Searching for Differential Paths in MD4 ⋆

Martin Schläffer and Elisabeth Oswald

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria
{martin.schlaeffer, elisabeth.oswald}@iaik.tugraz.at

Abstract. The ground-breaking results of Wang *et al.* have attracted a lot of attention to the collision resistance of hash functions. In their articles, Wang *et al.* give input differences, differential paths and the corresponding conditions that allow to find collisions with a high probability. However, Wang *et al.* do not explain how these paths were found. The common assumption is that they were found by hand with a great deal of intuition.
In this article, we present an algorithm that allows to find paths in an automated way. Our algorithm is successful for MD4. We have found over 1000 differential paths so far. Amongst them, there are paths that have fewer conditions in the second round than the path of Wang *et al.* for MD4. This makes them better suited for the message modification techniques that were also introduced by Wang *et al.*
Keywords: collision search, differential path, MD4

1 Introduction

The cryptanalysis of hash functions has become a hot topic within the cryptographic community over the last two years. Especially the ground breaking results of Wang *et al.* have drawn significant attention towards the security claims that were made for commonly used hash functions.

During the last two years, most hash functions have succumbed to the attacks of Wang *et al.* At first, the hash functions MD4 (as well as RIPEMD) and MD5 were analyzed by Wang *et al.* in [WLF+05] and [WY05]. Based on the techniques that have been introduced in these two papers, more advanced attacks on SHA-0 and SHA-1 have been published some time later in [WYY05b] and [WYY05a]. In all articles published by Wang *et al.* so far, only little details about the way in which the differences and the conditions were determined, have been published. Except for the article of Hawkes *et al.* [HPR04] that provides some musings on the techniques used for MD5, the PhD thesis of Magnus Daum [Dau05] and an ECRYPT deliverable [ABB+05] that both provide some high level discussions of the techniques of Wang *et al.*, we are not aware of any other article that gives insights into the techniques of Wang *et al.* In particular, there exist, to our knowledge, no insights about the techniques that Wang *et al.* used to find

⋆ The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

so-called differential paths, *i.e.* to find the specific sequence of differences over a given number of steps that produces a local collision.

It is therefore easy to motivate and to explain the aim of the research that we present in this paper. We have tried to come up with an algorithm that finds differential paths in an automated way. As target for our path-searching-algorithm we picked MD4. The reason for choosing MD4 is also easily motivated; it is the simplest of the well known hashing algorithms and it is the basis for many other algorithms such as MD5, RIPEMD, SHA-0 and SHA-1.

Our algorithm is successful: given a difference for the input message it computes differential paths for MD4 in an automated way. Among the differential paths that we have found so far, there are paths that are even slightly better than the path that Wang *et al.* reported in their original article. Our path has less conditions in the second round.

This article is organized as follows. In Sect. 2 we briefly review the attack by Wang *et al.* on MD4. In Sect. 3, we introduce the notation that we use to describe our algorithm. In Sect. 4, we explain our algorithm and in Sect. 5, we report on the results that we have obtained with it. We conclude this article in Sect. 6. There are several appendices to this article. They give more information about our algorithm and the best path that we found with it.

2 The Wang *et al.* approach

In this section we outline the approach by Wang *et al.* based on the example of the MD4 hash function. We first review the working principle of MD4 and then we focus on the attack of Wang *et al.*

2.1 The MD4 Hash Function

The MD4 algorithm hashes an input of arbitrary length to a 128-bit value. The algorithm proceeds as follows. The input message M is modified by a specific padding rule to a message with a length that is a multiple of 512. Then, the padded message is subjected to the MD4 compression function. The compression function consists of three rounds having 16 steps each. Each round uses a different Boolean function f_i: in the first round it is the *IF* function, in the second round it is the *MAJ* (majority) function and in the third round it is the *XOR* function.

In every step in MD4, a 32-bit variable r_i is updated according to the rule given in (1). Later in this article, we use the notation that the j-th bit of r_i is denoted by $r_{i,j}$. In (1), the operator $+$ denotes the addition modulo 2^{32} and the operator $\lll s_i$ denotes a circular left shift (rotation) by s_i positions. The variable m_{w_i} defines a message word and the variable k_i defines a round constant. The order of accessing the message words is given in Tab.1.

$$r_i = (r_{i\text{-}4} + f_i(r_{i\text{-}1}, r_{i\text{-}2}, r_{i\text{-}3}) + m_{w_i} + k_i) \lll s_i, \;\; 0 \leq i \leq 47. \tag{1}$$

The number of bit positions s_i in a rotation is either $\{3, 7, 11, 19\}$ in the first round, $\{3, 5, 9, 13\}$ in the second round, or it is $\{3, 9, 11, 15\}$ in the third round.

