely determined by the values of $(\widehat{Q}_t)_{t=t_b-4}^{t_e+1}$. Next, consider another experiment:

Experiment 7.3. *This experiment is a modification of Experiment 7.2. As above, we set*

$$\begin{aligned}\widehat{Q}'_{t_b-4} &= RR(RL(\widehat{Q}_{t_b-4}, 30) + \delta RL(Q_{t_b-4}, 30), 30),\\ \widehat{Q}'_i &= \widehat{Q}_i + \delta Q_i && \text{for } i = t_b - 3, \ldots, t_b.\end{aligned}$$

However, instead of setting $\widehat{W}'_t = \widehat{W}_t + \delta W_t$ *for* $t = t_b, \ldots, t_e$ *and computing values for* $\widehat{Q}'_{t_b+1}, \ldots, \widehat{Q}'_{t_e+1}$*, one sets* $\widehat{Q}'_{t+1} = \widehat{Q}_{t+1} + \delta Q_{t+1}$ *for* $t = t_b, \ldots, t_e$ *and computes values for* $\widehat{W}'_{t_b}, \ldots, \widehat{W}'_{t_e}$*:*

$$\begin{aligned}\widehat{F}'_t &= f_t(\widehat{Q}'_{t-1}, RL(\widehat{Q}'_{t-2}, 30), RL(\widehat{Q}'_{t-3}, 30)),\\ \widehat{W}'_t &= \widehat{Q}'_{t+1} - RL(\widehat{Q}'_t, 5) - RL(\widehat{Q}'_{t-4}, 30) - \widehat{F}'_t - AC_t.\end{aligned}$$

The success requirement is left unchanged. In particular, one does not need an additional check that $\delta\widehat{W}_t = \delta W_t$ as in case of success this is implied by Equation 7.15:

$$\begin{aligned}\delta\widehat{W}_t &= \delta\widehat{Q}_{t+1} - \sigma(RL(\Delta\widehat{Q}_t, 5)) - \delta RL(\widehat{Q}_{t-4}, 30)) - \sigma(\Delta\widehat{F}_t)\\ &= \delta Q_{t+1} - \sigma(RL(\Delta Q_t, 5)) - \delta RL(Q_{t-4}, 30)) - \sigma(\Delta F_t)\\ &= \delta W_t.\end{aligned}$$

Lemma 7.1. *For fixed values $(\widehat{Q}_t)_{t=t_b-4}^{t_e+1}$, Experiment 7.3 succeeds if and only if Experiment 7.2 succeeds.*

Proof. If Experiment 7.3 succeeds then Eq. 7.15 must hold resulting in $(\delta\widehat{W}_t)_{t=t_b}^{t_e} = (\delta W_t)_{t=t_b}^{t_e}$. This implies that Experiment 7.2 also succeeds, since it will obtain identical values for both $(\widehat{W}'_t)_{t=t_b}^{t_e}$ and $(\widehat{Q}'_{t+1})_{t=t_b}^{t_e}$ as Experiment 7.3.

Suppose Experiment 7.3 fails. If $(\delta\widehat{W}_t)_{t=t_b}^{t_e} = (\delta W_t)_{t=t_b}^{t_e}$ then again Experiment 7.2 will obtain identical values for both $(\widehat{W}'_t)_{t=t_b}^{t_e}$ and $(\widehat{Q}'_{t+1})_{t=t_b}^{t_e}$ as Experiment 7.3 and thus Experiment 7.2 also fails. Otherwise, let $\widetilde{t}$ be the smallest $t \in \{t_b, \ldots, t_e\}$ for which $\delta\widehat{W}_t \neq \delta W_t$. This implies that Experiment 7.2 obtains identical values for $(\widehat{W}'_t)_{t=t_b}^{\widetilde{t}-1}$, $(\widehat{Q}'_t)_{t=t_b+1}^{\widetilde{t}}$ and $\Delta\widehat{F}_{\widetilde{t}}$ as Experiment 7.3. Assume that $\Delta\widehat{Q}_t = \Delta Q_t$ holds for all $t = t_b-3, \ldots, \widetilde{t}$ and $2^b\Delta\widehat{F}_{\widetilde{t}}[b] = 2^b\Delta F_{\widetilde{t}}[b] \bmod 2^{32}$ holds for all $b \in \{0, \ldots, 31\}$. Then Equation 7.15 implies that $\delta\widehat{W}_{\widetilde{t}} = \delta W_{\widetilde{t}}$ which contradicts the choice of $\widetilde{t}$, therefore this assumption does not hold. This failed assumption together with the fact that Experiment 7.2 obtains identical values for $(\widehat{Q}'_t)_{t=t_b+1}^{\widetilde{t}}$ and $\Delta\widehat{F}_{\widetilde{t}}$ as Experiment 7.3 directly implies that Experiment 7.2 must also fail. □

We use these experiments to show that the probability $\Pr[\mathcal{P}]$ of such a differential path can be determined as the fraction $N_{\mathcal{P}}/2^{32(t_e-t_b+6)}$ where $N_{\mathcal{P}}$ is the number of possible values $(\widehat{Q}_t)_{t=t_b-4}^{t_e+1} \in \mathbb{Z}_{2^{32}}^{t_e-t_b+6}$ for which this third experiment succeeds. In other words, $N_{\mathcal{P}}$ is the number of possible values $(\widehat{Q}_t)_{t=t_b-4}^{t_e+1} \in \mathbb{Z}_{2^{32}}^{t_e-t_b+6}$ for which

- for $t = t_b - 3, \ldots, t_e$: $\Delta Q_t = \Delta\widehat{Q}_t$;
- for $t = t_b, \ldots, t_e$ and $b = 0, \ldots, 31$:

$$\begin{aligned}(2^b\Delta F_t[b] \bmod 2^{32}) \;=\; &(f_t(\widehat{Q}'_{t-1}, RL(\widehat{Q}'_{t-2}, 30), RL(\widehat{Q}'_{t-3}, 30)) \wedge 2^b)\\ &- (f_t(\widehat{Q}_{t-1}, RL(\widehat{Q}_{t-2}, 30), RL(\widehat{Q}_{t-3}, 30)) \wedge 2^b),\end{aligned}$$

where $\widehat{Q}'_t = \widehat{Q}_t + \delta Q_t$ for $t \in \{t_b - 3, \ldots, t_e\}$.

An efficient way to determine the probability $\Pr[\mathcal{P}]$ is based on the fact that we can partition the bits $\widehat{Q}_t[b]$ into parts $G_{\Delta Q}$, $G_0, \ldots, G_K$ for some $K \in \mathbb{N}$ that each contribute a factor to $\Pr[\mathcal{P}]$. One important part $G_{\Delta Q}$ consists of all indices (j, i) such that $\Delta Q_j[i] \neq 0$ where $j \in \{t_b - 3, \ldots, t_e\}$ and $i \in \{0, \ldots, 31\}$. Since the values $\widehat{Q}'_j[i]$ and $\widehat{Q}_j[i]$ are uniquely determined for all $(j, i) \in G_{\Delta Q}$, this partition contributes the factor of $p_{\Delta Q} = 1/2^{|G_{\Delta Q}|}$ to $\Pr[\mathcal{P}]$.

Consider the set $\mathcal{S}_F$ of all indices (t, b) where $t \in \{t_b, \ldots, t_e\}$ and $b \in \{0, \ldots, 31\}$ such that $|V_{t,b}| > 1$ and thus $\Delta F_t[b]$ is not trivially fulfilled. Let $\mathcal{S}_Q$ be the set of all indices (j, i) where $j \in \{t_b - 4, \ldots, t_e + 1\}$ and $i \in \{0, \ldots, 31\}$ such that $\Delta Q_j[i] = 0$ and $Q_j[i]$ is involved with some $\Delta F_t[b]$ with $(t, b) \in \mathcal{S}_F$:

$$\{(j+1, i), (j+2, i+2 \bmod 32), (j+3, i+2 \bmod 32)\} \cap \mathcal{S}_F \neq \emptyset.$$

All indices (j, i) of bits $Q_j[i]$ where $(j, i) \notin \mathcal{S}_Q \cup G_{\Delta Q}$ for $j \in \{t_b - 4, \ldots, t_e + 1\}$, $i \in \{0, \ldots, 31\}$ form part G_0. Part G_0 consists by construction of all indices of free bits $Q_j[i]$ whose values do not affect ΔQ_j or any of the non-trivially fulfilled ΔF_t and thus contributes a factor of $p_0 = 2^{|G_0|}/2^{|G_0|} = 1$ to $\Pr[\mathcal{P}]$.

The set of remaining indices $\mathcal{S}_Q$ is further partitioned by constructing a graph $\mathcal{G}$ consisting of vertices $F_t[b]$ for all $(t, b) \in \mathcal{S}_F$ and vertices $Q_j[i]$ for all $(j, i) \in \mathcal{S}_Q$. There is an edge between two nodes $F_t[b]$ and $Q_j[i]$ if and only if:

$$(t, b) \in \{(j+1, i), (j+2, i+2 \bmod 32), (j+3, i+2 \bmod 32)\}, \tag{7.17}$$

i.e., $Q_j[i]$ is involved with $F_t[b]$. The graph $\mathcal{G}$ can be uniquely partitioned into connected subgraphs $\mathcal{G}_1, \ldots, \mathcal{G}_K$. This partition $\mathcal{G}_1, \ldots, \mathcal{G}_K$ of $\mathcal{G}$ defines a partition $G_1, \ldots, G_K$ of $\mathcal{S}_Q$ as follows:

$$G_k = \{(j, i) \mid Q_j[i] \in \mathcal{G}_k\}, \quad k \in \{1, \ldots, K\}.$$

By construction, all bits $Q_j[i]$ with associated nodes in the partition G_k influence a non-trivially fulfilled $\Delta F_t[b]$ if and only if there is an associated node $F_t[b]$ in $\mathcal{G}_k$. The probability p_k can be determined as $N_{\mathcal{P},k} \cdot 2^{-|G_k|}$, where $N_{\mathcal{P},k}$ is the number of different values of $(Q_j[i])_{(j,i)\in G_k}$ that result in the correct value of all $\Delta F_t[b]$, where $F_t[b]$ is a node in $\mathcal{G}_k$, and assuming $Q'_j[i] = Q_j[i] + \Delta Q_j[i]$ for all $(j, i) \in G_{\Delta Q}$.

Lemma 7.2. *The probability* $\Pr[\mathcal{P}]$ *is the product of* $p_{\Delta Q}$, p_0, $p_1, \ldots,$ p_K*:*

$$\Pr[\mathcal{P}] = p_{\Delta Q} \cdot p_0 \cdot \prod_{k=1}^{K} p_k = 2^{-|G_{\Delta Q}|} \prod_{k=1}^{K} \frac{N_{\mathcal{P},k}}{2^{|G_k|}}.$$

Proof. This lemma follows directly from the above construction. □

As a simple example, a single local collision starting with $\delta W_t = 2^b$ (without carry in δQ_{t+1}) results in five parts: $G_{\Delta Q}, G_0, G_1, G_2, G_3$. Part $G_{\Delta Q}$ consists solely of the disturbance $(t+1, b)$. Parts G_1, G_2 and G_3 consist each of two bit indices namely the other two $Q_i[j]$ involved with $Q_{t+1}[b]$ in the boolean function in step $t+2$, $t+3$ and $t+4$, respectively.

7.5.3 Extending

Extending a differential path $\mathcal{P}$ forward or backward with step l is done as follows. First a δW_l consistent with DW_l is chosen:

$$\delta W_l \in \Big\{\sigma(\Delta W_l) \;\Big|\; \Delta W_l[i] \in \{-DW_l[i], +DW_l[i]\} \text{ for } i = 0, \ldots, 31\Big\}.$$

In the case of a forward extension choose any BSDR ΔQ_l of δQ_l and a valid ΔF_l: $\Delta F_l[i] \in V_{l,i}$ for $i = 0, \ldots, 31$ (see Equation 7.16). Now δQ_{l+1} is determined as

$$\delta Q_{l+1} = \sigma(RL(\Delta Q_l, 5)) + \sigma(RL(\Delta Q_{l-4}, 30)) + \sigma(\Delta F_l) + \delta W_l$$

and $\mathcal{P}$ is extended by appending ΔQ_l, δQ_{l+1}, ΔF_l and δW_l.

Otherwise for a backward extension choose any BSDR ΔQ_{l-3} of δQ_{l-3} and a valid ΔF_l: $\Delta F_l[i] \in V_{l,i}$ for $i = 0, \ldots, 31$. Then $\delta RL(Q_{l-4}, 30)$ is determined as

$$\delta RL(Q_{l-4}, 30) = \sigma(\Delta Q_{l+1}) - \sigma(RL(\Delta Q_l, 5)) - \sigma(\Delta F_l) - \delta W_l$$

and $\mathcal{P}$ is extended by prepending $\delta RL(Q_{l-4}, 30)$, ΔQ_{l-3} and ΔF_l and δW_l.

7.5.4 Reduction

As said before, we are interested in the sum of success probabilities of differential paths that share the same message differences, the first five working state differences and the last five working state differences. The differential paths themselves are of lesser interest.

To reduce the total number of differential paths after extending a given step, we try to sensibly remove differences $\Delta Q_t[b]$ and $\Delta F_i[j]$ whose fulfillment probabilities are independent of any possible forward or backward extension choices. By keeping a graph-like structure of all intermediary differential paths one can always reconstruct non-reduced differential paths over all prescribed steps by performing the same extension choices without the subsequent reductions. Furthermore, we show in the next section how to use intermediary probabilities that accumulate the probabilities of such removed differences.

Given a differential path $\mathcal{P}$ over steps $t = t_b, \ldots, t_e$, we determine which differences $\Delta Q_j[i]$ and $\Delta F_t[b]$ can safely be removed. This is done by constructing the following graph $\widetilde{\mathcal{G}}$:

1. For $t = t_b, \ldots, t_e$ and $b = 0, \ldots, 31$, we add a node $F_t[b]$ if and only if $\Delta Q_{t-1}[b] \neq 0$ or $\Delta Q_{t-2}[b + 2 \bmod 32] \neq 0$ or $\Delta Q_{t-3}[b + 2 \bmod 32] \neq 0$.

2. For $j = t_b - 4, \ldots, t_e + 1$ and $i = 0, \ldots, 31$, we add a node $Q_j[i]$ if and only if at least one of the following differences is present in $\mathcal{P}$ and non-zero: $\Delta Q_j[i]$, $\Delta Q_{j-1}[i + 2 \bmod 32]$, $\Delta Q_{j-2}[i + 2 \bmod 32]$, $\Delta Q_{j+1}[i - 2 \bmod 32]$, $\Delta Q_{j-1}[i]$, $\Delta Q_{j+2}[i - 2 \bmod 32]$, $\Delta Q_{j+1}[i]$.

3. We connect each node $F_t[b]$ in the graph with edges to the nodes $Q_{j-1}[i]$, $Q_{j-2}[i + 2 \bmod 32]$ and $Q_{j-3}[i + 2 \bmod 32]$.

Consider all connected subgraphs $\widetilde{\mathcal{G}}_1, \ldots, \widetilde{\mathcal{G}}_K$ of $\widetilde{\mathcal{G}}$. Let $k \in \{1, \ldots, K\}$, if for all nodes $Q_j[i]$ of the connected subgraph $\widetilde{\mathcal{G}}_k$ we have $\Delta Q_j[i] = 0$ or $j \in \{t_b + 1, \ldots, t_e - 4\}$ then all differences $\Delta Q_j[i]$ and $\Delta F_t[b]$ associated with the respective nodes $Q_j[i]$ and $F_t[b]$ in $\widetilde{\mathcal{G}}_k$ can be safely removed. Since Equation 7.15 must hold, the necessary corrections are made in δW_t. For all nodes $Q_i[j] \in \widetilde{\mathcal{G}}_k$ where $\Delta Q_i[j] \neq 0$ we do the following:

1. Replace the value of δW_{j+4} by $\delta W_{j+4} + \Delta Q_j[i] \cdot 2^{i-2 \bmod 32}$;
2. Replace the value of δW_j by $\delta W_j + \Delta Q_j[i] \cdot 2^{i+5 \bmod 32}$;
3. Replace the value of δW_{j-1} by $\delta W_{j-1} - \Delta Q_j[i] \cdot 2^i$;
4. Replace the value of $\Delta Q_j[i]$ by 0.

For all nodes $F_t[b] \in \widetilde{\mathcal{G}}_k$ we do the following:

1. Replace the value of δW_t by $\delta W_t + \Delta F_t[b] \cdot 2^b$;
2. Replace the value of $\delta F_t[b]$ by 0.

Note that $\Delta Q_{t_b-4}, \ldots, Q_{t_b}$ and $\Delta Q_{t_e-3}, \ldots, \Delta Q_{t_e+1}$ remain untouched. Also, it can be seen that the graph $\mathcal{G}$ from Section 7.5.2 is a subgraph of $\widetilde{\mathcal{G}}$.

Lemma 7.3. *Given a differential path $\mathcal{P}$ over steps $t_b, \ldots, t_e$ and its reduced version $\widehat{\mathcal{P}}$, let $\widetilde{\mathcal{P}}$ be defined over steps $t_b, \ldots, t_e$ by:*

$$\begin{aligned} \Delta\widetilde{Q}_j[i] &= \Delta Q_j[i] - \Delta\widehat{Q}_j[i] && \text{for } j = t_b - 4, \ldots, t_e + 1,\ i = 0, \ldots, 31; \\ \Delta\widetilde{F}_t[b] &= \Delta F_t[b] - \Delta\widehat{F}_t[b] && \text{for } t = t_b, \ldots, t_e,\ b = 0, \ldots, 31; \\ \delta\widetilde{W}_t &= \delta W_t - \delta\widehat{W}_t && \text{for } t = t_b, \ldots, t_e. \end{aligned}$$

(Thus $\widetilde{\mathcal{P}}$ is defined by the eliminated differences $\Delta Q_j[i]$ and $\Delta F_t[b]$ and the negative sum of corrections to δW_t.) Then $\widehat{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ are valid differential paths and

$$\Pr[\mathcal{P}] = \Pr[\widehat{\mathcal{P}}] \cdot \Pr[\widetilde{\mathcal{P}}].$$

Proof. Let $k \in \{1, \ldots, K\}$. Then for every $Q_j[i]$ in $\widetilde{\mathcal{G}}_k$ with $\Delta Q_j[i] \neq 0$, also all related $F_t[b]$ are in $\widetilde{\mathcal{G}}_k$ by construction. (As before a $Q_j[i]$ and $F_t[b]$ are related if Equation 7.17 holds.) Similarly, for every $F_t[b]$ in $\widetilde{\mathcal{G}}_k$ also all related $Q_j[i]$ are in $\widetilde{\mathcal{G}}_k$ by construction.

Let $\mathcal{K} \subset \{1, \ldots, K\}$ be the index set of all connected subgraphs $\widetilde{\mathcal{G}}_k$ such that $\Delta Q_j[i] = 0$ or $j \in \{t_b + 1, \ldots, t_e - 4\}$ for all nodes $Q_j[i]$ in $\widetilde{\mathcal{G}}_k$. Then the differences $\Delta Q_j[i]$ and $\Delta F_t[b]$ associated with the respective nodes $Q_j[i]$ and $F_t[b]$ in $\widetilde{\mathcal{G}}_k$ are either present in $\widetilde{\mathcal{P}}$ if $k \in \mathcal{K}$ or in $\widehat{\mathcal{P}}$ if $k \notin \mathcal{K}$. Furthermore, all other differences $\Delta Q_j[i]$ and $\Delta F_t[b]$ present in $\widehat{\mathcal{P}}$ and $\widetilde{\mathcal{P}}$ are zero.

The probability $\Pr[\mathcal{P}]$ is the product of $p_{\Delta Q}, p_1, \ldots, p_L$ as in Section 7.5.2 ($p_0 = 1$ is always trivial). First, $p_{\Delta Q}$ is determined by all differences $\Delta Q_j[i] \neq 0$ in $\mathcal{P}$ where each such difference $\Delta Q_j[i]$ contributes a factor of $1/2$ to $p_{\Delta Q}$. Since each such difference $\Delta Q_j[i]$ is either present in $\widehat{\mathcal{P}}$ or in $\widetilde{\mathcal{P}}$ it follows that $p_{\Delta Q} = \widehat{p}_{\Delta Q} \cdot \widetilde{p}_{\Delta Q}$. Since the graph $\mathcal{G}$ from Section 7.5.2 is a subgraph of $\widetilde{\mathcal{G}}$, it follows that each connected subgraph $\mathcal{G}_l$ for $l \in \{1, \ldots, L\}$ is a subgraph of some $\widetilde{\mathcal{G}}_k$. If $k \in \mathcal{K}$ then the probability p_l associated with $\mathcal{G}_l$ is a factor of $\Pr[\widetilde{\mathcal{P}}]$ and not of $\Pr[\widehat{\mathcal{P}}]$. Otherwise, $k \notin \mathcal{K}$ and p_l

is a factor of $\Pr[\widehat{\mathcal{P}}]$ and not of $\Pr[\widetilde{\mathcal{P}}]$. Since $p_{\Delta Q}$ is divided into two factors, one for $\widehat{\mathcal{P}}$ and one for $\widetilde{\mathcal{P}}$ and the probabilities $p_1, \ldots, p_L$ have been partitioned between $\Pr[\widehat{\mathcal{P}}]$ and $\Pr[\widetilde{\mathcal{P}}]$, one can conclude that $\Pr[\widehat{\mathcal{P}}] \cdot \Pr[\widetilde{\mathcal{P}}]$. □

For a given differential path $\mathcal{P}$ we denote by $\text{Reduce}(\mathcal{P})$ the differential path resulting from reducing $\mathcal{P}$ by the above method.

Observation 7.1. *Let $\mathcal{P}$ be a differential path over steps $t_b, \ldots, t_e$ for which $\sigma(\Delta Q_i) = 0$ for $i = t_b - 4, \ldots, t_b$ and for $i = t_e - 3, \ldots, t_e + 1$. Then $\widehat{\mathcal{P}} = \text{Reduce}(\mathcal{P})$ is trivial: $\widehat{\mathcal{P}} = (0, ((0)_{j=0}^{31})_{i=t_b-4}^{t_e+1}, 0, ((0)_{j=0}^{31})_{i=t_b}^{t_e}, (0)_{i=t_b}^{t_e})$.*

7.5.5 Single local collision analysis

In this section we analyze the probabilities of local collisions either with or without additional carries. In Section 7.5.7 we extend this analysis to combinations of local collisions as prescribed by a disturbance vector. For now, let $(DV_t)_{t=0}^{79} \in \mathbb{Z}_{2^{32}}^{80}$ be a disturbance vector consisting of a single disturbance starting between step 0 and 74, which implies that after step 79 all corrections for this single disturbance have been made. Although $(DV_t)_{t=0}^{79}$ is not a valid disturbance vector for either SHA-0 or SHA-1, it allows for differential cryptanalysis of a single local collision that is easily extended to valid disturbance vectors.

First we present a number of definitions that we need later on. For $t = 1, \ldots, 80$, we denote by $\mathcal{Q}_t$ the set of allowed values for ΔQ_t. We offer two choices for $\mathcal{Q}_t$. The first choice is to select one from the family of sets $\mathcal{Q}_{\text{c},u,t}$ for $u \in \{0, \ldots, 32\}$ of allowed values for ΔQ_t as prescribed by the disturbance(s) in $(DV_t)_{t=0}^{79}$ allowing carries and where the weight is bounded by u plus the NAF weight:

$$\mathcal{Q}_{\text{c},u,t} = \left\{ \text{BSDR } Y \;\middle|\; \begin{array}{l} \sigma(Y) = \sigma(Z), \\ Z[i] \in \{-DV_{t-1}[i], DV_{t-1}[i]\},\ i = 0, \ldots, 31, \\ w(Y) \leq w(\text{NAF}(\sigma(Y))) + u \end{array} \right\}.$$

For $u = 32$, the bound on the weight is trivially fulfilled and we denote $\mathcal{Q}_{\text{c},t}$ for $\mathcal{Q}_{\text{c},32,t}$. The set $\mathcal{Q}_{\text{c},t}$ is the preferred choice for $\mathcal{Q}_t$.

The second choice for $\mathcal{Q}_t$ is the set $\mathcal{Q}_{\text{nc},t}$ which is defined as the set of allowed values for ΔQ_t as prescribed by the disturbance(s) in $(DV_t)_{t=0}^{79}$ not allowing carries:

$$\mathcal{Q}_{\text{nc},t} = \left\{ Y \in \mathcal{Q}_{\text{c},0,t} \;\middle|\; Y[i] \in \{0, -DV_{t-1}[i], DV_{t-1}[i]\} \right\}.$$

No carries can be present as $Y \in \mathcal{Q}_{\text{c},0,t}$. Nevertheless, $\mathcal{Q}_{\text{c},0,t}$ still allows the disturbances $\Delta Q_t = \{2, \overline{0}\}$ and $\Delta Q_t = \{1, 0\}$ (using compact notation, see p. 18) in case of two serial local collisions starting at step $t-1$: $DV_{t-1} = 2^1 + 2^0$. The condition $Y[i] \in \{0, -DV_{t-1}[i], DV_{t-1}[i]\}$ prevents disturbances associated with '0'-bits in the disturbance vector.

We assume a choice has been made for $\mathcal{Q}_t$ and preferably this is $\mathcal{Q}_{\text{c},t}$, however we may need another choice for $\mathcal{Q}_t$ in the upcoming sections. To simplify notation, we

use $\sigma(\mathcal{Q}_t)$ and $\sigma(RL(\mathcal{Q}_t, n))$ to denote $\{\sigma(Y) \mid Y \in \mathcal{Q}_t\}$ and $\{\sigma(RL(Y, n)) \mid Y \in \mathcal{Q}_t\}$, respectively. The remaining definitions in this section may depend on the particular choice of $\mathcal{Q}_t$.

Let $(DW_t)_{t=0}^{79} \in \mathbb{Z}_{2^{32}}^{80}$ be the associated message expansion XOR difference vector of the disturbance vector (see Equation 7.9, p. 121). Then for $t = 0, \ldots, 79$, we define the set $\mathcal{W}_t$ as the set of possible δW_t given XOR difference DW_t:

$$\mathcal{W}_t = \Big\{\sigma(\Delta W_t) \;\Big|\; \Delta W_t[i] \in \{-DW_t[i], DW_t[i]\},\ i = 0, \ldots, 31\Big\}.$$

Let $\mathcal{D}_{[i,j]}$ be the set of all differential paths $\mathcal{P}$ over steps $i, \ldots, j$ with:

- $\delta W_t \in \mathcal{W}_t$ for $t = i, \ldots, j$;
- $\Delta Q_t \in \mathcal{Q}_t$ for $t = i-3, \ldots, j$;
- $\delta RL(Q_{i-4}, 30) \in \sigma(RL(\mathcal{Q}_{i-4}, 30))$;
- $\delta Q_{j+1} \in \sigma(\mathcal{Q}_{j+1})$;
- $\Pr[\mathcal{P}] > 0$.

Informally, $\mathcal{D}_{[i,j]}$ is the set of all possible differential paths over steps $i, \ldots, j$ that follow the local collision differences as prescribed by the disturbance vector where carries are limited through the choice for $\mathcal{Q}_t$. These are the differential paths that we are interested in. Let $\mathcal{R}_{[i,j]}$ denote the set $\{\text{Reduce}(\mathcal{P}) \mid \mathcal{P} \in \mathcal{D}_{[i,j]}\}$ of all reduced versions of the differential paths in $\mathcal{D}_{[i,j]}$. We like to point out that $|\mathcal{R}_{[i,j]}| \leq |\mathcal{D}_{[i,j]}|$ and that $|\mathcal{R}_{[i,j]}|$ can be significantly smaller than $|\mathcal{D}_{[i,j]}|$, especially for a large number of steps $j - i + 1$.

Now we can analyze the success probability of the local collision. The success probability $p_{w,[t_b,t_e]}$ of the single local collision in $(DV_t)_{t=0}^{79}$ which begins at step $t_b \geq 0$ and ends with step $t_e < 80$ using given message difference vector $w = (\delta W_t)_{t=t_b}^{t_e} \in (\mathcal{W}_t)_{t=t_b}^{t_e}$ is determined as:

$$p_{w,[t_b,t_e]} = \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ (\delta \widehat{W}_t)_{t=t_b}^{t_e} = w}} \Pr[\widehat{\mathcal{P}}].$$

Thus to analyze the local collision we generate differential paths $\mathcal{P} \in \mathcal{D}_{[i,j]}$. Instead of storing all possible differential paths, we keep only reduced differential paths along with intermediary probabilities. For a reduced differential path $\mathcal{P}_r$ over steps $i, \ldots, j$ and message difference vector $w \in (\mathcal{W}_t)_{t=i}^{j}$, we define the intermediary probability $p_{(w,\mathcal{P}_r,[i,j])}$ as:

$$p_{(w,\mathcal{P}_r,[i,j])} = \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[i,j]} \\ \text{Reduce}(\widehat{\mathcal{P}}) = \mathcal{P}_r \\ (\delta \widehat{W}_t)_{t=i}^{j} = w}} \Pr[\widehat{\mathcal{P}}] / \Pr[\mathcal{P}_r].$$

Algorithm 7-2 Local collision analysis (forward)

Let $(DV_t)_{t=0}^{79}$, $(DW_t)_{t=0}^{79}$, $(\mathcal{W}_t)_{t=0}^{79}$ and $(\mathcal{Q}_{t+1})_{t=0}^{79}$ be as defined in Section 7.5.11 and let $I = [t_b, t_e]$ be an interval of steps $t_b, \ldots, t_e$ such that $DV_i = 0$ for $i \in \{t_b{-}5, \ldots, t_b{-}1\}$.

1. We start with $\mathcal{A}_{t_b,t_b} = \emptyset$;
2. For all differential paths $\mathcal{P}$ over step t_b of the following form:

$$\delta RL(Q_{t_b-4}, 30) = 0, \quad \Delta Q_{t_b-3} = \ldots = \Delta Q_{t_b} = 0, \quad \Delta F_{t_b} = 0,$$

$$\delta Q_{t_b+1} = \delta W_{t_b} \in \mathcal{W}_{t_b} \cap \sigma(\mathcal{Q}_{t_b+1}),$$

   we insert the tuple $(\mathcal{P}, \{((\delta W_i)_{i=t_b}^{t_b}, 1)\})$ in the set $\mathcal{A}_{t_b,t_b}$;
3. For steps $t = t_b + 1, \ldots, t_e$ in that order we do the following:
4. Let $\mathcal{A}_{t_b,t} = \emptyset$;
5. For all tuples $(\mathcal{P}, \mathcal{S}) \in \mathcal{A}_{t_b,t-1}$ we extend $\mathcal{P}$ forward:
6. For all BSDRs $\Delta Q_t \in \mathcal{Q}_t$ of δQ_t:
7. For all $\delta W_t \in \mathcal{W}_t$, $\delta Q_{t+1} \in \sigma(\mathcal{Q}_{t+1})$ and ΔF_t such that

$$(\Delta F_t[b])_{b=0}^{31} \in (V_{t,b})_{b=0}^{31} \quad \dagger \quad \text{and}$$

$$\delta Q_{t+1} = \sigma(RL(\Delta Q_t, 5)) + \sigma(RL(\Delta Q_{t-4}, 30)) + \sigma(\Delta F_t) + \delta W_t$$

   do the following:
8. Let $\mathcal{P}_e$ be $\mathcal{P}$ extended with step t using ΔQ_t, ΔF_t, δW_t and δQ_{t+1};
9. If $\Pr[\mathcal{P}_e] > 0$ then do:
10. Let $\mathcal{P}_r = \text{Reduce}(\mathcal{P}_e)$, $p_r = \Pr[\mathcal{P}_e] / \Pr[\mathcal{P}_r]$ and

$$\widehat{\mathcal{S}} = \left\{ ((\delta\widehat{W}_i)_{i=t_b}^{t}, p_r \cdot \widehat{p}) \;\middle|\; ((\delta\widehat{W}_i)_{i=t_b}^{t-1}, \widehat{p}) \in \mathcal{S} \right\},$$

    where $\delta\widehat{W}_t = \delta W_t$;
11. If there exists a tuple $(\mathcal{P}_r, \widetilde{\mathcal{S}}) \in \mathcal{A}_{t_b,t}$ for some $\widetilde{\mathcal{S}}$ then replace $(\mathcal{P}_r, \widetilde{\mathcal{S}})$ in $\mathcal{A}_{t_b,t}$ by the tuple $(\mathcal{P}_r, \mathcal{S}_{\mathcal{P}_r})$, where

$$\mathcal{S}_{\mathcal{P}_r} = \left\{ \left(w, \sum_{(w,p') \in \widehat{\mathcal{S}} \cup \widetilde{\mathcal{S}}} p' \right) \;\middle|\; w \in \left\{ w' \middle| (w', p') \in \widehat{\mathcal{S}} \cup \widetilde{\mathcal{S}} \right\} \right\},$$

12. otherwise insert $(\mathcal{P}_r, \widehat{\mathcal{S}})$ in $\mathcal{A}_{t_b,t}$.
13. Return $\mathcal{A}_{t_b,t_e}$.

† See Equation 7.16 (p. 142).

The set $\mathcal{S}_{\mathcal{P}_r}$ for a reduced differential path $\mathcal{P}_r \in \mathcal{R}_{[i,j]}$ is defined as the set of all possible tuples $(w, p_{(w,\mathcal{P}_r,[i,j])})$ with $w \in (\mathcal{W}_t)_{t=i}^{j}$ and $p_{(w,\mathcal{P}_r,[i,j])} > 0$. Then for $0 \le i \le j < 80$, the set $\mathcal{A}_{i,j}$ is defined as the set of all tuples $(\mathcal{P}_r, \mathcal{S}_{\mathcal{P}_r})$ of differential paths $\mathcal{P}_r \in \mathcal{R}_{[i,j]}$ with $\mathcal{S}_{\mathcal{P}_r} \neq \emptyset$. The set $\mathcal{A}_{t_b,t_e}$ thus consists of all reduced versions of differential paths in $\mathcal{D}_{[t_b,t_e]}$ together with the intermediary probabilities for each possible message difference vector. To compute the set $\mathcal{A}_{t_b,t_e}$, we use Algorithm 7-2

Algorithm 7-3 Local collision analysis (backward)

Let $(DV_t)_{t=0}^{79}$, $(DW_t)_{t=0}^{79}$, $(\mathcal{W}_t)_{t=0}^{79}$ and $(\mathcal{Q}_{t+1})_{t=0}^{79}$ be as defined in Section 7.5.11 and let $I = [t_b, t_e]$ be an interval of steps $t_b, \ldots, t_e$ such that $DV_i = 0$ for $i \in \{t_e{-}4, \ldots, t_e\}$.

1. We start with $\mathcal{A}_{t_e,t_e} = \emptyset$;
2. For all differential paths $\mathcal{P}$ over step t_e of the following form:

$$\Delta Q_{t_e-3} = \ldots = \Delta Q_{t_e} = 0, \quad \delta Q_{t_e+1} = 0, \quad \Delta F_{t_e} = 0,$$

$$\delta RL(Q_{t_e-4}, 30) = -\delta W_{t_e} \in \sigma(RL(\mathcal{Q}_{t_e-4}, 30)), \quad \delta W_{t_e} \in \mathcal{W}_{t_e}$$

   we insert the tuple $(\mathcal{P}, \{((\delta W_i)_{i=t_e}^{t_e}, 1)\})$ in the set $\mathcal{A}_{t_e,t_e}$;
3. For steps $t = t_e - 1, \ldots, t_b$ in that order we do the following:
4. Let $\mathcal{A}_{t,t_e} = \emptyset$;
5. For all tuples $(\mathcal{P}, \mathcal{S}) \in \mathcal{A}_{t+1,t_e}$ we extend $\mathcal{P}$ backwards:
6. For all BSDRs $Y \in \mathcal{Q}_{t-3}$ with $\sigma(RL(Y, 30)) = \delta RL(Q_{t-3}, 30)$:
7. Let $\Delta Q_{t-3} = Y$;
8. For all $\delta W_t \in \mathcal{W}_t$, $\delta RL(Q_{t-4}, 30) \in \sigma(RL(\mathcal{Q}_{t-4}, 30))$ and ΔF_t such that $(\Delta F_t[b])_{b=0}^{31} \in (V_{t,b})_{b=0}^{31}$ and

$$\delta RL(Q_{t-4}, 30) = \sigma(\Delta Q_{t+1}) - \sigma(RL(\Delta Q_t, 5)) - \sigma(\Delta F_t) - \delta W_t$$

   do the following:
9. Let $\mathcal{P}_e$ be $\mathcal{P}$ extended with step t using $\delta RL(Q_{t-4}, 30)$, ΔQ_{t-3}, ΔF_t and δW_t;
10. If $\Pr[\mathcal{P}_e] > 0$ then do:
11. Let $\mathcal{P}_r = \text{Reduce}(\mathcal{P}_e)$, $p_r = \Pr[\mathcal{P}_e] / \Pr[\mathcal{P}_r]$ and

$$\widehat{\mathcal{S}} = \left\{ ((\delta\widehat{W}_i)_{i=t}^{t_e}, p_r \cdot \widehat{p}) \;\middle|\; ((\delta\widehat{W}_i)_{i=t+1}^{t_e}, \widehat{p}) \in \mathcal{S} \right\},$$

    where $\delta\widehat{W}_t = \delta W_t$;
12. If there exists a tuple $(\mathcal{P}_r, \widetilde{\mathcal{S}}) \in \mathcal{A}_{t,t_e}$ for some $\widetilde{\mathcal{S}}$ then replace $(\mathcal{P}_r, \widetilde{\mathcal{S}})$ in $\mathcal{A}_{t,t_e}$ by the tuple $(\mathcal{P}_r, \mathcal{S}_{\mathcal{P}_r})$, where

$$\mathcal{S}_{\mathcal{P}_r} = \left\{ \left(w, \sum_{(w,p') \in \widehat{\mathcal{S}} \cup \widetilde{\mathcal{S}}} p' \right) \;\middle|\; w \in \left\{ w' \middle| (w', p') \in \widehat{\mathcal{S}} \cup \widetilde{\mathcal{S}} \right\} \right\},$$

13. otherwise insert $(\mathcal{P}_r, \widehat{\mathcal{S}})$ in $\mathcal{A}_{t,t_e}$.
14. Return $\mathcal{A}_{t_b,t_e}$.

(p. 150) to iteratively construct the sets $\mathcal{A}_{t_b,j}$ for $j = t_b, \ldots, t_e$.