Table 1. The order of message words in MD4.

i	w_i
0...15	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
16...31	0,4,8,12,1,5,9,13,2,6,10,14,3,7,11,15
21...47	0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15

The initial values are in hexadecimal notation:

$$(r_{\text{-}4}, r_{\text{-}3}, r_{\text{-}2}, r_{\text{-}1}) = (\text{0x67452301}, \text{0x10325476}, \text{0x98badcfe}, \text{0xefcdab89})$$

These initial values are used to initialize the four 32-bit chaining variables (A, B, C, D). After 48 steps, the values $(r_{44}, r_{45}, r_{46}, r_{47})$ are added to the chaining variables (A, B, C, D). If all message blocks have been processed, then the hash value of the input message is determined by the concatenation of the four chaining variables.

2.2 Selecting an Input Difference

In the first step of the Wang *et al.* attack, one determines the difference Δ between the two input messages M and M' (*i.e.* the input difference). In contrast to Dobbertin's attack [Dob98] on MD4, Wang *et al.* do not aim on producing one local collision within MD4 but two. The idea is to have one local collision in the third round that is easily fulfilled and to have another local collision over some steps in the first two rounds. The local collision in the third round determines the input difference (see [Sch06] for further details).

Differential properties of the *XOR* function. There are two simple observations that are the foundation for producing the local collision in the third round in the Wang *et al.* attack. The first observation is that an input difference of $2^{31} \pmod{2^{32}}$ implies that only the 31st bit in the message words differ. The second observation is that if two input values of the *XOR* function have a difference of 2^{31}, then this difference is canceled.

Differential properties of the update rule in the third round. In the third round of MD4, the function f_i in the update rule (1) is the *XOR* function. We look at step i and assume hereby that there are no differences in the four previous steps. Choosing the message difference to be 2^{31-s_i} in step i, causes the difference after the i-th step to be 2^{31}. In the $(i+1)$-st step, the difference of 2^{31} propagates through the *XOR* function. In order to cancel it, we choose the difference of $m_{w_{i+1}}$ to be 2^{31} (also -2^{31} would work). Because the difference from the i-th step also goes into the *XOR* function in the $(i+2)$-nd step, it is clever to set the difference of $m_{w_{i+1}}$ to be $2^{31} + 2^{31-s_{i+1}}$. In this way we cancel the 2^{31} difference in the $(i+1)$-st step and insert a $+2^{31-s_{i+1}}$ difference that becomes a 2^{31} difference after the rotation by s_{i+1} bit positions. In the

Table 2. Propagation of differences in the third round of MD4 according to the update rule $r_i = (r_{i\text{-}4} + XOR(r_{i\text{-}3}, r_{i\text{-}2}, r_{i\text{-}1}) + m_{w_i} + k_i) \lll s_i$.

Step	$\Delta r_{i\text{-}4}$	$\Delta r_{i\text{-}3}$	$\Delta r_{i\text{-}2}$	$\Delta r_{i\text{-}1}$	Δr_i	Δm_{w_i}
i	0	0	0	0	2^{31}	2^{31-s_i}
i+1	0	0	0	2^{31}	2^{31}	$2^{31} + 2^{31-s_i}$
i+2	0	0	2^{31}	2^{31}	0	0
i+3	0	2^{31}	2^{31}	0	0	0
i+4	2^{31}	2^{31}	0	0	0	0
i+5	2^{31}	0	0	0	0	2^{31}
i+6	0	0	0	0	0	0

$(i + 2)$-nd step, we have on two inputs of the *XOR* function a difference of 2^{31}, which cancel each other. Hence, the difference after the $(i + 2)$-nd step is zero. The same argument holds for step $i + 3$. In step $i + 4$, the input r_{i+1} of the *XOR* function has difference 2^{31}, hence it propagates and gets canceled by r_i in the addition. Consequently, in the $(i + 5)$-th step, there is the 2^{31} difference in r_{i+1} value that needs to be canceled. This can be done by inserting the same difference in $m_{w_{i+5}}$. This differential behavior is summarized in Tab. 2. One can choose the starting step i for this local collision. The choice of i determines in which message words the differences are introduced. This in turn determines the length of the differential path that describes the local collision over the steps in the first two rounds. We can also choose the sign of the differences. The choice $i = 35$ leads to $\Delta m_1 = 2^{31}$, $\Delta m_2 = 2^{31} + 2^{28}$ and $\Delta m_{12} = 2^{16}$. As indicated before, we may choose other signs for the differences: the differences $\Delta m_1 = 2^{31}$, $\Delta m_2 = 2^{31} - 2^{28}$ and $\Delta m_{12} = -2^{16}$ also lead to a local collision. In our experiments, this particular choice of i turned out to be the best choice.

2.3 Finding a Differential Path

The second crucial step is to find a so-called differential path that cancels the differences between steps 1 and 24. In the articles of Wang *et al.* such paths were given. However, no insight was provided how these paths were found. It is therefore assumed that the paths were found by hand. This means, that a great deal of intuition by the researchers was needed to determine the paths. The main contribution of this article is the automated search algorithm, which is described in Sect. 4.

2.4 Message modification

The result of the second step is a differential path and the conditions on the intermediate values that are needed to fulfill the path. These conditions can be translated into equations for the message words that allow to pre-fulfill the conditions. In the third and last step of the attack, one applies different message modification techniques to the message in order to pre-fulfill as many conditions as possible.

Different message modification techniques were introduced by Wang *et al.* There is the single-step message modification technique, which allows to pre-fulfill all conditions that occur in the first round. The second technique is the multi-step message modification and allows to pre-fulfill some conditions in the second round. Other ideas for message modification techniques are the so-called advanced multi-step message modification techniques [WLF+05] and techniques that have been mentioned in [ABB+05]. The number of conditions that cannot be pre-fulfilled determines the overall complexity of the attack.

The main difference between the single-step and the multi-step message modifications is that the singe-step modifications always succeed. This means that all conditions that occur in the first round can be pre-fulfilled whereas this is not the case for the conditions in the second round. Consequently, it is desirable for a path search algorithm to look for paths that have most conditions in the first round of MD4.

3 Notation

This section details the notation that we will use in the remainder of this article. Furthermore we discuss the carry expansion of signed differences and the differential properties of the functions *IF* and *MAJ*. The input messages are denoted by $M = (m_0, m_1, ..., m_{15})$ and $M' = (m'_0, m'_1, ..., m'_{15})$. The intermediate steps in MD4 are computed according to (1) and the results are typically represented by the variable r_i.

3.1 Signed Differences

We follow the idea by Wang *et al.* and use signed differences. Because we only use signed differences, we will often refer to them simply as differences.

Definition 1 *The signed difference Δx between two 32-bit words x and x' is defined bitwise by*

$$\Delta x = x' - x = (\delta x_{31}, ..., \delta x_0) \quad \textit{with} \quad \delta x_j = x'_j - x_j \in \{\text{-}1, 0, 1\},\ 0 \leq j \leq 31$$

We use the following abbreviation for Δx:

$$\Delta x = \Delta[d_1, d_2, ..., d_w] \quad \textit{where} \quad d_i = \begin{cases} \text{-}j & \textit{if}\ \ \delta x_j = \text{-}1 \\ j & \textit{if}\ \ \delta x_j = 1 \end{cases}$$

Definition 2 *For a given difference $\Delta[d_1, d_2, \ldots, d_w]$, the value $|d_i|$ defines a bit position. The value w is the Hamming weight of the difference. The value $\Delta[]$ denotes the zero difference.*

A nonzero difference $\Delta x = x' - x$ already determines the values of the corresponding bits in x (and therefore in x'):

$$\Delta x = \Delta[d_1, d_2, ..., d_w] \quad \Rightarrow \quad x_{|d_i|} = \begin{cases} 0 & \text{if}\ \ sign(d_i) = 1 \\ 1 & \text{if}\ \ sign(d_i) = \text{-}1 \end{cases}$$

The difference Δx also imposes conditions on the value x.

Example 1 *The difference Δx can be represented as follows:*

$$\begin{aligned}\Delta x =& x' - x = \Delta[\text{-}27, 15, \text{-}3] \\ =&(0,0,0,0,\text{-}1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,\text{-}1,0,0,0),\end{aligned}$$

and it implies that $x_{27} = 1, x_{15} = 0, x_3 = 1$ *and* $x'_{27} = 0, x'_{15} = 1, x'_3 = 0$.

Remark. Because we use signed bit differences in our differential analysis, we need to be able to add and rotate signed bit differences throughout each step. For a detailed definition of these operations see [Dau05] or [Sch06]. We only provide one simple case and one example here.

Lemma 1 *When adding two signed bit differences $\Delta x = \Delta[dx_1]$ and $\Delta y = \Delta[dy_1]$ with hamming weight 1 the following four cases can occur:*

$$\Delta[dx_1] + \Delta[dy_1] = \begin{cases} \Delta[] & \text{if} \quad dx_1 = \text{-}dy_1 \\ \Delta[dx_1 + 1] & \text{if} \quad dx_1 = dy_1 \text{ and } sign(dx_1) = 1 \\ \Delta[\text{-}(|dx_1| + 1)] & \text{if} \quad dx_1 = dy_1 \text{ and } sign(dx_1) = \text{-}1 \\ \Delta[dx_1, dy_1] & \text{otherwise} \end{cases}$$