Since $DV_t = 0$ for $t = t_b - 5, \ldots, t_b - 1$ and for $t = t_e - 4, \ldots, t_e$ for this single local collision, it follows that for all $(\mathcal{P}, \mathcal{S}) \in \mathcal{A}_{t_b,t_e}$ we have for $\mathcal{P}$ by definition $\delta RL(Q_{t_b-4}, 30) = 0$, $\Delta Q_i = (0)_{j=0}^{31}$ for $i = t_b - 3, \ldots, t_b$, $\Delta Q_i = (0)_{j=0}^{31}$ for $i =$

$t_e - 3, \ldots, t_e$ and $\delta Q_{t_e+1} = 0$. Observation 7.1 (p. 148) implies that $\mathcal{P}$ is trivial:

$$\mathcal{P} = (0, ((0)_{j=0}^{31})_{i=t_b-4}^{t_e+1}, 0, ((0)_{j=0}^{31})_{i=t_b}^{t_e}, (0)_{i=t_b}^{t_e}).$$

Hence, $\mathcal{A}_{t_b,t_e}$ consists of a single tuple $(\mathcal{P}, \mathcal{S})$ where $\mathcal{P}$ is trivial and the set $\mathcal{S}$ contains all tuples $((\delta W_t)_{t=t_b}^{t_e}, p)$ where $p > 0$ by definition is the desired success probability $p_{w,[t_b,t_e]}$ of $w = (\delta W_t)_{t=t_b}^{t_e}$. Most importantly we have determined the highest success probability $p_{[t_b,t_e]} = \max\{p \mid (w,p) \in \mathcal{S}\}$ and all message difference vectors that attain that probability: $\{w \mid (w, p_{[t_b,t_e]}) \in \mathcal{S}\}$.

A similar analysis is possible backwards by iteratively constructing sets $\mathcal{A}_{i,t_e}$ for $i = t_e, \ldots, t_b$ using Algorithm 7-3 (p. 151). Note that Algorithm 7-2 expects $DV_t = 0$ for $t \in \{t_b - 5, \ldots, t_b - 1\}$, whereas Algorithm 7-3 expects $DV_t = 0$ for $t \in \{t_e - 4, \ldots, t_e\}$. Hence, working backwards provides an alternative whenever $DV_t \neq 0$ for some $t \in \{t_b - 5, \ldots, t_b - 1\}$ and $DV_t = 0$ for $t \in \{t_e - 4, \ldots, t_e\}$.

7.5.6 Message difference compression

Since the number of possible message difference vectors grows exponentially in the number of steps, we employ another technique that allows us to 'combine' message difference vectors. This is nothing more than a smart representation of $\mathcal{A}_{i,j}$ by a pair $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ that has significantly reduced memory footprint and also allows an optimization that reduces the runtime complexity. In the implementations of Algorithms 7-2 and 7-3, we use the pair $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ representing $\mathcal{A}_{i,j}$ to improve runtime and memory complexity. Nevertheless, in the upcoming sections we still use $\mathcal{A}_{i,j}$ instead of the equivalent representation $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ for ease of notation.

First we introduce some necessary notation. We denote a message difference vector substitution rule as $(w_n)_{n=i}^{j} \hookrightarrow (v_n)_{n=i}^{j}$. For a message difference vector $(x_k)_{k=l}^{m}$ and a message difference vector substitution rule $(w_n)_{n=i}^{j} \hookrightarrow (v_n)_{n=i}^{j}$ such that $l \leq i \leq j \leq m$, we define $\Pi((x_k)_{k=l}^{m}, (w_n)_{n=i}^{j} \hookrightarrow (v_n)_{n=i}^{j})$ as the message difference vector $(y_k)_{k=l}^{m}$, where for $k \in \{l, \ldots, m\}$:

$$y_k = \begin{cases} x_k & \text{if } (x_n)_{n=i}^{j} \neq (w_n)_{n=i}^{j} \vee k \notin \{i, \ldots, j\}; \\ v_k & \text{otherwise.} \end{cases}$$

This definition of Π is extended to a set $\mathcal{SR}$ of message difference vector substitution rules:

$$\Pi(X, \mathcal{SR}) = \left\{\Pi(\cdots \Pi(X, s_1), \cdots, s_n) \;\middle|\; n \in \{0, \ldots, |\mathcal{SR}|\}, s_1, \ldots, s_n \in \mathcal{SR}\right\},$$

which includes X itself by using $n = 0$.

Now we can represent $\mathcal{A}_{i,j}$ in Algorithms 7-2 and 7-3 as a pair $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ of a set of substitution rules $\mathcal{SR}_{i,j}$ and a set $\mathcal{B}_{i,j}$ of the form

$$\mathcal{B}_{i,j} = \{(\mathcal{P}, \mathcal{S}'_{\mathcal{P}}) \mid (\mathcal{P}, \mathcal{S}_{\mathcal{P}}) \in \mathcal{A}_{i,j}\}$$

such that for all $\mathcal{P}$ the set $\mathcal{S}'_{\mathcal{P}}$ together with $\mathcal{SR}_{i,j}$ generates $\mathcal{S}_{\mathcal{P}}$:

$$\mathcal{S}_{\mathcal{P}} = \bigcup_{(w,p)\in\mathcal{S}'_{\mathcal{P}}} \{(v,p) \mid v \in \Pi(w, \mathcal{SR}_{i,j})\}.^{33}$$

To obtain a representation $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ of $\mathcal{A}_{i,j}$, we do the following. We start with $\mathcal{B}_{t_b,t} = \mathcal{A}_{t_b,t}$ and $\mathcal{SR}_{t_b,t} = \emptyset$. Let

$$\mathcal{W} = \{w \mid (w,p) \in \mathcal{S}_{\mathcal{P}}, (\mathcal{P}, \mathcal{S}_{\mathcal{P}}) \in \mathcal{A}_{i,j}\}$$

be the set of all message difference vectors w under consideration. For each $w \in \mathcal{W}$ we define the function θ_w that maps differential paths $\mathcal{P}$ to the respective intermediary probability of w and $\mathcal{P}$:

$$\theta_w : \{\mathcal{P} \mid (\mathcal{P}, \mathcal{S}) \in \mathcal{A}_{i,j}\} \to [0,1]$$

$$\theta_w : \mathcal{P} \mapsto p, \quad \text{for which } (w,p) \in \mathcal{S}_{\mathcal{P}}$$

Now for all groups $w_1, \ldots, w_K \in \mathcal{W}$ of message difference vectors that have identical functions $\theta_{w_1} = \ldots = \theta_{w_K}$, we remove all occurrences of $w_2, \ldots, w_K$ in $\mathcal{B}_{i,j}$ and compensate by inserting message difference vectors substitution rules $w_1 \hookrightarrow w_2$, $w_1 \hookrightarrow w_3$, ..., $w_1 \hookrightarrow w_K$ into $\mathcal{SR}_{i,j}$.[34] We denote the resulting representation $(\mathcal{SR}_{i,j}, \mathcal{B}_{i,j})$ of $\mathcal{A}_{i,j}$ as $\text{Compact}(\mathcal{A}_{i,j})$.

Let $\mathcal{A}_{t_b,t} = \text{Extend}(\mathcal{A}_{t_b,t-1})$ denote the computation of $\mathcal{A}_{t_b,t}$ using $\mathcal{A}_{t_b,t-1}$ in steps 4–12 of Algorithm 7-2. Then $(\mathcal{SR}_{t_b,t}, \mathcal{B}_{t_b,t})$ can be computed from $(\mathcal{SR}_{t_b,t-1}, \mathcal{B}_{t_b,t-1})$ as

$$(\mathcal{SR}_{t_b,t}, \mathcal{B}_{t_b,t}) = \text{Compact}\big(\text{Extend}\big((\mathcal{SR}_{t_b,t-1}, \mathcal{B}_{t_b,t-1})\big)\big).$$

Although this implies that we still need to store the entire set $\mathcal{A}_{t_b,t}$ at some point.

Note that in the steps 4 to 12 of Algorithm 7-2 for any $\mathcal{P}$ the values $\mathcal{P}_r$ and p_r do not directly depend on values $w \in \{v \mid (v,p) \in \mathcal{S}\}$. One can thus observe that for any two message difference vectors w and v used in $\mathcal{A}_{i,j}$ for which $\theta_w = \theta_v$, it follows that also $\theta_{w'} = \theta_{v'}$ for any extensions $w' = w||x$ and $v' = v||x$ of w and v by the same message differences x (as in step 10 of Algorithm 7-2). This implies that in the reduction of $\mathcal{A}_{t_b,t}$ to $(\mathcal{SR}_{t_b,t}, \mathcal{B}_{t_b,t}) = \text{Compact}(\mathcal{A}_{t_b,t})$ all these message difference vectors v' can be removed by using the substitution rule $w \hookrightarrow v$.

More specifically, let $(\mathcal{SR}_{t_b,t-1}, \mathcal{B}_{t_b,t-1})$ be a representation of $\mathcal{A}_{t_b,t-1}$, then for step t in Algorithm 7-2 the above observation holds for all w and $v \in \Pi(w, \mathcal{SR}_{t_b,t-1})$ and thus all extensions of such v in $\mathcal{A}_{t_b,t}$ can be removed and compensated by the substitution rule $w \hookrightarrow v$. This implies that

$$(\mathcal{SR}_{t_b,t-1}, \text{Extend}(\mathcal{B}_{t_b,t-1}))$$

33. This directly implies that for $\mathcal{SR}_{i,j} = \emptyset$ we have that $\mathcal{B}_{i,j} = \mathcal{A}_{i,j}$. Note that for any two message difference vectors w and $v \in \Pi(w, \mathcal{SR}_{i,j})$ their respective probabilities for all differential paths $\mathcal{P}$ must be equal: $\forall \mathcal{P} : (w,p) \in \mathcal{S}_{\mathcal{P}} \Leftrightarrow (v,p) \in \mathcal{S}_{\mathcal{P}}$.

34. To efficiently determine these groups with identical functions θ_w, we compute and compare hash values of a representation of θ_w for all $w \in \mathcal{W}$ instead of these function θ_w themselves.

is a representation of $\mathcal{A}_{t_b,t}$, which naturally may be further reduced.

We can thus obtain a representation $(\mathcal{SR}_{t_b,t}, \mathcal{B}_{t_b,t})$ of $\mathcal{A}_{t_b,t}$ more efficiently than simply computing $\text{Compact}\big(\text{Extend}\big((\mathcal{SR}_{t_b,t-1}, \mathcal{B}_{t_b,t-1})\big)\big)$ as follows:

$$(\widetilde{\mathcal{SR}}_{t_b,t}, \widetilde{\mathcal{B}}_{t_b,t}) = \text{Compact}(\text{Extend}(\mathcal{B}_{t_b,t-1}));$$
$$(\mathcal{SR}_{t_b,t}, \mathcal{B}_{t_b,t}) = (\widetilde{\mathcal{SR}}_{t_b,t} \cup \mathcal{SR}_{t_b,t-1}, \widetilde{\mathcal{B}}_{t_b,t}).$$

In this manner we can obtain a representation of the set $\mathcal{A}_{t_b,t_e}$, but in a way that does not require anymore that the entire set $\mathcal{A}_{t_b,t_e}$ must be computed and stored in memory. Furthermore, only the message difference vectors present in $\mathcal{B}_{t_b,t-1}$ are processed instead of all message difference vectors present in $\mathcal{A}_{t_b,t-1}$ in the processing of step t in step 3 of Algorithm 7-2.

Algorithm 7-3 is treated analogously.

7.5.7 Single local collision analysis: δIHV_{diff}

Consider a single local collision starting at step $t_b \geq 75$ and ending with step $t_e = 79$. Since $79 = t_e < t_b + 5$, not all corrections have been made after step 79 and it follows that $\mathcal{A}_{t_b,t_e}$ does not consist of a single trivial reduced differential path. In this case we are interested in the most likely differences $\delta IHV_{\text{diff}} = \delta IHV_{\text{out}} - \delta IHV_{\text{in}}$ added to the IHV according to a differential path $\mathcal{P}$ over steps $t_b, \ldots, 79$:

$$\phi(\widehat{\mathcal{P}}) = (d_i)_{i=76}^{80}, \quad d_i = \begin{cases} \sigma(RL(\Delta\widehat{Q}_i, 30)), & i = 76, 77, 78; \\ \sigma(\Delta\widehat{Q}_i), & i = 79; \\ \delta\widehat{Q}_i, & i = 80. \end{cases}$$

Using Algorithm 7-2 we compute the set $\mathcal{A}_{t_b,79}$ and we determine the set $\mathcal{I}$ of possible δIHV_{diff}: $\mathcal{I} = \{\phi(\widehat{\mathcal{P}}) \mid (\widehat{\mathcal{P}}, \mathcal{S}) \in \mathcal{A}_{t_b,79}\}$.

We can determine the success probability $p_{(w,\delta IHV_{\text{diff}},[t_b,79])}$ for each message difference vector $w = (\delta W_t)_{t=t_b}^{79} \in (\mathcal{W}_t)_{t=t_b}^{79}$ and for each $\delta IHV_{\text{diff}} \in \mathcal{I}$:

$$\begin{aligned} p_{(w,\delta IHV_{\text{diff}},[t_b,79])} &= \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,79]} \\ \phi(\widehat{\mathcal{P}}) = \delta IHV_{\text{diff}} \\ (\delta\widehat{W}_t)_{t=t_b}^{79} = w}} \Pr[\widehat{\mathcal{P}}] \\ &= \sum_{\substack{(\widehat{\mathcal{P}}, \mathcal{S}) \in \mathcal{A}_{t_b,79} \\ \phi(\widehat{\mathcal{P}}) = \delta IHV_{\text{diff}}}} \sum_{(w, p_{(w,\widehat{\mathcal{P}},[t_b,79])}) \in \mathcal{S}} p_{(w,\widehat{\mathcal{P}},[t_b,79])} \cdot \Pr[\widehat{\mathcal{P}}]. \end{aligned}$$

Finally, we can determine the highest success probability:

$$p_{[t_b,79]} = \max\{p_{(w,\delta IHV_{\text{diff}},[t_b,79])} \mid \delta IHV_{\text{diff}} \in \mathcal{I},\ w \in (\mathcal{W}_t)_{t=t_b}^{79}\}$$

and all δIHV_{diff} that attain that probability (almost):

$$\{\delta IHV_{\text{diff}} \mid \delta IHV_{\text{diff}} \in \mathcal{I},\ \exists w : p_{(w,\delta IHV,[t_b,79])} \geq p_{[t_b,79]} \cdot \alpha\},$$

where $\alpha \in [0,1]$ is a given factor, i.e., $\alpha = 0.95$. For each such δIHV_{diff} we determine all w for which $p_{(w,\delta IHV,[t_b,79])} \geq p_{[t_b,79]} \cdot \alpha$:

$$\{w \mid w \in (\mathcal{W}_t)_{t=t_b}^{79},\ p_{(w,\delta IHV_{\text{diff}},[t_b,79])} \geq p_{[t_b,79]} \cdot \alpha\}.$$

7.5.8 Single local collision analysis: alternative δIHV_{diff}

Our analysis so far only allowed working state differences as prescribed by the disturbances in $(DV_t)_{t=0}^{79}$ either with or without carries. However, for the last steps it is not necessary anymore to restrict ΔQ_t. We are interested in δIHV_{diff} with high probability, not necessarily one exactly as prescribed by disturbances in $(DV_t)_{t=0}^{79}$. To that end we choose $\mathcal{Q}_t$ for $t \in \{76,\ldots,80\}$ as the set of all low weight BSDRs or even by the set of all BSDRs as desired. Then using Algorithm 7-4 we iteratively generate sets $\mathcal{Y}_{t_b,i}$ for $i = t_b,\ldots,79$ that are defined as the set of all tuples $(\mathcal{P}_r, s_{\mathcal{P}_r})$ of differential paths $\mathcal{P}_r \in \mathcal{R}_{[t_b,i]}$ and associated weights $s_{\mathcal{P}_r}$. The associated weight $s_{\mathcal{P}_r}$ of a reduced differential path $\mathcal{P}_r \in \mathcal{R}_{[t_b,i]}$ is defined as

$$s_{\mathcal{P}_r} = \sum_{w \in (\mathcal{W})_{t=t_b}^{i}} p_{(w,\mathcal{P}_r,[t_b,i])},$$

where $p_{(w,\mathcal{P}_r,[t_b,i])}$ is the probability as defined earlier (using our choice of $\mathcal{Q}_t$).

The last set $\mathcal{Y}_{t_b,79}$ is used to determine for possible δIHV_{diff} a weight $s_{\delta IHV_{\text{diff}}}$ which is the sum of success probabilities over allowed message difference vectors:

$$\begin{aligned}
s_{\delta IHV_{\text{diff}}} &= \sum_{\substack{(\mathcal{P},s_{\mathcal{P}}) \in \mathcal{Y}_{t_b,79} \\ \phi(\mathcal{P}) = \delta IHV_{\text{diff}}}} s_{\mathcal{P}} \cdot \Pr[\mathcal{P}] \\
&= \sum_{\substack{(\mathcal{P},s_{\mathcal{P}}) \in \mathcal{Y}_{t_b,79} \\ \phi(\mathcal{P}) = \delta IHV_{\text{diff}}}} \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{79}} \Pr[\mathcal{P}] \cdot p_{(w,\mathcal{P},[t_b,79])} \\
&= \sum_{\substack{(\mathcal{P},s_{\mathcal{P}}) \in \mathcal{Y}_{t_b,79} \\ \phi(\mathcal{P}) = \delta IHV_{\text{diff}}}} \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{79}} \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,79]} \\ \text{Reduce}(\widehat{\mathcal{P}}) = \mathcal{P} \\ (\delta \widehat{W}_t)_{t=t_b}^{79} = w}} \Pr[\widehat{\mathcal{P}}] \\
&= \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{79}} \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,79]} \\ \phi(\widehat{\mathcal{P}}) = \delta IHV_{\text{diff}} \\ (\delta \widehat{W}_t)_{t=t_b}^{79} = w}} \Pr[\widehat{\mathcal{P}}] \\
&= \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,79]} \\ \phi(\widehat{\mathcal{P}}) = \delta IHV_{\text{diff}}}} \Pr[\widehat{\mathcal{P}}].
\end{aligned}$$

Algorithm 7-4 δIHV_{diff} analysis (forward)

Let $(DV_t)_{t=0}^{79}$, $(DW_t)_{t=0}^{79}$, $(\mathcal{W}_t)_{t=0}^{79}$ and $(\mathcal{Q}_{t+1})_{t=0}^{79}$ be as defined in Section 7.5.11 and let $I = [t_b, 79]$ be an interval of steps $t_b, \ldots, 79$ such that $DV_i = 0$ for $i \in \{t_b - 5, \ldots, t_b - 1\}$.

1. We start with $\mathcal{Y}_{t_b,t_b} = \emptyset$;
2. For all differential paths $\mathcal{P}$ over step t_b of the following form:

$$\delta RL(Q_{t_b-4}, 30) = 0, \quad \Delta Q_{t_b-3} = \ldots = \Delta Q_{t_b} = 0, \quad \Delta F_{t_b} = 0,$$

$$\delta Q_{t_b+1} = \delta W_{t_b} \in \mathcal{W}_{t_b} \cap \sigma(\mathcal{Q}_{t_b+1}),$$

   we insert the tuple $(\mathcal{P}, 1)$ in the set $\mathcal{Y}_{t_b,t_b}$;
3. For steps $t = t_b + 1, \ldots, 79$ in that order we do the following:
4. Let $\mathcal{Y}_{t_b,t} = \emptyset$;
5. For all tuples $(\mathcal{P}, s) \in \mathcal{Y}_{t_b,t-1}$ we extend $\mathcal{P}$ forward:
6. For all BSDRs $\Delta Q_t \in \mathcal{Q}_t$ of δQ_t:
7. For all $\delta W_t \in \mathcal{W}_t$, $\delta Q_{t+1} \in \sigma(\mathcal{Q}_{t+1})$ and ΔF_t such that

$$(\Delta F_t[b])_{b=0}^{31} \in (V_{t,b})_{b=0}^{31} \quad \dagger \quad \text{and}$$

$$\delta Q_{t+1} = \sigma(RL(\Delta Q_t, 5)) + \sigma(RL(\Delta Q_{t-4}, 30)) + \sigma(\Delta F_t) + \delta W_t$$

   do the following:
8. Let $\mathcal{P}_e$ be $\mathcal{P}$ extended with step t using ΔQ_t, ΔF_t, δW_t and δQ_{t+1};
9. If $\Pr[\mathcal{P}_e] > 0$ then do:
10. Let $\mathcal{P}_r = \text{Reduce}(\mathcal{P}_e)$ and $s_r = s \cdot \Pr[\mathcal{P}_e] / \Pr[\mathcal{P}_r]$;
11. If there exists a tuple $(\mathcal{P}_r, \widetilde{s}) \in \mathcal{Y}_{t_b,t}$ for some $\widetilde{s}$ then replace $(\mathcal{P}_r, \widetilde{s})$ in $\mathcal{Y}_{t_b,t}$ by the tuple $(\mathcal{P}_r, s_r + \widetilde{s})$,
12. otherwise insert $(\mathcal{P}_r, s_r)$ in $\mathcal{Y}_{t_b,t}$.
13. Return $\mathcal{Y}_{t_b,79}$.