A difference at position 32 is always discarded. When adding signed bit differences with Hamming weight $w > 1$ a signed carry effect may occur. To rotate a signed bit difference $\Delta x = \Delta[dx_1, dx_2, ..., dx_w]$ each element is rotated as follows:

$$\Delta[dx_i] \lll s = \Delta[dy_i] \text{ where } dy_i = \begin{cases} dx_i + s \bmod 32 & \text{if } sign(dx_i) = 1 \\ \text{-}(|dx_i| + s \bmod 32) & \text{if } sign(dx_i) = \text{-}1 \end{cases}$$

Example 2 *We look at the sum of $\Delta x = \Delta[31, 27, 16, 15, 4]$ and of $\Delta y = \Delta[31, \text{-}27, 15, \text{-}3]$. Carries at positions 15, 16 and 31 occur. The carry that comes from position 31 is discarded.*

$$\begin{aligned}\Delta x + \Delta y &= \Delta[31, 27, 16, 15, 4] + \Delta[31, \text{-}27, 15, \text{-}3] = \Delta[17, 4, \text{-}3] \\ \Delta x \lll s &= \Delta[31, \text{-}27, 15, \text{-}3] \lll 5 = \Delta[20, \text{-}8, 4, \text{-}0]\end{aligned}$$

3.2 Carry Expansion of Signed Differences

Because the representation of signed differences is redundant, every (nonzero) element d_i of a signed difference can be expanded as described in the following. Note that differences at position 32 are discarded.

$$\Delta[d_1, ..., d_i, ..., d_w] = \begin{cases} \Delta[d_1, ..., \text{-}d_i, ..., d_w] + \Delta[d_i + 1] & \text{if } sign(d_i) = 1 \\ \Delta[d_1, ..., \text{-}d_i, ..., d_w] + \Delta[\text{-}(|d_i| + 1)] & \text{if } sign(d_i) = \text{-}1 \end{cases}$$

This step can be applied recursively on the resulting signed difference, and on the previous signed difference but for a different bit position. We call the number of expansion steps for each element d_i *additional carries*. A specific representation is achieved by imposing conditions on the difference.

Example 3 *In this example the difference* $\Delta x = \Delta[\text{-}11, 9]$ *is expanded, where the maximum number of expansion steps performed in each recursion branch, and thus the number of* additional carries, *is* 2 *and where the expanded element is marked with* $\overleftarrow{d_i}$*:*

$$\begin{aligned}\Delta x \to \Delta[\text{-}11, \overleftarrow{9}\,] &\to \Delta[\text{-}11, \overleftarrow{10}, \text{-}9] \to \Delta[\text{-}10, \text{-}9] \\ &\to \Delta[\text{-}\overleftarrow{11}, 10, \text{-}9] \to \Delta[\text{-}12, 11, 10, \text{-}9] \\ \to \Delta[\text{-}\overleftarrow{11}, 9] &\to \Delta[\text{-}\overleftarrow{12}, 11, 9] \to \Delta[\text{-}13, 12, 11, 9] \\ &\to \Delta[\text{-}12, \overleftarrow{11}, 9] \to \Delta[\text{-}12, 11, 10, \text{-}9]\end{aligned}$$

Hence, all representations for Δx *with a maximum carry expansion of two, sorted by their Hamming weight, are:*

$$\begin{aligned}\Delta x &= \Delta[\text{-}11, 9] = \Delta[\text{-}10, \text{-}9] \\ &= \Delta[\text{-}11, 10, \text{-}9] = \Delta[\text{-}12, 11, 9] \\ &= \Delta[\text{-}12, 11, 10, \text{-}9] = \Delta[\text{-}13, 12, 11, 9]\end{aligned}$$

An expanded signed difference can be reduced to an equivalent difference with minimum weight again. However, this is not true if the difference is rotated between expansion and reduction.

Example 4 *This example shows that the weight of an expanded difference cannot be reduced to a difference with equal weight, if the expanded part is rotated over position 31:*

$$\Delta[12] \lll 19 = \Delta[13, \text{-}12] \lll 19 = \Delta[\text{-}31, 0] \neq \Delta[31] = \Delta[12] \lll 19$$

3.3 Properties of the Functions *IF* and *MAJ*

In this section we discuss the propagation of signed differences through the functions *IF* and *MAJ*. In order to control the propagation of differences through these functions, we need to impose conditions on the input values. Table 5 (see App. A) shows all cases and conditions that allow to achieve a specific output difference of these functions.

For the *IF* function, the majority of input cases can be manipulated. However, consecutive ones in the input differences have to be avoided if a zero output difference is desired. For the *MAJ* function, we can only influence the output difference if the number of input differences is exactly one. Table 5 shows that the input difference of the *IF* function can be flipped if δx is not zero. Therefore, it can be assumed that in the first two rounds a zero output difference is possible by imposing conditions in most cases.