† See Equation 7.16 (p. 142).

We select a set $\widehat{\mathcal{I}}$ of differences δIHV_{diff} with high weights $s_{\delta IHV_{\text{diff}}}$ and define the sets $\mathcal{Q}_{\text{ihv},t}$ for $t = 1, \ldots, 80$:

$$\mathcal{Q}_{\text{ihv},t} = \begin{cases} \left\{ \Delta Q_t \;\middle|\; \exists (X_i)_{i=76}^{80} \in \widehat{\mathcal{I}} : \sigma(RL(\Delta Q_t, 30)) = X_t \right\}, & t \in \{76, 77, 78\}; \\ \left\{ \Delta Q_t \;\middle|\; \exists (X_i)_{i=76}^{80} \in \widehat{\mathcal{I}} : \sigma(\Delta Q_t) = X_t \right\}, & t \in \{79, 80\}; \\ \mathcal{Q}_t, & t < 76. \end{cases}$$

We can define related sets $\mathcal{Q}_{\text{ihv},u,t}$ for $u \in \{0,\ldots,32\}$ and $t \in \{1,\ldots,80\}$ where the weights are bounded by u plus the NAF weight:

$$\mathcal{Q}_{\text{ihv},u,t} = \{Y \mid Y \in \mathcal{Q}_{\text{ihv},t} \wedge w(Y) \leq w(\text{NAF}(\sigma(Y))) + u\}.$$

By repeating the analysis in Section 7.5.7 using the sets $\mathcal{Q}_{\text{ihv},u,t}$ for $\mathcal{Q}_t$, we can determine the success probabilities for each $\delta IHV_{\text{diff}} \in \widehat{\mathcal{I}}$. In this way we also find the highest success probability among those δIHV_{diff} as well as the message difference vectors that enable this highest success probability.

7.5.9 Single local collision analysis: round 1

If $DV_t = 0$ for $t \in \{t_e - 4,\ldots,t_e\}$ but not for all $t \in \{t_b - 5,\ldots,t_b - 1\}$ (e.g., if $20 = t_b > t_e - 5$) then the local collision is truncated and $\mathcal{A}_{t_b,t_e}$ does not consist of a single trivial reduced differential path. Instead we assume that t_b is chosen as the first step whose success probability should be taken into account, i.e., steps before t_b are ignored in the cost of a local collision since they are assumed to be fulfilled by message modification techniques. Thus we are interested in the most likely working state differences $\Lambda = \psi(\widehat{\mathcal{P}})$ of $\widehat{\mathcal{P}}$:

$$\psi(\widehat{\mathcal{P}}) = (d_i)_{i=t_b-4}^{t_b}, \quad d_i = \begin{cases} RR(\text{NAF}(\delta RL(\widehat{Q}_i, 30)), 30), & i = t_b - 4; \\ \Delta\widehat{Q}_i, & i = t_b - 3,\ldots,t_b; \end{cases}$$

Here we use $\Delta\widehat{Q}_t$ instead of $\delta\widehat{Q}_t$ as we assume that the final bitconditions disallow additional carries. Nevertheless, it is only a minor modification to use $\delta\widehat{Q}_t$ in the analysis below. The following analysis is very similar to the previous section.

Using Algorithm 7-3 we compute the set $\mathcal{A}_{t_b,t_e}$ and we determine the set $\mathcal{J}$ of possible Λ: $\mathcal{J} = \{\psi(\widehat{\mathcal{P}}) \mid (\widehat{\mathcal{P}},\mathcal{S}) \in \mathcal{A}_{t_b,t_e}\}$. We can determine the success probability $p_{(w,\Lambda,[t_b,t_e])}$ given message difference vector $w = (\delta W_t)_{t=t_b}^{t_e} \in (\mathcal{W}_t)_{t=t_b}^{t_e}$ for each $\Lambda \in \mathcal{J}$:

$$\begin{aligned} p_{(w,\Lambda,[t_b,t_e])} &= \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ \psi(\widehat{\mathcal{P}}) = \Lambda \\ (\delta\widehat{W}_t)_{t=t_b}^{t_e} = w}} \Pr[\widehat{\mathcal{P}}] \\ &= \sum_{\substack{(\widehat{\mathcal{P}},\mathcal{S}) \in \mathcal{A}_{t_b,t_e} \\ \psi(\widehat{\mathcal{P}}) = \Lambda}} \sum_{(w,p_{(w,\widehat{\mathcal{P}},[t_b,t_e])}) \in \mathcal{S}} p_{(w,\widehat{\mathcal{P}},[t_b,t_e])} \cdot \Pr[\widehat{\mathcal{P}}]. \end{aligned}$$

Finally, we can determine the highest success probability:

$$p_{[t_b,t_e]} = \max\{p_{(w,\Lambda,[t_b,t_e])} \mid \Lambda \in \mathcal{J},\ w \in (\mathcal{W}_t)_{t=t_b}^{t_e}\}$$

and all Λ that attain that probability:

$$\{\Lambda \mid \Lambda \in \mathcal{J},\ \exists w : p_{(w,\Lambda,[t_b,t_e])} = p_{[t_b,t_e]}\}.$$

For each such Λ we determine all w for which $p_{(w,\Lambda,[t_b,t_e])} = p_{[t_b,t_e]}$:

$$\{w \mid w \in (\mathcal{W}_t)_{t=t_b}^{t_e},\ p_{(w,\delta IHV_{\text{diff}},[t_b,t_e])} = p_{[t_b,t_e]}\}.$$

7.5.10 Single local collision analysis: alternate round 1

In the first round the differential path construction algorithms from Section 7.4 deviate from the local collisions as prescribed by the disturbance vector. Therefore, similar to the analysis of the last few steps, one can also allow all differences in the first few steps if they lead to higher success probabilities. We are interested in Λ with high probability. To that end we choose $\mathcal{Q}_t$ for $t \in \{t_b - 4, \ldots, t_b\}$ as the set of all low weight BSDRs. Then using Algorithm 7-5 we iteratively generate sets $\mathcal{Z}_{i,t_e}$ for $i = t_e - 1, \ldots, t_b$ that are defined as the set of all tuples $(\mathcal{P}_r, s_{\mathcal{P}_r})$ of differential paths $\mathcal{P}_r \in \mathcal{R}_{[t_b,t_e]}$ and associated weights $s_{\mathcal{P}_r}$. For each $\mathcal{P}_r$, the associated weight $s_{\mathcal{P}_r}$ is defined as

$$s_{\mathcal{P}_r} = \sum_{w \in (\mathcal{W})_{t=i}^{t_e}} p_{(w,\mathcal{P}_r,[i,t_e])},$$

where $p_{(w,\mathcal{P}_r,[i,t_e])}$ is the probability as defined earlier (using our choice of $\mathcal{Q}_t$).[35]

The last set $\mathcal{Z}_{t_b,t_e}$ is used to determine for possible Λ a weight s_Λ which is the sum of success probabilities over allowed message difference vectors:

$$\begin{aligned}
s_\Lambda &= \sum_{\substack{(\mathcal{P},s_\mathcal{P}) \in \mathcal{Z}_{t_b,t_e} \\ \psi(\mathcal{P}) = \Lambda}} s_\mathcal{P} \cdot \Pr[\mathcal{P}] \\
&= \sum_{\substack{(\mathcal{P},s_\mathcal{P}) \in \mathcal{Z}_{t_b,t_e} \\ \psi(\mathcal{P}) = \Lambda}} \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{t_e}} \Pr[\mathcal{P}] \cdot p_{(w,\mathcal{P},[t_b,t_e])} \\
&= \sum_{\substack{(\mathcal{P},s_\mathcal{P}) \in \mathcal{Z}_{t_b,t_e} \\ \psi(\mathcal{P}) = \Lambda}} \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{t_e}} \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ \mathrm{Reduce}(\widehat{\mathcal{P}}) = \mathcal{P} \\ (\delta \widehat{W}_t)_{t=t_b}^{t_e} = w}} \Pr[\widehat{\mathcal{P}}] \\
&= \sum_{w \in (\mathcal{W}_t)_{t=t_b}^{t_e}} \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ \psi(\widehat{\mathcal{P}}) = \Lambda \\ (\delta \widehat{W}_t)_{t=t_b}^{t_e} = w}} \Pr[\widehat{\mathcal{P}}] \\
&= \sum_{\substack{\widehat{\mathcal{P}} \in \mathcal{D}_{[t_b,t_e]} \\ \psi(\widehat{\mathcal{P}}) = \Lambda}} \Pr[\widehat{\mathcal{P}}].
\end{aligned}$$

We select a set $\widehat{\mathcal{J}}$ of differences Λ with high weights s_Λ and define the sets $\mathcal{Q}_{\mathrm{rnd1},t}$ for $t = 1, \ldots, 80$:

$$\mathcal{Q}_{\mathrm{rnd1},t} = \begin{cases} \left\{ \Delta Q_t \;\middle|\; (\Delta Q_i)_{i=t_b-4}^{t_b} \in \widehat{\mathcal{J}} \right\}, & t_b - 4 \le t \le t_b; \\ \mathcal{Q}_t, & \text{otherwise.} \end{cases}$$

35. This definition of $s_{\mathcal{P}_r}$ matches the definition from Section 7.5.8.

Algorithm 7-5 Λ analysis (backward)

Let $(DV_t)_{t=0}^{79}$, $(DW_t)_{t=0}^{79}$, $(\mathcal{W}_t)_{t=0}^{79}$ and $(\mathcal{Q}_{t+1})_{t=0}^{79}$ be as defined in Section 7.5.11 and let $I = [t_b, t_e]$ be an interval of steps $t_b, \ldots, t_e$ such that $DV_i = 0$ for $i \in \{t_e{-}4, \ldots, t_e\}$.

1. We start with $\mathcal{Z}_{t_e,t_e} = \emptyset$;
2. For all differential paths $\mathcal{P}$ over step t_e of the following form:

$$\Delta Q_{t_e-3} = \ldots = \Delta Q_{t_e} = 0, \quad \delta Q_{t_e+1} = 0, \quad \Delta F_{t_e} = 0,$$

$$\delta RL(Q_{t_e-4}, 30) = -\delta W_{t_e} \in \sigma(RL(\mathcal{Q}_{t_e-4}, 30)), \quad \delta W_{t_e} \in \mathcal{W}_{t_e}$$

   we insert the tuple $(\mathcal{P}, 1)$ in the set $\mathcal{Z}_{t_e,t_e}$;
3. For steps $t = t_e - 1, \ldots, t_b$ in that order we do the following:
4. Let $\mathcal{Z}_{t,t_e} = \emptyset$;
5. For all tuples $(\mathcal{P}, s) \in \mathcal{Z}_{t+1,t_e}$ we extend $\mathcal{P}$ backwards:
6. For all BSDRs $Y \in \mathcal{Q}_{t-3}$ with $\sigma(RL(Y, 30)) = \delta RL(Q_{t-3}, 30)$:
7. Let $\Delta Q_{t-3} = Y$;
8. For all $\delta W_t \in \mathcal{W}_t$, $\delta RL(Q_{t-4}, 30) \in \sigma(RL(\mathcal{Q}_{t-4}, 30))$ and ΔF_t such that $(\Delta F_t[b])_{b=0}^{31} \in (V_{t,b})_{b=0}^{31}$ and

$$\delta RL(Q_{t-4}, 30) = \sigma(\Delta Q_{t+1}) - \sigma(RL(\Delta Q_t, 5)) - \sigma(\Delta F_t) - \delta W_t$$

   do the following:
9. Let $\mathcal{P}_e$ be $\mathcal{P}$ extended with step t using $\delta RL(Q_{t-4}, 30)$, ΔQ_{t-3}, ΔF_t and δW_t;
10. If $\Pr[\mathcal{P}_e] > 0$ then do:
11. Not let $\mathcal{P}_r = \text{Reduce}(\mathcal{P}_e)$ and $s_r = s \cdot \Pr[\mathcal{P}_e] / \Pr[\mathcal{P}_r]$;
12. If there exists a tuple $(\mathcal{P}_r, \widetilde{s}) \in \mathcal{Z}_{t,t_e}$ then replace $(\mathcal{P}_r, \widetilde{s})$ in $\mathcal{Z}_{t,t_e}$ by the tuple $(\mathcal{P}_r, s_r + \widetilde{s})$,
13. otherwise insert $(\mathcal{P}_r, s_r)$ in $\mathcal{Z}_{t,t_e}$.
14. Return $\mathcal{Z}_{t_b,t_e}$.

We can define related sets $\mathcal{Q}_{\text{rnd1},u,t}$ for $u \in \{0, \ldots, 32\}$ and $t \in \{1, \ldots, 80\}$ where the weights are bounded by u plus the NAF weight:

$$\mathcal{Q}_{\text{rnd1},u,t} = \{Y \mid Y \in \mathcal{Q}_{\text{rnd1},t} \wedge w(Y) \leq w(\text{NAF}(\sigma(Y))) + u\}.$$

By repeating the analysis at the beginning of this section using sets $\mathcal{Q}_{\text{rnd1},u,t}$ for $\mathcal{Q}_t$, we can determine the success probabilities for each $\Lambda \in \widehat{\mathcal{J}}$. In this way we also

find the highest success probability among those Λ as well as the message expansion differences that enable this highest success probability.

7.5.11 Disturbance vector analysis

Analyzing a disturbance vector $(DV_t)_{t=0}^{79}$ and associated message expansion XOR difference vector $(DW_t)_{t=0}^{79}$ for SHA-1 (or SHA-0) can now be done using the methods described in the previous sections. In this section we define several cost functions and compare them on local collision (in)dependence, (dis)allowed carries and on both full and truncated disturbance vectors.

First we define the sets $\mathcal{D}_{u,[i,j]}$ and $\mathcal{D}_{\mathrm{nc},[i,j]}$ as the set $\mathcal{D}_{[i,j]}$ when using the underlying sets $\mathcal{Q}_{\mathrm{c},u,t}$ and $\mathcal{Q}_{\mathrm{nc},t}$, respectively. We define the following disturbance vector cost functions for $u \in \{0,\ldots,32\}$:

$$\mathrm{FDC}_{u,t_b}\left((DV_t)_{t=0}^{79}\right) = \max_{w,\delta IHV_{\mathrm{diff}},\Lambda} \sum_{\substack{\widehat{\mathcal{P}}\in\mathcal{D}_{u,[t_b,79]} \\ (\delta\widehat{W}_t)_{t=t_b}^{79}=w \\ \phi(\widehat{\mathcal{P}})=\delta IHV_{\mathrm{diff}} \\ \psi(\widehat{\mathcal{P}})=\Lambda}} \Pr[\widehat{\mathcal{P}}]\cdot 2^{w(\Delta\widehat{Q}_{t_b-3})+w(\Delta\widehat{Q}_{t_b-2})};$$

$$\mathrm{FDN}_{t_b}\left((DV_t)_{t=0}^{79}\right) = \max_{w,\delta IHV_{\mathrm{diff}},\Lambda} \sum_{\substack{\widehat{\mathcal{P}}\in\mathcal{D}_{\mathrm{nc},[t_b,79]} \\ (\delta\widehat{W}_t)_{t=t_b}^{79}=w \\ \phi(\widehat{\mathcal{P}})=\delta IHV_{\mathrm{diff}} \\ \psi(\widehat{\mathcal{P}})=\Lambda}} \Pr[\widehat{\mathcal{P}}]\cdot 2^{w(\Delta\widehat{Q}_{t_b-3})+w(\Delta\widehat{Q}_{t_b-2})}.$$

The first disturbance vector cost function FDC_{u,t_b} is defined as the maximal success probability (with a correction) over steps $t_b,\ldots,79$ where carries are allowed (bounded by u plus NAF weight) taken over all combinations of message difference vector w, starting working state Λ and IHV differences $\delta IHV_{\mathrm{diff}}$. The second disturbance vector cost function FDN_{t_b} is defined identically except carries are not allowed. The correction $2^{w(\Delta\widehat{Q}_{t_b-3})+w(\Delta\widehat{Q}_{t_b-2})}$ (which is determined by Λ) is chosen so that FDC and FDN are more closely related to the near-collision search. In the near-collision search we use bitconditions $\mathfrak{q}_i$ to search for valid (Q_i, Q'_i) and only then we proceed to (Q_{i+1}, Q'_{i+1}). This correction models that the differential bitconditions in $\mathfrak{q}_{t_b-3}$ and $\mathfrak{q}_{t_b-2}$ have been pre-fulfilled in this manner.[36]

These two disturbance vector cost functions can be efficiently determined using the methods of the previous sections. To that end, we split the range of steps $t_b,\ldots,79$ into independent intervals. As the disturbance of a local collision at step t is corrected within the next five steps, if no other local collision is started within those five steps then we can split the interval $[t_b, t_e]$ into two independent intervals: $[t_b, t+5]$ and $[t+6, t_e]$. Using this rule we split the range of steps $[t_b, 79]$ into intervals $I_1 =$

36. More ideally, we would have liked to have a correction that also includes the boolean function bitconditions with respect to ΔF_{t_b} in $\mathfrak{q}_{t_b-3}$ and $\mathfrak{q}_{t_b-2}$. However, such a correction would no longer be completely determined by Λ.

$[t_{b,1}, t_{e,1}] = [t_b, t_{e,1}]$, $I_2 = [t_{b,2}, t_{e,2}], \ldots, I_K = [t_{b,K}, 79]$ until no interval can be split further. The disturbance vectors in consideration (see Table 7-2) always result in at least two intervals as is required by our analysis. Intervals $I_2, \ldots, I_{K-1}$ are analyzed as in Section 7.5.5. The first and last interval can be analyzed as in Sections 7.5.9 and 7.5.7, respectively.