4 Our Algorithm for the Differential-Path Search

In Sect. 2.2, we have selected the input difference ΔM as $\Delta m_1 = 2^{31} = \Delta[31]$, $\Delta m_2 = 2^{31} - 2^{28} = \Delta[31, \text{-}28]$ and $\Delta m_{12} = \text{-}2^{16} = \Delta[\text{-}16]$. These differences are introduced in steps 1, 2 and 12 of round one and in steps 19, 20 and 24 of round two. Thus, in order to derive a differential path for MD4, the differences between step 0 and 24 have to cancel each other. The complexity of a brute-force search through all possible paths is too high. To reduce the search space of our algorithm, we have tried to avoid any uncontrolled propagation of differences through the function f_i (see Sect. 3.3) or by carry propagation (see Sect. 3.2). In order to reduce the resulting number of conditions, low weight signed differences are used by default.

The algorithm consists of three major parts which are the *target differences computation* (see Sect. 4.1), the *cancelation search* (see Sect. 4.2), and the *correction step* (see Sect. 4.3). An overview of the algorithm is given in Fig. 4 (see App. B).

In the first part, the target output difference for the function f_i is determined for every step. Therefore, the message differences are computed backward and forward to derive the so-called correction and disturbance differences. They are then combined to define the target differences.

During the cancelation search, all variations of the elements of the target differences are considered. Note, that this is done for every step of MD4. The elements of the target differences need to be canceled by using the properties of the function f_i. To achieve an output difference for f_i at a specific bit position, the input differences $\Delta r_{i\text{-}1}$, $\Delta r_{i\text{-}2}$ and $\Delta r_{i\text{-}3}$ need to be expanded. Finally, the conditions for each step are derived.

In the correction step, impossible output differences are resolved without searching for a new differential path first. If some contradictions cannot be corrected, additional differences are added to the target differences. These disturbance differences, which we typically derived by hand, distribute the conditions such that a new differential path without contradictions can be found.

4.1 Target Differences Computation

The goal of an algorithm for finding a differential path is to cancel out all differences that are introduced by the message words. Because one of the four state variables is updated in one step, a message difference can be canceled every fourth step. Hence, a message difference introduced in step i can be canceled by introducing an opposite difference in all steps $(i \pm 4k)$. To know where to introduce a difference and to determine its position and sign, the message differences are compute backward and forward (see Fig. 1 left). To reduce the complexity, no propagation through the function f_i or by a carry expansion is considered while deriving the target differences.

Disturbance differences Δd_i are simply derived by *forward* computing the message differences:

$$\begin{aligned}&\Delta d_i = \Delta d_{i-4} \lll s_{i-4} + \Delta m_{w_i}\\ &\text{with } i := \{0, 1, ..., 24\} \text{ and } \Delta d_{\text{-}4} = \Delta d_{\text{-}3} = \Delta d_{\text{-}2} = \Delta d_{\text{-}1} = 0\end{aligned}$$

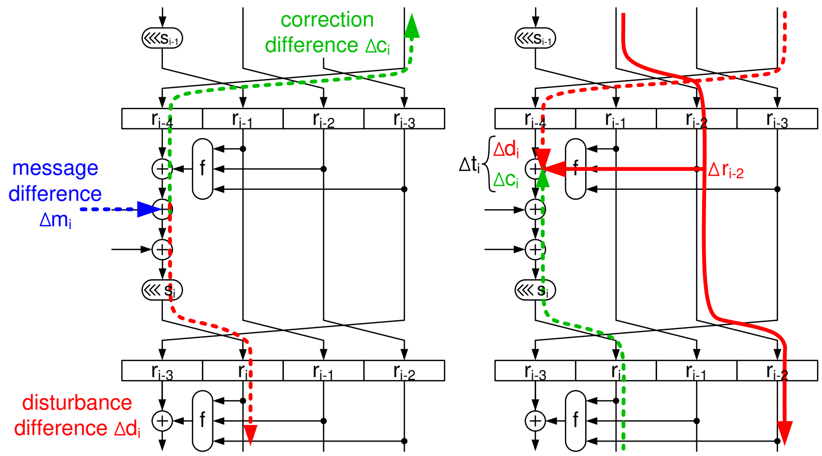

Fig. 1. Left: Forward and backward computation of the message differences to get a target difference for each step. Right: Fulfilling the target difference using the input differences of the function f_i.