These cost functions can also be applied when the disturbance vector consists of a single local collision. For an arbitrary disturbance vector we can treat local collisions as independent (even though they are not) by applying the cost function to each individual local collision in said disturbance vector and taking the product over the resulting probabilities. This allows us to compare our cost function which takes into account the dependence of local collisions with similar cost functions that are based on the assumption that local collisions are independent. More specifically, we define the function Ω that compresses consecutive '1'-bits[37] in DV_i and the function Ψ that splits a disturbance vector into separate disturbance vectors each containing a single disturbance:

$$\Omega\left((DV_i)_{i=0}^{79}\right) = \left(DV_i \wedge \left(\neg RL(DV_i, 1) \vee \left(1 + 2^2 + 2^{27}\right)\right)\right)_{i=0}^{79},$$

$$\Psi\left((DV_i)_{i=0}^{79}\right) = \left\{ \begin{array}{c} (Y_i)_{i=0}^{79} \text{ where} \\ Y_t[b]{=}1, \\ Y_i[j]{=}0,\ i{\neq}t \vee j{\neq}b \end{array} \middle| \begin{array}{c} t{\in}\{0,\ldots,79\} \\ b{\in}\{0,\ldots,31\} \\ \text{such that } DV_t[b]{=}1 \end{array} \right\}.$$

Now we can define the following variants on FDC and FDN that assume independence of local collisions:

$$\begin{aligned} \mathrm{FIC}_{u,t_b}\left((DV_t)_{t=0}^{79}\right) &= \prod_{(Y_i)_{i=0}^{79} \in \Psi(\Omega((DV_t)_{t=0}^{79}))} \mathrm{FDC}_{u,t_b}((Y_i)_{i=0}^{79}); \\ \mathrm{FIN}_{t_b}\left((DV_t)_{t=0}^{79}\right) &= \prod_{(Y_i)_{i=0}^{79} \in \Psi(\Omega((DV_t)_{t=0}^{79}))} \mathrm{FIC}_{t_b}((Y_i)_{i=0}^{79}). \end{aligned}$$

Most cost functions in the literature only consider local collisions starting at step t_b or thereafter, all other local collisions are ignored even if some of their corrections take place at step t_b or later. This can be applied by 'truncating' the disturbance vector:

$$\Gamma_{t_b}\left((DV_i)_{i=0}^{79}\right) = (Y_i)_{i=0}^{79}, \quad Y_i = \begin{cases} DV_i, & i \geq t_b; \\ 0, & i < t_b. \end{cases}$$

Using Γ_{t_b} we can define variants of FDC, FIC, FDN and FIN:

$$\begin{aligned} \mathrm{TDC}_{u,t_b}\left((DV_t)_{t=0}^{79}\right) &= \mathrm{FDC}_{u,t_b}\left(\Gamma_{t_b}\left((DV_t)_{t=0}^{79}\right)\right); \\ \mathrm{TIC}_{u,t_b}\left((DV_t)_{t=0}^{79}\right) &= \mathrm{FIC}_{u,t_b}\left(\Gamma_{t_b}\left((DV_t)_{t=0}^{79}\right)\right); \\ \mathrm{TDN}_{t_b}\left((DV_t)_{t=0}^{79}\right) &= \mathrm{FDN}_{t_b}\left(\Gamma_{t_b}\left((DV_t)_{t=0}^{79}\right)\right); \\ \mathrm{TIN}_{t_b}\left((DV_t)_{t=0}^{79}\right) &= \mathrm{FIN}_{t_b}\left(\Gamma_{t_b}\left((DV_t)_{t=0}^{79}\right)\right). \end{aligned}$$

37. Bits 0, 2 and 27 are always kept, since bit positions pairs (31,0), (1,2) and (26,27) are not considered consecutive. See also Section 7.3.2

Applying Γ_{t_b} to a disturbance vector implies that there are no differences in Λ. This in turn implies that the correction $2^{w(\Delta\widehat{Q}_{t_b-3})+w(\Delta\widehat{Q}_{t_b-2})}$ is always one.

Our cost functions consist of three letters that indicate the following properties:

F Full D.V.: uses all local collisions that influence steps $t \geq t_b$;

T Truncated D.V.: uses only local collisions starting at steps $t \geq t_b$;

D Use dependence of local collisions;

I Assume independence of local collisions;

C Allow additional carries (total weight bounded by u plus NAF weight);

N Disallow carries.

The last cost function we define here is $\mathrm{HW}_{t_b}\left((DV_t)_{t=0}^{79}\right) = \sum_{i=t_b}^{79} w(DV_i)$ which is the most crude cost function as it counts the number of '1'-bits in $DV_{t_b}, \ldots, DV_{79}$.

Below we provide selected results in Table 7-3 to compare cost functions for SHA-1. For the cases in Table 7-3, the results of the cost function $\mathrm{FDC}_{7,20}$ are a factor between $2^{0.3}$ and $2^{12.5}$ higher than $\mathrm{FIC}_{32,20}$. This clearly shows that using the joint probability instead of using the product of individual probabilities leads to higher maximum success probabilities.

To compare FDC with varying starting step t_b for SHA-1, selected results are shown in Table 7-4. A more complete analysis of disturbance vectors for SHA-1 using FDC can be found in Appendix F.

Section 7.6 uses this analysis to obtain target values for $\delta IHV_{\mathrm{diff}}$ and message bitrelations for disturbance vector II(52,0). The maximum success probabilities for intervals $[33, 52]$, $[53, 60]$ and $[67, 79]$ as determined using this analysis have been confirmed by experiments as described in Section 7.6.9.

Table 7-3: *SHA-1 disturbance vector analysis - cost function comparison*

DV	FDC	FIC	FDN	FIN	TDC	TIC	TDN	TIN	HW
I(48, 0)	71.4	80.5	77	83	65.4	74.5	71	77	27
I(49, 0)	72.2	79.6	77	83	67.2	74.6	72	78	27
I(50, 0)	71.9	81.4	75	83	65.9	73.4	69	75	26
I(51, 0)	73.3	85.8	77	88	67.3	74.8	71	77	25
I(48, 2)	73.8	75.7	79	79	69.8	71.7	75	75	27
I(49, 2)†	73.8	74.1	78	78	70.8	71.1	75	75	27
II(50, 0)	73.0	77.4	78	80	68.0	70.4	73	73	27
II(51, 0)	71.9	77.7	77	81	67.6	69.7	73	73	26
II(52, 0)‡	71.8	79.4	75	81	65.4	67.4	69	69	25

The eight columns FDC to TIN show the negative $\log_2$ results from the cost functions $\mathrm{FDC}_{7,20}$, FDN_{20}, $\mathrm{FIC}_{32,20}$, FIN_{20}, $\mathrm{TDC}_{7,20}$, FDN_{20}, $\mathrm{TIC}_{32,20}$, TIN_{20}, respectively. The last column shows the result from the cost function HW_{20}. The disturbance vectors marked by † and ‡ are used in Wang et al.'s collision attack [WYY05b] and our near-collision attack in Section 7.6, respectively.

Table 7-4: *SHA-1 disturbance vector analysis - FDC with varying starting step*

	t_b								
DV	18	19	20	21	22	23	24	25	26
I$(48, 0)$	78.3	75.3	71.4	70.4	67.4	66.4	65.0	63.0	61.0
I$(49, 0)$	80.2	75.2	72.2	69.3	68.3	65.3	64.3	62.9	60.9
I$(50, 0)$	79.9	75.9	71.9	70.9	68.1	67.1	64.1	63.1	61.7
II$(51, 0)$	78.7	74.9	71.9	68.9	67.9	66.5	64.5	58.5	56.5
II$(52, 0)$	78.6	75.6	71.8	69.8	66.8	65.8	64.3	62.3	56.3

The columns are the negative $\log_2$ results from the cost function FDC_{8,t_b}.

7.6 SHA-1 near-collision attack

7.6.1 Overview

In this section we present the construction of a near-collision attack against SHA-1 using our methods from Sections 7.4 and 7.5. The construction consists of the following steps that each are discussed in the following sections:

1. We discuss the selection of the disturbance vector and our choice for II(52,0) in Section 7.6.2.

2. In Section 7.6.3 we perform a precise analysis of disturbance vector II(52,0) to obtain sets of optimal values for δIHV_{diff} and Λ.

3. In Section 7.6.4 we construct valid first round differential paths where we use the found set of optimal values for Λ to derive the first round upper partial differential paths.

4. The bitconditions given by these first round differential paths are extended into round two in Section 7.6.5 for the purposes of the early-stopping and message modification techniques.

5. Section 7.6.6 details the derivation of the smallest set of message bitrelations over all four rounds that lead (almost) to the highest success probability under the restrictions given by the bitconditions over round one and two.

6. In Section 7.6.7 we present our basic collision searching algorithm that efficiently searches for message blocks that fulfill all bitconditions up to Q_{17} and all message bitrelations.

7. Lastly, we significantly speed up our basic collision searching algorithm using tunnels in Section 7.6.8.

Our implementation of this near-collision attack is published as part of project HashClash [HC]. The attack has a complexity equivalent to about $2^{57.5}$ calls to the SHA-1 compression function. This improves upon the near-collision attack by Wang et al. with a complexity of about 2^{68} SHA-1 compressions. So far no near-collision blocks have been found using our near-collision attack that provide explicit proof of the correctness and complexity of our attack. For that reason we discuss verification of the correctness and the complexity of our near-collision attack in Section 7.6.9.

Our near-collision attack can directly be used in a two-block identical-prefix collision attack against SHA-1. It should be noted that such a two-block identical-prefix collision attack actually consists of three blocks where the first block is part of the identical-prefix part and is used to gain 160-bits of freedom and to satisfy the bitconditions $\mathfrak{q}_{-4}, \ldots, \mathfrak{q}_0$ of our near-collision attack. The remaining two blocks are two sequential near-collision blocks where the second block cancels the δIHV_{out} resulting from the first block.

A lower-bound for the complexity of a complete two-block identical-prefix collision attack based on our current near-collision implementation is approximately $(1 + 6) \cdot 2^{57.5} \approx 2^{60.3}$ SHA-1 compressions. This follows from the fact that the first near-collision attack has the luxury of six allowed values for δIHV_{diff} for each possible vector of message differences $(\delta W_t)_{t=0}^{79}$, whereas the second near-collision attack must target one specific δIHV_{diff}.

The second near-collision attack has a lesser amount of freedom to exploit compared to the first near-collision attack, i.e., it cannot use a prior block to gain 160-bits of freedom and it requires more message bitrelations. This may further restrict the number of tunnels that can be used. Hence, the complexity of the identical-prefix collision attack as directly based on our near-collision attack will in all likelihood be larger than $2^{60.3}$ SHA-1 compressions. Nevertheless, we expect it will be only a small factor larger than this lower bound. Taking into account the extra message bitrelations and up to four fewer tunnels, one arrives at the conservative upper bound of $2^{65.3}$ SHA-1 compressions.

There is a wide gap between the lower bound $2^{60.3}$ and the conservative upper bound $2^{65.3}$ wherein the actual complexity of the second near-collision attack will lie. The complexity of a complete two-block identical-prefix collision attack may be more accurately estimated when the first near-collision block(s) has been found and thus the second near-collision attack can be constructed.

7.6.2 Disturbance vector selection

The selection of which disturbance vector to use is the most important choice in constructing a near-collision attack. This choice directly determines the message bit differences up to sign DW_t, the set of optimal δIHV_{diff} to target and the set of optimal Λ. It also determines most of the message bitrelations (some of which are only determined up to parity) and thereby also limits which tunnels may be used as is discussed in Section 7.6.8.

In Appendix F we present the results of our analysis of many disturbance vectors, see also Section 7.5. Table F-1 shows the results of the seven disturbance vectors that (almost) result in a success probability of 2^{-73} or higher over the last 60 steps: I(48,0), I(49,0), I(50,0), II(46,0), II(50,0), II(51,0) and II(52,0).

We have chosen for disturbance vector II(52,0) as it is the second best under our cost function $\text{FDC}_{u,20}$ *and* it showed the most promising results in a preliminary analysis from Section 7.5.8. We do not claim that disturbance vector II(52,0) is the best, since our choice is only based on the analysis of the last 60 steps and not on the other factors that the choice of disturbance vector influences such as number of message bitrelations and which tunnels can be used. In fact we encourage further analysis of the remaining disturbance vectors and construction of near-collision attacks under these disturbance vectors.

7.6.3 Finding optimal δIHV_{diff} and Λ

Disturbance vector II(52,0) can be analyzed over the last 60 steps using the following independent intervals of steps: $I_1 = [20, 32]$, $I_2 = [33, 52]$, $I_3 = [53, 60]$ and $I_4 = [67, 79]$. The steps $t = 61, \ldots, 66$ are trivial differential steps with no differences in the working state or message words with success probability 1.

First we analyze the last interval using the methods presented in Section 7.5. We optimize for the first near-collision attack in a two-block collision attack. Below we provide a speedup only for the first near-collision attack, thus the second near-collision attack complexity is the most important term in the two-block collision attack complexity. For that reason we desire that the contribution of steps $67, \ldots, 79$ to the second near-collision attack complexity is as low as possible. This can be achieved by considering only δIHV_{diff} for which there exist some message difference vector w such that the success probability $p_{(w,\delta IHV_{\text{diff}},[67,79])}$ is (almost) the highest success probability called $p_{[67,79]}$.[38]

For each possible message difference vector $w \in (\mathcal{W})_{t=67}^{79}$ we count the number N_w of δIHV_{diff} for which the success probability $p_{(w,\delta IHV_{\text{diff}},[67,79])}$ is (almost) $p_{[67,79]}$. To speed up the first near-collision attack by a factor of $N_{\max} = \max_{w \in (\mathcal{W})_{t=67}^{79}} N_w$, we use only those message difference vectors $w \in (\mathcal{W})_{t=67}^{79}$ for which $N_w = N_{\max}$. The speed up by the factor of $N_{\max}$ follows from the fact that the first near-collision attack always has $N_{\max}$ chances of finding a target δIHV_{diff}.

More precisely, we do the following:

1. We apply the analysis in Section 7.5.8 over $I_4 = [67, 79]$ to determine a set $\widehat{\mathcal{I}}$ of differences δIHV_{diff} with high weights. The set $\widehat{\mathcal{I}}$ defines the sets $\mathcal{Q}_{\text{ihv},u,t}$.

2. Using the sets $\mathcal{Q}_{\text{ihv},u,t}$ for the analysis in Section 7.5.7 we determine success probabilities $p_{(w,\delta IHV_{\text{diff}},[67,79])}$ for message difference vectors $w \in (\mathcal{W}_t)_{t=67}^{79}$ and IHV differences $\delta IHV_{\text{diff}} \in \widehat{\mathcal{I}}$.

3. Let $p_{[67,79]} = \max\{p_{(w,\delta IHV_{\text{diff}},[67,79])} \mid w \in (\mathcal{W}_t)_{t=67}^{79}, \delta IHV_{\text{diff}} \in \widehat{\mathcal{I}}\}$ be the maximum success probability of all $p_{(w,\delta IHV_{\text{diff}},[67,79])}$.

4. For each $w \in (\mathcal{W}_t)_{t=67}^{79}$, let N_w be the number of target δIHV_{diff} with high success probability (where $\alpha = 0.90$):

$$N_w = \left| \left\{ \delta IHV_{\text{diff}} \in \widehat{\mathcal{I}} \;\middle|\; p_{(w,\delta IHV_{\text{diff}},[67,79])} \geq p_{[67,79]} \cdot \alpha \right\} \right|.$$

5. Let $N_{\max} = \max_{w \in (\mathcal{W}_t)_{t=67}^{79}} N_w$ and

$$\mathfrak{W}_{[67,79]} = \left\{ w \in (\mathcal{W}_t)_{t=67}^{79} \;\middle|\; N_w = N_{\max} \right\}$$

be the set of all w for which $N_w = N_{\max}$.

38. See Section 7.5.7, p. 154.

6. Now we determine the set of target δIHV_{diff} values:

$$\widetilde{\mathcal{I}} = \left\{ \delta IHV_{\text{diff}} \in \widehat{\mathcal{I}} \;\middle|\; \exists w \in \mathfrak{W}_{[67,79]} : p_{(w,\delta IHV_{\text{diff}},[67,79]} \geq p_{[67,79]} \cdot \alpha \right\}.$$

The value α influences the obtained average success probability over steps $67, \ldots, 79$ and the obtained $N_{\max}$. A smaller value for α can increase $N_{\max}$ (thus providing a speedup for the first near-collision attack) at the cost of a lower average success probability (thus increasing the complexity of both the first and second near-collision attacks).

For our near-collision attack we have found the highest success probability over $I_4 = [67, 79]$ to be $p_{[67,79]} \approx 2^{-19.2}$. This proves the value of Section 7.5.8 as $2^{-20.4}$ approximates the highest success probability over $I_4 = [67, 79]$ allowing only working state differences as prescribed by the disturbances in the disturbance vector. Thus Section 7.5.8 provided a speed up of a factor of about $2^{1.2}$. We have found a set of message difference vectors such that $N_{\max} = 6$ and $\widetilde{\mathcal{I}}$ presented in Table 7-5 consists of 192 values for δIHV_{diff}. This implies that the success probability over I_4 of our first near-collision attack is $6 \cdot p_{[67,79]} \approx 2^{-16.6}$.

We analyze the first interval to determine the set of optimal values for Λ with (almost) the highest success probability. After constructing differential paths using this set, we are limited to one specific Λ-value and only then we can determine

$$\begin{aligned}
\mathcal{I}_0 = \big\{ & (2^{11} + 2^4 - 2^2, 2^6, 2^{31}, 2^1, 2^{31}), \\
& (2^{12} + 2^3 + 2^1, 2^7, 0, 2^1, 2^{31}), \\
& (2^{12} + 2^4 - 2^1, 2^7, 0, 2^1, 2^{31}), \\
& (2^{11} + 2^9 + 2^4 - 2^2, 2^6 + 2^4, 2^{31}, 2^1, 2^{31}), \\
& (2^{12} + 2^9 + 2^3 + 2^1, 2^7 + 2^4, 0, 2^1, 2^{31}), \\
& (2^{12} + 2^9 + 2^4 - 2^1, 2^7 + 2^4, 0, 2^1, 2^{31}) \big\}; \\
\mathcal{I}_1 = \mathcal{I}_0 \cup \big\{ & (2^{12} + 2^{11} + 2^4 - 2^2, 2^7 + 2^6, 2^{31}, 2^1, 2^{31}), \\
& (2^{12} + 2^{11} + 2^9 + 2^4 - 2^2, 2^7 + 2^6 + 2^4, 2^{31}, 2^1, 2^{31}) \big\}; \\
\mathcal{I}_2 = \big\{ & (v_1 - c \cdot 2^5, v_2, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_1,\ c \in \{0, 1\} \big\}; \\
\mathcal{I}_3 = \big\{ & (v_1 + c \cdot 2^3, v_2, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_2,\ c \in \{0, 1\} \big\}; \\
\mathcal{I}_4 = \big\{ & (v_1 - c \cdot 2^{13}, v_2 - c \cdot 2^8, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_3,\ c \in \{0, 1\} \big\}; \\
\mathcal{I}_5 = \big\{ & (v_1 - c \cdot 2^9, v_2 - c \cdot 2^4, v_3, v_4, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_4,\ c \in \{0, 1\} \big\}; \\
\widetilde{\mathcal{I}} = \big\{ & (v_1, v_2, v_3, v_4 - c \cdot 2^2, v_5) \;\big|\; (v_i)_{i=1}^5 \in \mathcal{I}_5,\ c \in \{0, 1\} \big\};
\end{aligned}$$

The resulting set $\widetilde{\mathcal{I}}$ is the set of 192 target δIHV_{diff} values. Note that some of the target δIHV_{diff} values can be constructed in several manners in the above sets, otherwise the cardinality of $\widetilde{\mathcal{I}}$ would be $(6 + 2) \cdot 2^5 = 256$. Furthermore, for any $\delta IHV_{\text{diff}} \in \widetilde{\mathcal{I}}$ also $-\delta IHV_{\text{diff}} \in \widetilde{\mathcal{I}}$.

Table 7-5: *SHA-1 near-collision attack target δIHV_{diff} values*

the corresponding optimal set of message difference vectors. Although it would be possible to use the analysis in Section 7.5.10 to possibly obtain even higher success probabilities, it diverts from the differences prescribed by local collisions which may lead to a higher number of bitconditions in the first round and thus a lower amount of freedom. Cf. our differential path in Table 7-6 (p. 169) uses local collisions over steps $10, \ldots, 19$ and has very few bitconditions over these steps, even though it has been constructed algorithmically as in Section 7.4.3. We leave it to future research to determine whether the analysis in Section 7.5.10 can provide an improvement. For now, we directly use the analysis in Section 7.5.9 and do the following:

1. We apply the analysis in Section 7.5.9 over $I_1 = [20, 32]$ to determine the set $\mathcal{J}$ of possible Λ together with the success probabilities $p_{(w,\Lambda,[20,32])}$ for $w \in (\mathcal{W}_t)_{t=20}^{32}$ and $\Lambda \in \mathcal{J}$.