Correction differences Δc_i are *backward* computed message differences. Using the correction differences it can be determined where to introduce a difference, which in turn can cancel a message difference in a subsequent step:

$$\Delta c_i = (\Delta c_{i+4} + \Delta m_{w_{i+4}}) \ggg s_i$$
$$\text{with } i := \{20, 19, ..., 0\} \text{ and } \Delta c_{24} = \Delta c_{23} = \Delta c_{22} = \Delta c_{21} = 0$$

Target differences are the merged correction and disturbance differences of each step. A target difference Δt_i in step i is the target output difference of the function f_i. This difference is known to cancel a message difference in a previous or later step. The target differences are defined by the sum of the disturbance and correction differences (2). Table 6 shows the target differences of steps $0-24$.

$$\Delta t_i = -(\Delta d_i + \Delta c_i) \tag{2}$$

4.2 Cancelation Search

In this section we describe how to find candidates for differential paths. The main concept is to cancel the target differences in each step using the function f_i. The search is performed recursively over all steps $i = \{0, ..., 24\}$. In order to cancel an element of the target difference we use carry expansions.

Variation of Target Difference Elements It is not known in advance which elements of the target differences should be canceled in what step. Therefore, in every step i all variations of the elements of the target difference Δt_i have to be considered. These variations are simply called *target variations* in the remainder. If the target difference Δt_i has Hamming weight w then there are 2^w possibilities to cancel elements of the target difference (see Tab. 3).

Carry Expansions The target variations need to be canceled by the function f_i (see Fig. 1 right). A non-zero output difference of the function f_i is only possible, if a non-zero input difference at the same bit position is available. This is usually not the case. Therefore, the input differences of the function f_i are expanded. Note, that there are again many possibilities to achieve a specific bit position. Each element of the target variation could be canceled by any carry expansion of any input difference of f_i (see Tab. 3). To limit the complexity of the search algorithm, a predefined maximum length for each of the carry expansions (usually 3) is used. Besides limiting the search space, low weight differences reduce the number of conditions as well.

Table 3. This table shows all target variations of $\Delta t_i = \Delta[\text{-}29, 20, \text{-}17]$ and all carry expansions of $\Delta r_{i\text{-}1} = \Delta[19, \text{-}17]$, $\Delta r_{i\text{-}2} = \Delta[16]$ and $\Delta r_{i\text{-}3} = \Delta[14, \text{-}7]$ with a maximum length of 2. Each target variation may be canceled by any carry expansion of these inputs of $f_i(r_{i\text{-}1}, r_{i\text{-}2}, r_{i\text{-}3})$.

Δt_i	$\Delta r_{i\text{-}1}$	$\Delta r_{i\text{-}2}$	$\Delta r_{i\text{-}3}$
$\Delta[\text{-}29, 20, \text{-}17]$	$\Delta[19, \text{-}17]$	$\Delta[16]$	$\Delta[14, \text{-}7]$
$\Delta[\text{-}29, 20\quad]$	$\Delta[18,\ 17]$	$\Delta[17, \text{-}16]$	$\Delta[15, \text{-}14, \text{-}7]$
$\Delta[\text{-}29,\quad \text{-}17]$	$\Delta[20, \text{-}19, \text{-}17]$	$\Delta[18, \text{-}17, \text{-}16]$	$\Delta[14,\ \text{-}8,\ 7]$
$\Delta[\text{-}29\quad]$	$\Delta[19, \text{-}18,\ 17]$		$\Delta[15, \text{-}14,\ \text{-}8,\ 7]$
$\Delta[\quad 20, \text{-}17]$	$\Delta[20, \text{-}19, \text{-}18,\ 17]$		$\Delta[16, \text{-}15, \text{-}14, \text{-}7]$
$\Delta[\quad 20\quad]$	$\Delta[21, \text{-}20, \text{-}19, \text{-}17]$		$\Delta[14,\ \text{-}9,\ 8,\ 7]$
$\Delta[\quad \text{-}17]$			
$\Delta[\quad]$			

Cancel Possibilities In this step it is determined which target variation can be achieved by which carry expansion. To achieve one specific target variation, all combinations of the inputs $\Delta r_{i\text{-}1}$, $\Delta r_{i\text{-}2}$ and $\Delta r_{i\text{-}3}$ of f_i can be tried. However, a target variation can only be met by an input difference, if they share at least the same bit positions. Because most inputs of f_i cannot meet this requirement anyway, the search space can be significantly reduced by considering only combinations that are possible using this principle.

In every step i of the hash function, we first start with the difference $\Delta r_{i\text{-}1}$ of f_i and try to meet all target variations by carry expanding $\Delta r_{i\text{-}1}$. Some targets will not be met at all, whereas others can be met with several expansions of $\Delta r_{i\text{-}1}$ (see Ex. 5). Note, that it cannot be determined whether a specific output difference of the function f_i is indeed possible until all of its inputs are fixed. Therefore, it is first assumed that the desired target can be canceled and verified in a later step. The input difference with the lowest weight is used by default.

Example 5 *This example shows all cancel possibilities for all variations of the target difference* $\Delta t_i = \Delta[\text{-}29, 20, \text{-}17]$. *In this example the expansions of the input* $r_{i\text{-}1} = \Delta[19, \text{-}17]$ *are considered. A target variation containing a difference at position* 29 *cannot be achieved by any input difference listed, whereas the zero target variation can be achieved by all input differences.*

$$\begin{aligned}\Delta r_{i-1} &= \Delta[19, \text{-}17] = \Delta[18, 17] = \Delta[20, \text{-}19, \text{-}17] = \Delta[19, \text{-}18, 17] \\ &= \Delta[21, \text{-}20, \text{-}19, \text{-}17] = \Delta[20, \text{-}19, \text{-}18, 17]\end{aligned}$$

$$\begin{aligned}
\Delta t_i &= \Delta[\text{-}29,\ 20, \text{-}17] &&\Longrightarrow \textit{not possible} \\
\Delta t_i &= \Delta[\text{-}29,\ 20\ \ \ \ \ \ ] &&\Longrightarrow \textit{not possible} \\
\Delta t_i &= \Delta[\text{-}29,\ \ \ \ \ \ \ \text{-}17] &&\Longrightarrow \textit{not possible} \\
\Delta t_i &= \Delta[\text{-}29\ \ \ \ \ \ \ \ \ \ \ \ \ ] &&\Longrightarrow \textit{not possible} \\
\Delta t_i &= \Delta[\ \ \ \ \ \ \ 20, \text{-}17] &&\Longrightarrow \Delta[20, \text{-}19, \text{-}17] \\
\Delta t_i &= \Delta[\ \ \ \ \ \ \ 20\ \ \ \ \ \ ] &&\Longrightarrow \Delta[20, \text{-}19, \text{-}17], \Delta[20, \text{-}19, \text{-}18, 17] \\
\Delta t_i &= \Delta[\ \ \ \ \ \ \ \ \ \ \ \ \ \text{-}17] &&\Longrightarrow \Delta[19, \text{-}17], \Delta[20, \text{-}19, \text{-}17], \Delta[21, \text{-}20, \text{-}19, \text{-}17] \\
\Delta t_i &= \Delta[\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ ] &&\Longrightarrow \textit{all expansions of } \Delta r_{i-1}
\end{aligned}$$

Already canceled elements of a target difference are removed from the target and the remaining elements are canceled in a later step. All possible target variations are examined recursively. Thus, the message differences are tried to be canceled in *all* steps $i \pm 4k$. After having processed $\Delta r_{i\text{-}1}$, we continue with $\Delta r_{i\text{-}2}$ and $\Delta r_{i\text{-}3}$. Note, that $\Delta r_{i\text{-}2}$ and $\Delta r_{i\text{-}3}$ may have already been used to cancel a previous target. Further expansions are only possible if they do not contradict these cancelations. For example, the expansion $\Delta[18, 17] \rightarrow \Delta[19, \text{-}17]$ is not possible if $\Delta[18]$ has already been used to cancel a target in a previous step.

Deriving the Conditions The used carry expansions of the input differences finally determine the conditions. Only after we have fixed all three input differences of f_i, we can determine whether a certain output difference is really possible. This is often not the case. However, one of the other cancel possibilities can be tried. In addition, in many cases a previously set condition can contradict with a newly set condition and further expansions need to be tried.

After examining all possible expansions there are usually some contradictions left. A path with at least one contradiction is called an *impossible path*. To reduce the complexity, the search in a branch with too many contradictions is aborted. The result of the cancelation search are a number of paths from step 0 to step 24 that have a zero differences in step 24, but may still have a few contradictions.

4.3 Correction Step

In the correction step, contradictions within impossible paths are corrected. In such impossible paths a specific target difference cannot be met in some step or a zero output difference of the function f_i cannot be achieved. As a consequence, these additional (disturbance) differences induced by the contradictions need to be canceled in some other step.

Correction by Solving Contradictions To cancel these additional disturbances, they are computed forward and backward through the already determined differential path. As we only need to correct a few new disturbances, longer carry expansions can be allowed. However, this does not always work because further contradictions may occur which are even harder to resolve. The reason is, that in the case of MD4 the conditions and differences stick together throughout the whole differential path (see Fig. 2).

Correction by Dispersion Differences Typically, differences propagate from the least significant bit in the first few steps to the most significant bit in the last steps (see Fig. 2). The reason for this propagation is that the rotation values s_i are very similar for most differences. In order to spread the differences and thus the conditions, *dispersion differences* (Δp_i) are introduced in steps with a high rotation value, *i.e.* $s_3 = 19$. This high rotation allows the dispersion differences to spread within only a few steps. The dispersion differences are then used in the following steps to cancel differences in areas with a low condition density. The dispersion difference $\Delta p_3 = \Delta[6]$, which we determined by hand, leads to a differential path without contradictions (see Fig. 3).