2. Let $p_{[20,32]} = \max\{p_{(w,\Lambda,[20,32])} \mid w \in (\mathcal{W}_t)_{t=20}^{32}, \Lambda \in \mathcal{J}\}$ be the maximum success probability of all $p_{(w,\Lambda,[20,32])}$.

3. Let $\widetilde{\mathcal{J}} = \{\Lambda \mid \exists w \in (\mathcal{W}_t)_{t=20}^{32} : p_{(w,\Lambda,[20,32])} = p_{[20,32]}\}$ be the set of optimal values for Λ.

7.6.4 Constructing differential paths

We use our method from Section 7.4 to construct valid differential paths over the first round. The five connecting steps as in Section 7.4.4 are 3, 4, 5, 6 and 7. The forward differential path construction in Section 7.4.2 does not have a sufficient amount of freedom when lower connecting steps are chosen. For higher connecting steps, the many bitconditions around the connecting steps easily conflict with message modification techniques and the fulfillment of the message bitrelations over the last 64 steps.

In order to construct a valid first round differential path for our near-collision attack, we need sets of forward and backward partial differential paths. The forward partial differential paths are forward extensions up to step $t = 2$ of the trivial differential path defined by $\delta IHV_{\text{in}} = 0$. The backward partial differential paths are created using the set of optimal values for Λ and are then extended backwards. For each value $\Lambda = (\Delta Q_i)_{i=16}^{20}$ we directly obtain a partial differential path consisting of bitconditions $\mathfrak{q}_{16}, \ldots, \mathfrak{q}_{20}$ derived from $\Delta Q_{16}, \ldots, \Delta Q_{20}$ as in Table 6-1. These partial differential paths are extended backwards down to step $t = 8$.

Finally, we used Algorithm 7-1 and these sets of forward and backward partial differential paths to search for valid first round differential paths. The full differential path that we selected for our near-collision attack is shown in Table 7-6.

7.6.5 Second round bitconditions

Now that we have a full differential path we are bound to a specific value for Λ. Moreover, the bitconditions of this differential path may also affect steps in round two. For the purpose of early stopping techniques and tunnels, we also desire sufficient

Table 7-6: *SHA-1 near-collision differential path - round 1*

t	Bitconditions: $\mathtt{q}_t[31]\ldots\mathtt{q}_t[0]$	ΔW_t
$-4,-3,-2$	`........ ........ ........ ........`	
-1	`...1.... ........ ........ ....0...`	
0	`.^.0.1.. .....0.1 ...00.10 .1..1..1`	$\{1, 26, 27\}$
1	`.0.+^-^^ ^^^^^1^0 ^^^11^10 .0..1.+0`	$\{\overline{4}, \overline{30}, 31\}$
2	`1-...+-- -------- -------- --.-1.+0`	$\{\overline{2}, 3, \overline{4}, \overline{26}, 28, 29, 31\}$
3	`.-.-.0.1 11111111 11110++1 +-1-00-0`	$\{\overline{2}, 26, 27, 28, \overline{29}\}$
4	`.-...1.0 11111111 1111-+++ ++0.1.+1`	$\{1, \overline{3}, 4, 26, \overline{27}, \overline{28}, \overline{29}, 31\}$
5	`.-...0.. ........ ......0. .+.+10+0`	$\{\overline{4}, \overline{29}\}$
6	`.-.+.... ........ ......01 100-.0+.`	$\{2, 3, 4, 26, \overline{29}\}$
7	`-1...1.. ........ ........ ...0.0..`	$\{2, 4, \overline{26}, \overline{27}, 29, 30, 31\}$
8	`1.1-.1.. ........ ........ .....1..`	$\{\overline{1}, 26, \overline{27}\}$
9	`..-..0.. ........ ........ ........`	$\{\overline{4}, 30, 31\}$
10	`^...00.. ........ ........ .......1`	$\{\overline{2}, \overline{3}, 4, \overline{26}, \overline{28}, 29, 31\}$
11	`..-.1... ........ ........ .......0`	$\{2, \overline{26}, 27, \overline{29}\}$
12	`0-..1... ........ ........ ......!.`	$\{\overline{3}, 4, \overline{26}, \overline{27}, \overline{28}, 29, 31\}$
13	`+..01... ........ ........ ........`	$\{\overline{4}, 28, 29, 31\}$
14	`..-1.... ........ ........ ......!.`	$\{\overline{2}, 3\}$
15	`+.0.1... ........ ........ ......!^`	$\{\overline{4}, \overline{27}, \overline{28}, \overline{29}, 31\}$
16	`+-0.0... ........ ........ ......!.`	$\{3, \overline{4}, \overline{27}\}$
17	`+..1.... ........ ........ ......^.`	$\{\overline{4}, \overline{27}, \overline{28}, \overline{29}, 30\}$
18	`-.+0.... ........ ........ ........`	$\{\overline{2}, 4, \overline{27}\}$
19	`-....... ........ ........ ........`	$\{4, 28, 29, \overline{30}\}$
20	`..+..... ........ ........ ........`	

Note that we use the compact notation for the BSDRs ΔW_t (see Section 2.1.3) and the bitconditions from Table B-1.

bitconditions up to the last working state variable Q_i that may be corrected by tunnels. In our case we desire sufficient bitconditions up to Q_{25}.

For this purpose we define extra bitconditions in Table 7-7 for the second round boolean function of SHA-1.

Table 7-7: *Round two bitconditions for SHA-1.*

$\mathtt{q_t[i]}$	**condition on** $(\mathbf{Q_t[i]}, \mathbf{Q'_t[i]})$	**direct/indirect**	**direction**
`r`	$Q_t[i] = Q'_t[i] = RL(Q_{t-1}, 30)[i]$	indirect	backward
`u`	$Q_t[i] = Q'_t[i] = RR(Q_{t+1}, 30)[i]$	indirect	forward
`R`	$Q_t[i] = Q'_t[i] = \overline{RL(Q_{t-1}, 30)[i]}$	indirect	backward
`U`	$Q_t[i] = Q'_t[i] = \overline{RR(Q_{t+1}, 30)[i]}$	indirect	forward
`s`	$Q_t[i] = Q'_t[i] = RL(Q_{t-2}, 30)[i]$	indirect	backward
`c`	$Q_t[i] = Q'_t[i] = RR(Q_{t+2}, 30)[i]$	indirect	forward
`S`	$Q_t[i] = Q'_t[i] = \overline{RL(Q_{t-2}, 30)[i]}$	indirect	backward
`C`	$Q_t[i] = Q'_t[i] = \overline{RR(Q_{t+2}, 30)[i]}$	indirect	forward

To obtain the desired sufficient bitconditions for our near-collision attack we first

implemented our near-collision attack without second round bitconditions and tunnels that can correct Q_{19} and up. We used this preliminary attack to find and store many message block pairs that follow our first round differential path and the disturbance vector up to step 32, that is, all message block pairs that result in $\delta Q_{29} = \delta Q_{30} = \delta Q_{31} = \delta Q_{32} = \delta Q_{33} = 0$. For each message block pair we can derive the differential path it follows up to step 32 and thereby bitconditions up to Q_{33}. We are interested in the set of bitconditions up to Q_{25} that occurred most frequently over all found message block pairs found. Our resulting set of bitconditions $\mathfrak{q}_{19}, \ldots, \mathfrak{q}_{25}$ is shown in Table 7-8.

Table 7-8: *SHA-1 near-collision differential path - round two bitconditions*

t	Bitconditions: $\mathfrak{q}_t[31] \ldots \mathfrak{q}_t[0]$	ΔW_t
19	`-...s... ........ ........ ........`	
20	`..+.r... ........ ........ ........`	$\{2, 3, \overline{4}, \overline{27}, 28, \overline{29}, 31\}$
21	`^.r.s... ........ ........ ........`	$\{\overline{27}, 29, 30, 31\}$
22	`..+.r... ........ ........ ........`	$\{\overline{2}, 28, 29, 31\}$
23	`-...s... ........ ........ .......!`	$\{4, 27, 28, \overline{30}\}$
24	`.-+.R... ........ ........ ........`	$\{\overline{2}, 3, 28, \overline{29}, 31\}$
25	`..rSS... ........ ........ ........`	

Note that we use the compact notation for the BSDRs ΔW_t (see Section 2.1.3) and the bitconditions from Table B-1.

7.6.6 Finding optimal message bitrelations

Finding the optimal message bitrelations is split over five intervals: $I_0 = [0, 19]$, I_1, I_2, I_3 and I_4. For the first round we define $\mathfrak{W}_{[0,19]} = \{(\sigma(\Delta\widehat{W}_i))_{i=0}^{19}\}$ where as before $\Delta\widehat{W}_i$ is the message difference in step i from the differential paths in Table 7-6 and Table 7-8. For the last interval we already have found the optimal set $\mathfrak{W}_{[67,79]}$.

For the interval $I_2 = [33, 52]$ we simply apply the analysis of Section 7.5.5 resulting in the sets $\mathcal{A}_{33,52} = \{(\mathcal{P}, \mathcal{S})\}$ where $\mathcal{P}$ is trivial and $\mathcal{S}$ consists of pairs $(w, p_{w,[33,52]})$. Let $p_{[33,52]} = \max_{(w,p)\in\mathcal{S}} p$ be the maximum over all such $p_{w,[33,52]}$. Now we have found the optimal set of message difference vectors over the interval I_2:

$$\mathfrak{W}_{[33,52]} = \{w \mid (w, p) \in \mathcal{S},\ p = p_{[33,52]}\}.$$

The interval $I_3 = \{53, \ldots, 60\}$ is treated analogously to I_2 resulting in the set $\mathfrak{W}_{[53,60]}$.

To determine the optimal set of message difference vectors over the interval $I_1 = [20, 32]$ we apply the analysis of Section 7.5.9 with the following restrictions due to the bitconditions and message differences found in Tables 7-6 and 7-8:

- Let $\mathcal{Q}_{\mathrm{bc},i} = \{\Delta\widehat{Q}_i\}$ for $i \in \{-4, \ldots, 25\}$ where $\Delta\widehat{Q}_i$ follows from the differential bitconditions in $\mathfrak{q}_i$. Use $\mathcal{Q}_{\mathrm{bc},i}$ for the sets $\mathcal{Q}_i$ for $i \in \{-4, \ldots, 25\}$ in Section 7.5.9. For $i \in \{26, 33\}$, use the sets $\mathcal{Q}_{\mathrm{c},u,i}$ for $\mathcal{Q}_i$ for some value of u. In our case $u = 7$ suffices.

- Let $\Delta\widehat{F}_i$ be the boolean function differences that are implied by the bitconditions $\mathfrak{q}_{-4}, \ldots, \mathfrak{q}_{25}$ for $i \in \{0, \ldots, 26\}$. We add the restriction $\Delta F_t = \Delta\widehat{F}_t$ in step 8 of Algorithm 7-3 for $t \in \{20, \ldots, 26\}$.
- For $i \in \{0, \ldots, 24\}$, we restrict $\mathcal{W}_i$ to the single value $\{\sigma(\Delta\widehat{W}_i)\}$ where $\Delta\widehat{W}_i$ is the message difference from the differential paths in Table 7-6 and Table 7-8.

Let $\widehat{\mathcal{A}}_{20,32}$ be the resulting set of Algorithm 7-3 under these three added restrictions. For each $w \in (\mathcal{W}_t)_{t=20}^{32}$ we define its success probability under the restriction of our bitconditions as follows:

$$p_{\mathrm{bc},w,[20,32]} = \sum_{(\mathcal{P},\mathcal{S}) \in \widehat{\mathcal{A}}_{20,32}} \sum_{\substack{(w',p) \in \mathcal{S} \\ w = w'}} p.$$

This naturally leads to the maximum probability $p_{[20,32]} = \max_w p_{\mathrm{bc},w,[20,32]}$ and the optimal set of message difference vectors $\mathfrak{W}_{[20,32]}$ over interval I_1:

$$\mathfrak{W}_{[20,32]} = \left\{ w \in (\mathcal{W}_t)_{t=20}^{32} \;\middle|\; p_{\mathrm{bc},w,[20,32]} = p_{[20,32]} \right\}.$$

Now we translate each of the sets $\mathfrak{W}_{[0,19]}$, $\mathfrak{W}_{[20,32]}$, $\mathfrak{W}_{[33,52]}$, $\mathfrak{W}_{[53,60]}$ and $\mathfrak{W}_{[67,79]}$ to message bitrelations on the message words $(W_t)_{t=0}^{79}$ associated with M of the message block pair (M, M'). These message bitrelations are of the form

$$\sum_{t=0}^{79} \sum_{b=0}^{31} a_{i,t,b} \cdot W_t[b] = c_i \bmod 2$$

where $a_{i,t,b}, c_i \in \{0,1\}$. Together they can also be written as a matrix equation $A \cdot x = c$ over $\mathbb{F}_2$ where $x \in \mathbb{F}_2^{32 \cdot 80}$ represents the bits $W_t[b]$ for $t \in \{0, \ldots, 79\}$ and $b \in \{0, \ldots, 31\}$.

For $(t_b, t_e) \in \{(0,19), (20,32), (33,52), (53,60), (67,79)\}$, we do the following. For each $\widehat{w} = (\delta\widehat{W}_i)_{i=t_b}^{t_e} \in \mathfrak{W}_{[t_b,t_e]}$ we define the set $\mathcal{V}_{\widehat{w}}$ as the set of all $(W_i)_{i=0}^{79} \in \mathbb{Z}_{2^{32}}^{80}$ that are compatible with $\widehat{w}$:

$$(W_i \oplus DW_i) - W_i = \delta\widehat{W}_i, \quad \text{for } i \in \{t_b, \ldots, t_e\}.^{39}$$

Let the set $\mathcal{V} = \bigcup_{w \in \mathfrak{W}_{[t_b,t_e]}} \mathcal{V}_w$ consist of all $(W_t)_{t=0}^{79} \in \mathbb{Z}_{2^{32}}^{80}$ that are compatible with some $w \in \mathfrak{W}_{[t_b,t_e]}$.

Choose natural mappings from $\mathbb{Z}_{2^{32}}^{16}$ to $\mathbb{F}_2^{32 \cdot 80}$ and from $\mathbb{Z}_{2^{32}}^{80}$ to $\mathbb{F}_2^{32 \cdot 80}$. Let $\mathcal{V}'$ be the set consisting of all elements of $\mathcal{V}$ mapped to $\mathbb{F}_2^{32 \cdot 80}$. We search for an affine

39. Note that for $t \in \{0, \ldots, 79\}$ and $b \in \{0, \ldots, 31\}$ if $t \notin \{t_b, \ldots, t_e\}$ or $DW_t[b] = 0$ or $b = 31$ then the bit $W_t[b]$ is a free bit in $\mathcal{V}_{\widehat{w}}$, i.e., for $(W_i)_{i=0}^{79} \in \mathcal{V}$ also $(\widetilde{W}_i)_{i=0}^{79} \in \mathcal{V}$ where $\widetilde{W}_t = W_t \oplus 2^b$ and $\widetilde{W}_i = W_i$ for $i \neq t$. This implies that in practice we only need to consider those bits $W_t[b]$ for which $t \in \{t_b, \ldots, t_e\}$ and $b \neq 31$ and $DW_t[b] = 1$.

subspace $y + \mathcal{U} \subseteq \mathcal{V}'$ which is as large as possible. For our near-collision attack this affine subspace $y + \mathcal{U}$ is simply found by random trials where a random $y \in \mathcal{V}'$ is selected and beginning with $\mathcal{U} = \emptyset$. For randomly selected v and w from $\mathcal{V}'$ for which $v - w \notin \mathcal{U}$, we add $v - w$ to the span of $\mathcal{U}$ if and only if $y + \mathrm{span}(\mathcal{U}, v - w) \subseteq \mathcal{V}'$. This is repeated until we can no longer add elements to $\mathcal{U}$ in this manner. If $|\mathcal{U}| \geq |\mathcal{V}'|/2$ for the resulting y and $\mathcal{U}$ then this affine subspace is optimal. Otherwise, we repeat this construction several times in the hope of finding a larger affine subspace.

Having found an affine subspace $y + \mathcal{U} \subseteq \mathcal{V}'$, we determine the orthogonal complement $\mathcal{U}^\perp$ of the subspace $\mathcal{U}$. Choose any basis of $\mathcal{U}^\perp$ of size k and let the k rows of the matrix $A_{[t_b,t_e]} \in \mathbb{F}_2^{k \times (32 \cdot 80)}$ consist of the k basis vectors of $\mathcal{U}^\perp$. It follows that $x \in \mathcal{U} \Leftrightarrow A_{[t_b,t_e]} \cdot x = 0$ and thus

$$x \in y + \mathcal{U} \quad \Leftrightarrow \quad A_{[t_b,t_e]} \cdot x = A_{[t_b,t_e]} \cdot y.$$

Hence, the matrix equation $A_{[t_b,t_e]} \cdot x = c_{[t_b,t_e]}$ where $c_{[t_b,t_e]} = A_{[t_b,t_e]} \cdot y$ describes the desired sufficient message bitrelations over the interval $[t_b, t_e]$.

We present the message bitrelations of our near-collision attack for the last three rounds in Table 7-9. The message bitrelations for the first round can directly be read from Table 7-6: if $b \neq 31$ then $\Delta W_t[b] = -1$ and $\Delta W_t[b] = +1$ imply the message bitrelations $W_t[b] = 1$ and $W_t[b] = 0$, respectively. As $\Delta W_t[31] = -1$ and $\Delta W_t[31] = +1$ both result in $\delta m_t = 2^{31}$, no message bitrelations on bit position 31 are necessary.

The matrix equations found as above for $\mathfrak{W}_{[0,19]}$, $\mathfrak{W}_{[20,32]}$, $\mathfrak{W}_{[33,52]}$, $\mathfrak{W}_{[53,60]}$ and $\mathfrak{W}_{[67,79]}$ can be combined into a single matrix equation $A_{[0,79]} \cdot x = c_{[0,79]}$ that defines our message search space. It remains to reduce this matrix equation over the $32 \cdot 80$ message words bits to a matrix equation over the 512 message block bits using the message expansion relation. Let the message expansion be described by the matrix ME such that $ME \cdot m = w$ where $w \in \mathbb{F}_2^{80 \cdot 32}$ is the expanded message generated by $m \in \mathbb{F}_2^{16 \cdot 32}$ under the chosen natural mappings from $\mathbb{Z}_{2^{32}}^{16}$ to $\mathbb{F}_2^{32 \cdot 16}$ and from $\mathbb{Z}_{2^{32}}^{80}$ to $\mathbb{F}_2^{32 \cdot 80}$. Then the message bitrelations over the 512 message block bits is described by the matrix equation:

$$(A_{[0,79]} \cdot ME) \cdot x = c_{[0,79]}, \quad x \in \mathbb{F}_2^{32 \cdot 16}.$$

We use Gaussian elimination to obtain message bitrelations such that bitrelations on $m_t[b]$ are expressed in terms of the bits of $m_0, \ldots, m_{t-1}$ and $m_t[i]$ for $i < b$.

It may happen that these message bitrelations conflict and that there are no solutions to the above matrix equation. Since the first round has almost as many message bitrelations as the other three rounds together, one can try to use a different first round differential path with other message differences in Section 7.6.4.

Table 7-9: *SHA-1 near-collision rounds 2-4 message expansion conditions*

$W_{20}[2] = 0$	$W_{20}[3] = 0$	$W_{20}[4] = 1$	
$W_{20}[27] = 1$	$W_{20}[28] = 0$	$W_{20}[29] = 1$	
$W_{21}[27] = 1$	$W_{21}[29] = 0$	$W_{21}[30] = 0$	
$W_{22}[2] = 1$	$W_{22}[28] = 0$	$W_{22}[29] = 0$	
$W_{23}[4] = 0$	$W_{23}[27] = 0$	$W_{23}[28] = 0$	$W_{23}[30] = 1$
$W_{24}[2] = 1$	$W_{24}[3] = 0$	$W_{24}[28] = 0$	$W_{24}[29] = 1$
$W_{25}[27] = 0$	$W_{25}[30] = 1$		
$W_{26}[28] = 1$	$W_{26}[29] = 0$		
$W_{27}[4] + W_{29}[29] = 1$	$W_{27}[27] + W_{27}[28] = 1$		$W_{27}[29] = 0$
$W_{28}[4] + W_{32}[29] = 0$	$W_{28}[27] = 1$		$W_{28}[28] = 0$
$W_{36}[4] + W_{44}[29] = 0$	$W_{38}[4] + W_{44}[29] = 1$	$W_{39}[30] + W_{44}[29] = 0$	
$W_{40}[3] + W_{44}[29] = 1$	$W_{40}[4] + W_{44}[29] = 0$	$W_{41}[29] + W_{41}[30] = 0$	
$W_{42}[28] + W_{44}[29] = 1$	$W_{43}[4] + W_{47}[29] = 0$	$W_{43}[28] + W_{44}[29] = 1$	
$W_{43}[29] + W_{44}[29] = 0$	$W_{44}[28] + W_{44}[29] = 1$	$W_{45}[29] + W_{47}[29] = 0$	
$W_{46}[29] + W_{47}[29] = 0$	$W_{48}[4] + W_{52}[29] = 0$	$W_{50}[29] + W_{52}[29] = 0$	
$W_{51}[29] + W_{52}[29] = 0$			
$W_{54}[4] + W_{60}[29] = 1$	$W_{56}[4] + W_{60}[29] = 0$	$W_{56}[29] + W_{60}[29] = 1$	
$W_{57}[29] + W_{60}[29] = 1$	$W_{59}[29] + W_{60}[29] = 0$		
$W_{67}[0] + W_{72}[30] = 1$	$W_{68}[5] + W_{72}[30] = 0$	$W_{70}[1] + W_{71}[6] = 1$	
$W_{71}[0] + W_{76}[30] = 1$	$W_{72}[5] + W_{76}[30] = 0$	$W_{73}[2] + W_{78}[0] = 1$	
$W_{74}[1] + W_{75}[6] = 1$	$W_{74}[7] + W_{78}[0] = 0$	$W_{75}[1] + W_{76}[6] = 1$	
$W_{76}[0] + W_{76}[1] = 1$	$W_{76}[3] + W_{77}[8] = 1$	$W_{77}[1] + W_{77}[2] = 1$	

7.6.7 Basic collision search

The first step is to find an identical-prefix block such that IHV bitconditions $\mathfrak{q}_{-4}, \ldots, \mathfrak{q}_0$ for our near-collision attack are satisfied. This is done by simply trying random blocks until one is found that satisfies these bitconditions. Since there are 14 such bitconditions, this step has average complexity 2^{14} SHA-1 compressions.

Our near-collision algorithm can roughly be divided into three parts. The first part searches for message blocks that fulfill all bitconditions up to $\mathfrak{q}_{16}$ and all message bitrelations. The second part exploits message modification techniques, in our case tunnels, to find message blocks that fulfill all bitconditions up to $\mathfrak{q}_{25}$. The third part simply applies the message block difference, computes IHV_{out} and IHV'_{out} and checks whether the resulting δIHV_{out} is one of the target δIHV_{diff} values. The first part is discussed below, the second part is discussed in Section 7.6.8 and the third part needs no further explanation.

The first part consists of 16 steps $t = 0, \ldots, 15$. The working state $Q_{-4} = Q'_{-4}, \ldots, Q_0 = Q'_0$ is initialized using the IHV_{in} resulting from the identical-prefix block. Each step $t = 0, \ldots, 15$ does the following given values $Q_{-4}, \ldots, Q_t$ and $m_0, \ldots, m_{t-1}$:

1. Let R be the set of message bitrelations that use multiple bits of m_t and let $B \subseteq \{0, \ldots, 31\}$ be the set of bit positions b such that $m_t[b]$ is used in some message bitrelation in R. If $R = \emptyset$ then continue at step 2. Otherwise, for each of the possible values $(\widehat{m}_t[b])_{b \in B}$ that satisfy all message bitrelations in R we perform steps 2 through 6.

2. The message bitrelations imply target bit values for m_t. Let $m_{\mathrm{mask}',t}$ and $m_{\mathrm{val}',t}$ be words such that:

$$(m_t \oplus m_{\mathrm{val}',t}) \wedge m_{\mathrm{mask}',t} = 0 \quad \Leftrightarrow \quad m_t \text{ satisfies all bitrelations not in } R.$$

   We define $m_{\mathrm{mask},t}$ and $m_{\mathrm{val},t}$ using $m_{\mathrm{mask}',t}$, $m_{\mathrm{val}',t}$, B and $(\widehat{m}_t[b])_{b \in B}$:

$$m_{\mathrm{mask},t}[b] = \begin{cases} m_{\mathrm{mask}',t}[b] & \text{if } b \notin B; \\ 1 & \text{if } b \in B; \end{cases}$$

$$m_{\mathrm{val},t}[b] = \begin{cases} m_{\mathrm{val}',t}[b] & \text{if } b \notin B; \\ \widehat{m}_t[b] & \text{if } b \in B. \end{cases}$$

   It follows that $(m_t \oplus m_{\mathrm{val},t}) \wedge m_{\mathrm{mask},t} = 0$ implies that m_t satisfies all bitrelations.

3. The bitconditions $\mathfrak{q}_{t+1}$ using the given values Q_t and Q_{t-1} imply target bit values for Q_{t+1}. Let $Q_{\mathrm{mask},t+1}$ and $Q_{\mathrm{val},t+1}$ be words such that

$$(Q_{t+1} \oplus Q_{\mathrm{val},t+1}) \wedge Q_{\mathrm{mask},t+1} = 0 \quad \Leftrightarrow \quad Q_{t+1} \text{ satisfies } \mathfrak{q}_{t+1}.$$

4. Let $C = m_{\mathrm{mask},t} \oplus Q_{\mathrm{mask},t+1}$ be the mask whose '1'-bits describe bit positions b where either $Q_{t+1}[b]$ can be used to correct $m_t[b]$ or vice-versa.

5. If $w(Q_{\mathrm{mask},t+1} \wedge \overline{C}) > w(m_{\mathrm{mask},t} \wedge \overline{C})$ then the number of uncorrectable bits in Q_{t+1} that have to be satisfied by trial is larger than the number of such bits in m_t. We therefore iterate over correct values for Q_{t+1} and test for correct values of m_t:

   (a) Let $Q_{\mathrm{fixed},t+1} = (Q_{\mathrm{val},t+1} \wedge Q_{\mathrm{mask},t+1}) \oplus (\overline{m_{\mathrm{val},t}} \wedge C \wedge \overline{Q_{\mathrm{mask},t+1}})$ consist of the target bit values in $Q_{\mathrm{val},t+1}$ and the complemented bit values in $m_{\mathrm{val},t}$ that can be corrected.

   (b) For all Q_{t+1} such that $(Q_{t+1} \oplus Q_{\mathrm{fixed},t+1}) \wedge (C \vee Q_{\mathrm{mask},t+1}) = 0$ do the following:

      i. Compute

$$F_t = f_t(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30)),$$

$$m_t = Q_{t+1} - RL(Q_t, 5) - RL(Q_{t-4}, 30) - F_t - AC_t.$$

      ii. If $(m_t \oplus m_{\mathrm{val},t}) \wedge m_{\mathrm{mask},t} \wedge \overline{C} \neq 0$ then m_t does not satisfy the message bitrelations on some bit b which is uncorrectable ($C[b] = 0$). We continue at (b).

iii. We correct bits in m_t if necessary. Let $Z = (m_t \oplus m_{\mathrm{val},t}) \wedge C$ be the mask of bits that need correction.
iv. Set $\widehat{Q}_{t+1} = Q_{t+1} \oplus Z$ and $\widehat{m}_t = m_t \oplus Z$.
v. Proceed to the next step $t+1$ using the corrected values $\widehat{Q}_{t+1}$ and $\widehat{m}_t$ and continue at (b) afterwards.

6. Otherwise $w(Q_{\mathrm{mask},t+1} \wedge \overline{C}) \leq w(m_{\mathrm{mask},t} \wedge \overline{C})$ and we iterate over correct values for m_t and test for correct values of Q_{t+1}:
   (a) Let $m_{\mathrm{fixed},t} = (m_{\mathrm{val},t} \wedge m_{\mathrm{mask},t}) \oplus (\overline{Q_{\mathrm{val},t+1}} \wedge C \wedge \overline{m_{\mathrm{mask},t}})$ consist of the target bit values in $m_{\mathrm{val},t}$ and the complemented bit values in $Q_{\mathrm{val},t+1}$ that can be corrected.
   (b) For all m_t such that $(m_t \oplus m_{\mathrm{fixed},t}) \wedge (C \vee m_{\mathrm{mask},t}) = 0$ do the following:
      i. Compute
$$F_t = f_t(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30)),$$
$$Q_{t+1} = RL(Q_t, 5) + RL(Q_{t-4}, 30) + F_t + AC_t + m_t.$$
      ii. If $(Q_{t+1} \oplus Q_{\mathrm{val},t+1}) \wedge Q_{\mathrm{mask},t+1} \wedge \overline{C} \neq 0$ then Q_{t+1} does not satisfy the bitconditions $\mathfrak{q}_{t+1}$ on some bit b which is uncorrectable ($C[b] = 0$). We continue at (b).
      iii. We correct bits in Q_{t+1} if necessary. Let
$$Z = (Q_{t+1} \oplus Q_{\mathrm{val},t+1}) \wedge C$$
      be the mask of bits that need correction.
      iv. Set $\widehat{Q}_{t+1} = Q_{t+1} \oplus Z$ and $\widehat{m}_t = m_t \oplus Z$.
      v. Proceed to the next step $t+1$ using the corrected values $\widehat{Q}_{t+1}$ and $\widehat{m}_t$ and continue at (b) afterwards.

For now the last step $t = 16$ sets $m'_i = m_i \oplus DW_i$ for $i \in \{0, \ldots, 15\}$ and tests whether $\delta IHV_{\mathrm{out}}$ given by

$$\delta IHV_{\mathrm{out}} = \mathrm{SHA1Compress}(IHV_{\mathrm{in}}, (m'_i)_{i=0}^{15}) - \mathrm{SHA1Compress}(IHV_{\mathrm{in}}, (m_i)_{i=0}^{15})$$

matches one of the target $\delta IHV_{\mathrm{diff}}$ values.

The correctness of the corrections made using C and Z above is shown here for the case handled in step 4. The case handled in step 5 works analogously, where the roles of m_t and Q_{t+1} are interchanged. For some $b \in \{0, \ldots, 31\}$, let $Q_{t+1}[b]$ be an otherwise free bit and let $m_t[b]$ be restricted: $Q_{\mathrm{mask},t+1}[b] = 0$ and $m_{\mathrm{mask},t+1}[b] = 1$. Using C in step 4(b), the value of $Q_{t+1}[b]$ is fixed to the value $Q_{\mathrm{fixed},t+1}[b] = \overline{m_{\mathrm{val},t}}[b]$.

Suppose $m_t[b] = 0$ does not satisfy $m_{\mathrm{val},t}[b] = 1$. As $Q_{t+1}[b] = \overline{m_{\mathrm{val},t}}[b] = 0$, this can easily be corrected without affecting other bits by adding 2^b to m_t and Q_{t+1}. This correction maintains the equation $m_t = Q_{t+1} - RL(Q_t, 5) - RL(Q_{t-4}, 30) - F_t - AC_t$ implied by the SHA-1 step function. Suppose $m_t[b] = 1$ does not satisfy $m_{\mathrm{val},t}[b] = 0$. Then similarly, since $Q_{t+1}[b] = 1$, we can correct this without affecting other bits by subtracting 2^b from m_t and Q_{t+1}. Note that both cases only flip bit b and no other bits, exactly what is done using Z in 4(b)iii and 4(b)iv.

Table 7-10: *SHA-1 near-collision tunnels*

r	Tunnels
17	$(Q_1[7], Q_{15}[12], Q_{16}[17])$
18	$(Q_1[7], Q_{13}[10]), (Q_{14}[7]), (Q_{14}[8]), (Q_{14}[9]),$ $(Q_{15}[9]), (Q_{15}[10]), (Q_{15}[12])$
19	$(Q_{15}[5]), (Q_{15}[6]), (Q_{15}[7]), (Q_{15}[8])$
21	$(Q_{10}[6])$
22	$(Q_7[7], Q_{15[6]})$
23	$(Q_7[6]), (Q_7[8])$

These are the tunnels that are used in our near-collision attack at step r. Note that each tuple describes a tunnel through the working state bits that are changed. In principle a working state bit $Q_i[j]$ is always changed as $\Delta Q_i[j] = +1$, except if that working state bit has been changed before by a tunnel (e.g., $Q_1[7]$ in step 18). Another exception is $(Q_{12}[6])$ for which $\Delta Q_{12}[6] = -1$ is used so that that tunnel's message conditions could be combined with those of other tunnels. The additional bitconditions and message conditions used for these tunnels are presented in Table 7-11 and Table 7-12, respectively.

7.6.8 Tunnels

Similar to the tunnels for MD5, we use tunnels to speed up our near-collision search by modifying a message block pair (M, M') that fulfills all bitconditions up to some $\mathfrak{q}_k$ and all message bitrelations of Section 7.6.6. This modification from (M, M') to $(\widehat{M}, \widehat{M}')$ is such that $(\widehat{M}, \widehat{M}')$ also fulfills all bitconditions up to $\mathfrak{q}_k$ and all message bitrelations of Section 7.6.6. In this section we use differences not between M and M', but rather between M and $\widehat{M}$. For only this section we denote $\widehat{X} - X$ and $(\widehat{X}[b] - X[b])_{b=0}^{31}$ by δX and ΔX, respectively, where X and $\widehat{X}$ are associated variables in the computation of SHA1Compress of M and $\widehat{M}$, respectively.

A tunnel consists of a controlled change in the first 16 steps which results in a change somewhere in the next 16 steps. The controlled change is in fact a local collision using message differences

$$\delta m_t = 2^b, \quad \delta m_{t+1} = -2^{b+5 \bmod 32},$$

$$\delta m_{t+2} = \delta m_{t+3} = \delta m_{t+4} = 0 \quad \text{and} \quad \delta m_{t+5} = -2^{b+30 \bmod 32}.$$

We allow no carries in the working state difference. The tunnel requires therefore the following bitconditions:

$$Q_{t+1}[b] = Q'_{t+1}[b] = 0, \quad Q_{t-1}[b + 2 \bmod 32] = Q_{t-2}[b + 2 \bmod 32],$$

$$Q_{t+2}[b - 2 \bmod 32] = 0 \quad \text{and} \quad Q_{t+3}[b - 2 \bmod 32] = 1.$$

Evidently, a tunnel must be compatible with our near-collision attack so far. This implies that the bitconditions required for a tunnel must be compatible with the bitconditions of the near-collision attack *with respect to the first message block*. For instance, the bitcondition $Q_t[b] = 0$ is compatible with $\mathfrak{q}_t[b] =$'.', $\mathfrak{q}_t[b] =$'0' and counter-intuitively also with $\mathfrak{q}_t[b] =$'+'.

Furthermore, the message differences may not break the message bitrelations of our near-collision attack. We search for values Δm_t, Δm_{t+1} and Δm_{t+5} such that $\sigma(\Delta m_i) = \delta m_i$ for $i \in \{t, t+1, t+5\}$ and for all message bitrelations

$$\sum_{i=0}^{15} \sum_{j=0}^{31} c_{i,j} \cdot m_i[j] = a \bmod 2$$

we have that these message differences do not break the message bitrelation:

$$\sum_{i \in \{t,t+1,t+5\}} \sum_{j=0}^{31} c_{i,j} \cdot \Delta m_i[j] = 0 \bmod 2.$$

Most of the time there is only a single value for each of these Δm_i that is interesting for practical use[40], which directly implies additional message bitrelations: $\Delta m_i[j] = +1$ implies $m_i[j] = 0$ and $\Delta m_i[j] = -1$ implies $m_i[j] = 1$, except for $\Delta m_i[31] \neq 0$ for which no additional message bitrelations are necessary. If for any of the Δm_i there are multiple interesting values then we do not use message bitrelations. Instead, we test at step i whether adding δm_i to m_i results in one of the interesting Δm_i.

Tunnel message bitrelations are only a precondition when applying the tunnel, after which they do not have to be fulfilled anymore. Thus a tunnel may not break the original message bitrelations from Section 7.6.6 but also may not break any message bitrelations from other tunnels that are used later on in our near-collision search algorithm.

Since the tunnel's message bitdifferences in the first 16 steps have been decided on, we can determine the possible message bitdifferences in the next steps $t = 16, \ldots$. Since Δm_t, Δm_{t+1} and Δm_{t+5} are not always uniquely determined, neither are the possible message bitdifferences in steps $16, \ldots$. An important aspect of a tunnel is the first step $s > 16$ for which the possible message differences δm_s are non-zero. Instead of using a tunnel at step s, i.e., after the verification whether $\mathfrak{q}_s$ is satisfied, we use a tunnel at the highest step $r \geq s$ such that with almost certainty the tunnel does not affect any bit $Q_i[j]$ with $\mathfrak{q}_i[j] \neq$ '.' for $i \in \{16, \ldots r\}$. The choice of r implies that the tunnel with probability not almost 0 affects at least one bit $Q_{r+1}[b]$ for which $\mathfrak{q}_{r+1}[b] \neq$'.'. Depending on the tunnel used there may be an exactly predictable change in Q_{r+1} or the change may be not so predictable. If there is an exactly predictable change in Q_{r+1} then as a speed up we can do a quick test whether the new Q_{r+1} satisfies $\mathfrak{q}_{r+1}$ before actually applying the tunnel and fully recomputing steps $s, \ldots, r$.

For some tunnels we use the following modifications of the description above:

40. In this case we only find a Δm_i 'interesting' if $w(\Delta m_i) = 1$ or if there is an 'interesting' $\Delta \widehat{m}_i$ such that $w(\Delta m_i) = w(\Delta \widehat{m}_i) + 1$. This choice allows a very efficient check for a given m_i whether adding δm_i to m_i results in one of the 'interesting' Δm_i: either using message bitrelations or a simple check of the form $(m_i \wedge X) \neq Y$. Naturally if this does not lead to any 'interesting' Δm_i, we use only that allowed Δm_i with the lowest weight.

Table 7-11: *SHA-1 near-collision tunnel bitconditions*

t	Bitconditions: $\mathfrak{q}_t[31] \ldots \mathfrak{q}_t[0]$
−1	...1.... ....1. ....0...
0	.^.0.1.. .....0.1 ...00.10 .1..1..1
1	.0.+^-^^ ^^^^^1^0 ^^^11^10 00..1.+0
2	1-...+-- -------- -------- --0-1.+0
3	.-.-.0.1 11111111 11110++1 +-1-00-0
4	.-...1.0 11111111 1111-+++ ++0.1.+1
5	.-...0.. ....01 1+.+10+0
6	.-.+.... ....^01 100-.0+.
7	-1...1.. ....0 0000.0..
8	1.1-.1.. .... .0000 1..
9	..-..0.. ....^^ .1111...
10	^...00.. .... 000....1
11	..-.1... ....^ ..00...0
12	0-..1... ....^.... .1110.!.
13	+..01... ....0.0 ...0..0.
14	..-1.... ....0 ...1..1.
15	+.0.1... .... .0....0^

The additional bitconditions used for the tunnels are underlined.

- Restricting the set of Δm_i for $i \in \{t, t+1, t+5\}$ to the ones with lowest weight if this leads to a higher value of r.

- Restricting the set of Δm_i for $i \in \{t, t+1, t+5\}$ to the ones with lowest weight may also have as result that the first two message differences after step 15 are of the form: $\Delta m_s[b] = \pm 1$ and $\Delta m_{s+1}[b + 5 \bmod 32] = \pm 1$ and $\Delta m_s[i] = \Delta m_{s+1}[j] = 0$ for all other bits $i \neq b$ and $j \neq b + 5 \bmod 32$. In that case, adding the message bitrelation $m_s[b] + m_{s+1}[b + 5 \bmod 32] = 1 \bmod 2$ may also lead to a higher value of r as then $m_{s+1}[b + 5 \bmod 32]$ forms a correction for the disturbance caused by $m_s[b]$.

- For $t \geq 13$, not all of the three bitconditions

$$Q_{t-1}[b + 2 \bmod 32] = Q_{t-2}[b + 2 \bmod 32],$$

$$Q_{t+2}[b - 2 \bmod 32] = 0 \quad \text{and} \quad Q_{t+3}[b - 2 \bmod 32] = 1$$

  are strictly necessary for the first 16 steps. We remove not strictly necessary bitconditions if the removal does not lead to a lower value of r.

- Even though two separate tunnels cannot be used when both break some of the message bitrelations, if they break exactly the same set of message bitrelations

Table 7-12: *SHA-1 near-collision tunnel message conditions*

$W_0 \wedge (2^9 - 2^7) \neq (2^9 - 2^7)$
$W_1[12] = 1$
$W_5 \wedge (2^{10} - 2^5) \neq 0$
$W_6[10] = 0$
$W_7[10] = W_7[11] = W_7[12] = W_7[13] = 1$
$W_9[6] = W_9[7] = 0$
$W_{10}[11] = W_{10}[12] = 1$
$W_{11}[4] = W_{11}[5] = W_{11}[6] = W_{11}[7]$
$W_{12} \wedge (2^{10} - 2^8) \neq 0, \quad W_{12}[10] = W_{12}[11] = 0$
$W_{13}[15] = 1$
$W_{14}[4] = W_{14}[5] = 1, \quad W_{14}[6] = 0$
$W_{15}[11] = 1$

Note that the expression $(2^x - 2^y)$ with $x > y$ denotes a word for which the only '1'-bits occur at bit positions $y, \ldots, x - 1$.

then they can be used simultaneously. Similarly, a set of $n > 2$ separate tunnels may be used simultaneously if each of the message bitrelations is broken by an even number of tunnels from this set.

- For $t < 14$, if a tunnel breaks only message bitrelations over m_{14} and m_{15} then that tunnel can be used to speed up steps 14 and 15 instead of higher steps. Such a tunnel can be used in our near-collision search algorithm at step 14 (or 15 if it does not break message bitrelations over m_{14}) before $m_{\text{val}',14}$ (or $m_{\text{val}',15}$) is determined. Since such tunnels were not considered at the time of implementing our near-collision attack, these tunnels may lead to an improvement of our near-collision attack.

The tunnels that we use in our near-collision are described in Table 7-10. The additional bitconditions and message conditions they required are presented in Table 7-11 and Table 7-12, respectively. For more details on how these tunnels are used in the implementation of our near-collision attack we refer to [HC]. We like to note that even with the added conditions from the tunnels, there are at least 40 bits of freedom left in the 512-bit message space (taking into account the IHV_{in} conditions, but ignoring the degrees of freedom from the identical IHV_{in}) so that many solutions should exist for any given IHV_{in}.

7.6.9 Verification of correctness and runtime complexity

Although so far we were unable to find actual near-collision blocks using our near-collision attack, we show how to verify the correctness of our implementation and its runtime complexity. The implementation of our near-collision attack can be retrieved by checking out a copy of the hashclash source repository at Google Code [HC] using

the Subversion client `svn`[41]:

```
svn checkout http://hashclash.googlecode.com/svn/trunk/ .
```

The C++ source code of our near-collision attack can be found in the following subdirectory:

```
src/diffpathcollfind_sha1/collfind.cpp .
```

After all tunnels have been exploited, we the function `step_16_33()` to compute all remaining steps up to step 32 for both messages M and M' and to verify whether $Q'_i = Q_i$ for $i \in \{29, \ldots, 33\}$. Although uncommented for performance reasons, this function can at this point call the function `check40()` for a secondary check that $Q'_i = Q_i$ for $i \in \{29, \ldots, 33\}$ and thus that our implementation works correctly up to this point. If $Q'_i = Q_i$ for $i \in \{29, \ldots, 33\}$ then `step_16_33()` increases a counter that allows us to determine the average complexity C_0 of searching for blocks M and M' that follow our disturbance vector up to step 32. Our implementation prints this counter divided by the number of seconds as `timeavg 40`. We have determined in this manner that C_0 is equivalent to about $2^{11.97}$ SHA-1 compressions on an Intel Core2 Duo Mobile T9400 operating on Windows 7. The runtime complexity is thus equal to C_0/p, where p is the probability that a message block pair (M, M') leads to one of the target δIHV_{out} values, assuming that $Q'_i = Q_i$ for $i \in \{29, \ldots, 33\}$.

To check whether the current message block pair (M, M') leads to one of the target δIHV_{out} values, the function `check_nc()` is called. There are four independent intervals $I_1 = [0, 32]$, $I_2 = [33, 52]$, $I_3 = [53, 60]$ and $I_4 = [67, 79]$. The first interval is always successful whenever `check_nc()` is called. The other remaining intervals $[t_b, t_e]$ require that $Q'_i = Q_i$ for $i \in \{t_b - 4, \ldots, t_b\}$, i.e., that the previous intervals were successful. For the second interval this condition is thus guaranteed when `check_nc()` is called. We determine the probability p as the product of the success probabilities over the last three intervals.

For each of the last three intervals we can verify their success probabilities experimentally as follows. For interval $[t_b, t_e]$ let $(W_t)_{t=0}^{79}$ and $(W'_t)_{t=0}^{79}$ be the expanded messages from M and M', respectively. Set $Q_{-4}, \ldots, Q_0$ to the IHV_{in} resulting from the identical-prefix block using Equation 7.4 (see p. 119). Compute steps $t = 0, \ldots, t_e$ for the message block M:

$$\begin{aligned} F_t &= f_t(Q_{t-1}, RL(Q_{t-2}, 30), RL(Q_{t-3}, 30)), \\ Q_{t+1} &= F_t + AC_t + W_t + RL(Q_t, 5) + RL(Q_{t-4}, 30). \end{aligned}$$

To more quickly determine the success probability, we assume that the previous intervals were successful, i.e., we set Q'_i to Q_i for $i \in \{t_b - 4, \ldots, t_b\}$. Then we compute steps $t = t_b, \ldots, t_e$ for the message block M':

$$\begin{aligned} F'_t &= f_t(Q'_{t-1}, RL(Q'_{t-2}, 30), RL(Q'_{t-3}, 30)), \\ Q'_{t+1} &= F'_t + AC_t + W'_t + RL(Q'_t, 5) + RL(Q'_{t-4}, 30). \end{aligned}$$

41. Subversion is a version control system: `http://subversion.apache.org/`

For I_2 and I_3 the success probability of these intervals can thus be computed as the probability that $Q'_i = Q_i$ for $i \in \{t_e-3, \ldots, t_e+1\}$. This probability is experimentally approximated as y/x by counting the number of times x that `check_nc()` is called and the number of times y that after the computations above $Q'_i = Q_i$ for $i \in \{t_e - 3, \ldots, t_e + 1\}$. For I_4 we do the same except instead of using $Q'_i = Q_i$ for $i \in \{t_e-3, \ldots, t_e+1\}$ as the success condition, we use the condition whether $(RL(Q'_i, r_i) - RL(Q_i, r_i))_{i=76}^{80}$ is one of the target δIHV_{diff} values where $r_i = 30$ for $i \leq 78$ and $r_i = 0$ otherwise (see also Equation 7.6, p. 119).

The experimentally approximated success probabilities of the intervals I_2, I_3 and I_4 are printed by our near-collision attack as the numbers `avg 53 stats`, `avg 61 stats` and `avg 80 stats`, respectively. The success probabilities are in this manner estimated as $\Pr[I_2] = 2^{-20.91}$, $\Pr[I_3] = 2^{-8.00}$ and $\Pr[I_4] = 2^{-16.65}$ which accurately match the theoretical maximum success probabilities[42] as determined by the differential cryptanalysis of Section 7.5. Since these success probabilities are non-zero, our implementation also works correctly over steps $t > 32$. The runtime complexity of our near-collision attack is hereby estimated in the number of SHA-1 compressions as

$$\frac{C_0}{\Pr[I_2] \cdot \Pr[I_3] \cdot \Pr[I_4]} \approx 2^{11.97} \cdot 2^{20.91} \cdot 2^{8.00} \cdot 2^{16.65} = 2^{57.53}.$$

We have found an example message pair shown in Table 7-13 that satisfies our differential path up to I_4 (thus up to step 66), such message pairs can be found with an average complexity of about $2^{11.97} \cdot 2^{20.91} \cdot 2^{8.00} = 2^{40.9}$ SHA-1 compressions.

42. Taking into account the speed-up factor $N_{\max} = 6$ for I_4.

Table 7-13: *Example message pair each consisting of an identical-prefix block and a near-collision block satisfying our differential path up to step 66.*

First message
bc 7e 39 3a 04 70 f6 84 e0 a4 84 de a5 56 87 5a
cd df f9 c8 2d 02 01 6b 86 0e e7 f9 11 e1 84 18
71 bf bf f1 06 70 95 c9 ed 44 af ee 78 12 24 09
a3 b2 eb 2e 16 c0 cf c2 06 c5 20 28 10 38 3c 2b
73 e6 e2 c8 43 7f b1 3e 4e 4d 5d b6 e3 83 e0 1d
7b ea 24 2c 2b b6 30 54 68 45 b1 43 0c 21 94 ab
fb 52 36 be 2b c9 1e 19 1d 11 bf 8f 66 5e f9 ab
9f 8f e3 6a 40 2c bf 39 d7 7c 1f b4 3c b0 08 72
Second message
bc 7e 39 3a 04 70 f6 84 e0 a4 84 de a5 56 87 5a
cd df f9 c8 2d 02 01 6b 86 0e e7 f9 11 e1 84 18
71 bf bf f1 06 70 95 c9 ed 44 af ee 78 12 24 09
a3 b2 eb 2e 16 c0 cf c2 06 c5 20 28 10 38 3c 2b
7f e6 e2 ca 83 7f b1 2e fa 4d 5d aa df 83 e0 19
c7 ea 24 36 0b b6 30 44 4c 45 b1 5f e0 21 94 bf
f7 52 36 bc eb c9 1e 09 a9 11 bf 93 4a 5e f9 af
23 8f e3 72 f0 2c bf 29 d7 7c 1f b8 84 b0 08 62

7.7 Chosen-prefix collision attack

Theorem 3.6 (p. 36) allows us to construct a chosen-prefix collision attack against SHA-1 using the near-collision attack presented in Section 7.6. Given chosen prefixes P and P', we append padding bit strings S_r and S'_r such that the bit lengths of $P||S_\mathrm{r}$ and $P'||S'_\mathrm{r}$ are both equal to $N \cdot 512 - K$, where $N, K \in \mathbb{N}^+$ and K is a later to be defined constant value. Let IHV_{N-1} and IHV'_{N-1} be the intermediate hash values after processing the first $(N-1) \cdot 512$ bits of $P||S_\mathrm{r}$ and $P'||S'_\mathrm{r}$, respectively. Furthermore, let B and B' be the last $512 - K$ bits of $P||S_\mathrm{r}$ and $P'||S'_\mathrm{r}$, respectively.

7.7.1 Birthday search

Section 6.5.2 and [vOW99] explain how to perform a birthday search. We need to choose the birthday search space V and the birthday step function $f : V \to V$. Based on the 192 target δIHV_diff-values given in Table 7-5 (p. 167), we have chosen V and f as follows:

$$V = \mathbb{Z}_{2^{13}} \times \mathbb{Z}_{2^{18}} \times \mathbb{Z}_{2^{31}} \times \mathbb{Z}_{2^{25}} \times \mathbb{Z}_{2^{32}};$$

$$f(v) = \begin{cases} \phi\big(\mathrm{SHA1Compress}(IHV_{N-1}, B||v)\big) & \text{if } \tau(v) = 0; \\ \phi\big(\mathrm{SHA1Compress}(IHV'_{N-1}, B'||v) - (0,0,0,0,2^{31})\big) & \text{if } \tau(v) = 1, \end{cases}$$

where $\phi : \mathbb{Z}_{2^{32}}^5 \to V$ and $\tau : V \to \{0,1\}$ are defined as

$$\phi(a,b,c,d,e) = \big((a[i])_{i=19}^{31}, (b[i])_{i=14}^{31}, (c[i])_{i=0}^{30}, (d[i])_{i=7}^{31}, e\big);$$
$$\tau(a,b,c,d,e) = w(a) \bmod 2.$$

These choices were made with the following considerations:

- The 192 target δIHV_diff-values are all of the form $(a, b, \mu \cdot 2^{31}, \nu \cdot 2^1, 2^{31})$, where $\mu \in \{0,1\}$, $\nu \in \{-1,1\}$, and $a, b \in \mathbb{Z}_{2^{32}}$.
- For all δIHV_diff-values (a,b,c,d,e), we have that $a \in \{-2^{13}, \ldots, 2^{13}\}$. This implies that with low probability adding a to a randomly chosen $x \in \mathbb{Z}_{2^{32}}$ affects bit position 19 and higher.
- For all δIHV_diff-values (a,b,c,d,e), we have that $b \in \{-2^8, \ldots, 2^8\}$. This implies that with low probability adding b to a randomly chosen $x \in \mathbb{Z}_{2^{32}}$ affects bit position 14 and higher.
- For all δIHV_diff-values (a,b,c,d,e), we have that $d \in \{-2^1, 2^1\}$. This implies that with low probability adding d to a randomly chosen $x \in \mathbb{Z}_{2^{32}}$ affects bit position 7 and higher.

For a birthday search collision $f(v) = f(w)$ with $\tau(v) \neq \tau(w)$, let $(x,y) = (v,w)$ if $\tau(v) = 1$ and $(x,y) = (w,v)$ otherwise. Then

$$IHV'_N = (a',b',c',d',e') = \mathrm{SHA1Compress}(IHV'_{N-1}, B'||x),$$
$$IHV_N = (a,b,c,d,e) = \mathrm{SHA1Compress}(IHV_{N-1}, B||y).$$

The resulting $\delta IHV_N = (\delta a, \delta b, \delta c, \delta d, \delta e) = IHV'_N - IHV_N$ is of the form

- $\delta a \in \left\{ l - m \;\middle|\; l, m \in \mathbb{Z}_{2^{32}}, (l[i])_{i=19}^{31} = (m[i])_{i=19}^{31} \right\}$;
- $\delta b \in \left\{ l - m \;\middle|\; l, m \in \mathbb{Z}_{2^{32}}, (l[i])_{i=14}^{31} = (m[i])_{i=14}^{31} \right\}$;
- $\delta c \in \left\{ l - m \;\middle|\; l, m \in \mathbb{Z}_{2^{32}}, (l[i])_{i=0}^{30} = (m[i])_{i=0}^{30} \right\} = \{0, 2^{31}\}$;
- $\delta d \in \left\{ l - m \;\middle|\; l, m \in \mathbb{Z}_{2^{32}}, (l[i])_{i=7}^{31} = (m[i])_{i=7}^{31} \right\}$;
- $\delta e = 2^{31}$, since $e' - 2^{31} = e$ by definition of f and $f(x) = f(y)$.

For each of the 192 target δIHV_{diff} we can determine the probability $p_{\delta IHV_{\text{diff}}}$ that $\delta IHV_N = \delta IHV_{\text{diff}}$. The sum of these 192 probabilities $p_{\delta IHV_{\text{diff}}}$ is approximately $2^{-33.46}$.

Therefore, a birthday search collision pair v, w with $f(v) = f(w)$ has a probability of $q = 2^{-33.46-1}$ that $\tau(v) \neq \tau(w)$ and δIHV_N is one of the 192 target δIHV_{diff}-values. This implies that the expected birthday search complexity in SHA-1 compressions (ignoring the cost of computing collision points) is

$$\sqrt{\frac{\pi \cdot |V|}{2 \cdot q}} \approx 2^{77.06}.$$

Storing a single trail (beginpoint, endpoint, length) costs about 36 bytes. When using $2.5 \cdot 36/q \approx 2^{40.95}$ bytes (about 2TB) then the expected complexity of generating trails equals the expected complexity of computing the collision points. The expected complexity of computing collision points can be made significantly lower by using more memory. Hence, the overall expected birthday search complexity is approximately $2^{77.1}$ SHA-1 compressions.

7.7.2 Near-collision block

Assume we have found bit strings S_{b} and S'_{b} using the above birthday search such that δIHV_N is one of the 192 δIHV_{diff}-values, where IHV_N and IHV'_N are the intermediate hash values after processing the first $512 \cdot N$ bits of $P||S_{\text{r}}||S_{\text{b}}$ and $P'||S'_{\text{r}}||S'_{\text{b}}$, respectively. The remaining step is to execute a near-collision attack identical to the second near-collision attack in a two-block identical-prefix collision attack as described in Section 7.6.1. To construct this near-collision attack we follow the steps as described in Section 7.6 with the following modifications:

- In Section 7.6.3 for I_4, we set $\widehat{\mathcal{I}}$ to $-\delta IHV_N$ in step 1. This leads to $N_{\max} = 1$ and a smaller set of optimal message difference vectors $\mathfrak{W}_{[67,79]}$.
- Instead of using the trivial differential path defined by $\delta IHV_{\text{in}} = 0$ in Section 7.6.4, we use the differential path $\mathfrak{q}_{-4}, \ldots, \mathfrak{q}_0$ consisting of bitconditions '0', '1', '-' and '+' that match the values $Q_{-4}, \ldots, Q_0$ and $Q'_{-4}, \ldots, Q'_0$ as initialized by definition from IHV_N and IHV'_N.

Executing the constructed near-collision attack results in message blocks S_c and S'_c such that

$$\text{SHA-1}\big(P||S_r||S_b||S_c\big) = \text{SHA-1}\big(P'||S'_r||S'_b||S'_c\big).$$

7.7.3 Complexity

As mentioned in Section 7.6.1, an upper bound for the second near-collision attack in a two-block identical-prefix collision attack is about $2^{65.3}$ SHA-1 compressions. This same upper bound also holds for the above near-collision attack. The near-collision attack complexity is thus a factor of $2^{11.8}$ smaller than the expected birthday search cost of $2^{77.1}$ SHA-1 compressions. Hence, the overall cost of a chosen-prefix collision attack against SHA-1 is dominated by the expected $2^{77.1}$ SHA-1 compressions required to generate the birthday search trails. This complexity is currently infeasible and this chosen-prefix collision attack against SHA-1 remains a theoretical attack.