5 Experiments and results

In our experiments we tried carry expansions with different lengths and different dispersion differences. It turned out that at least in one step, a carry expansion of length three is needed. We further noticed that increasing the maximum length of the carry expansions up to 10 does not lead to less contradictions when using no dispersion differences. During the evaluation of different parameters, we were able to produce over 1000 (similar) differential paths without contradictions so far. One run of our algorithm takes only a couple of minutes. For example, using the disturbance difference $\Delta p_3 = \Delta[6]$ and a maximum carry expansion of 3, 13871 possibilities to cancel the elements of the target differences have been tried. 208 possibilities result in a zero differences in step 24 but have at least 2 contradictions. To correct the contradictions of these paths, 140068 different cancel possibilities to achieve the respective target differences were examined. Finally, 324 paths with no contradiction could be found. The overall number of steps performed was 1964131.

With respect to the message modification, the best of our paths is the one shown in Fig. 3. It has the smallest number of conditions in the second round. Remember that conditions in the first round can be easily pre-fulfilled by the single-step message modification technique. Further, most differences occur in the first few steps of the second round and are thus also easily pre-fulfilled. Our

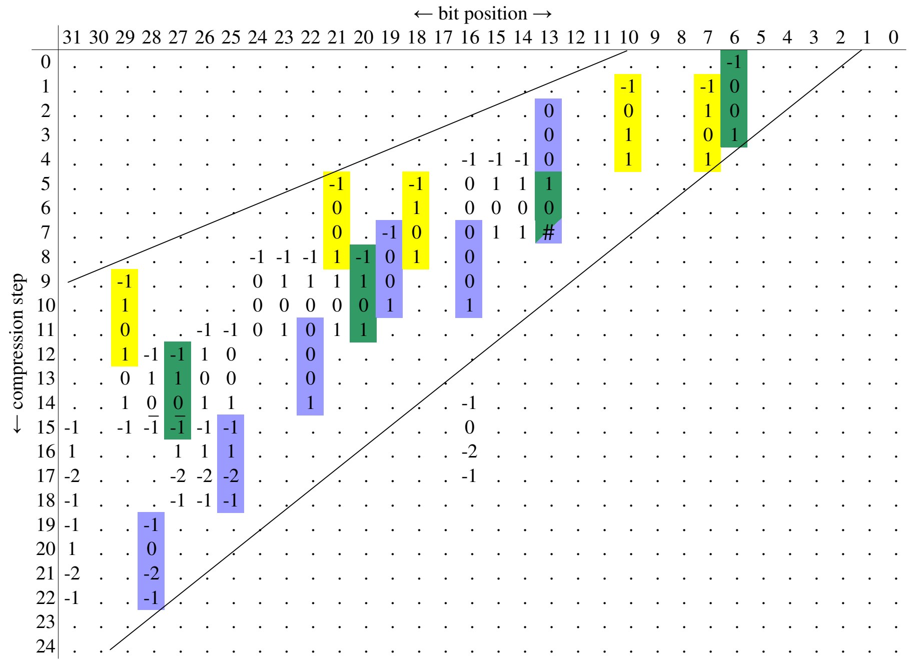

Fig. 2. This figure shows the conditions for a specific differential path. Most differences are rotated with low values. Thus, conditions do not spread (different shades show different rotation values) and so contradictions are more likely to occur although the number of conditions, which is 119, is small. A value of $c = 0$ or $c = 1$ requires $r_{i,j} = c$ for a specific bit j and step i. A negative value c or $\overline{c}$ requires the respective bit to be $r_{i,j} = r_{i\text{-}|c|,j}$ or $r_{i,j} \neq r_{i\text{-}|\overline{c}|,j}$. The entry marked by # denotes a contradiction.

path has 146 conditions with only 22 conditions in round two and 2 conditions in round three. In contrast, the path of Wang *et al.* has 122+2 conditions where 25 conditions occur in round two and 2 conditions occur in round three. The two additional conditions were found by [NSKO05]. Fig. 3 shows all conditions in our path in a graphical manner. Further details about the conditions are provided in App. D and an example of a collision is given in Tab. 4.

6 Conclusions

In this article, we have introduced an algorithm that finds differential paths for the first 24 steps of MD4 in an automated way. Our algorithm is successful: given a difference for the input message, it computes differential paths for MD4. Among the differential paths that we have found so far, there are paths that have fewer conditions in the second round than the path of Wang *et al.* This is an advantage with respect to the message modification techniques; the complexity of a collision attack based on our path is therefore lower.

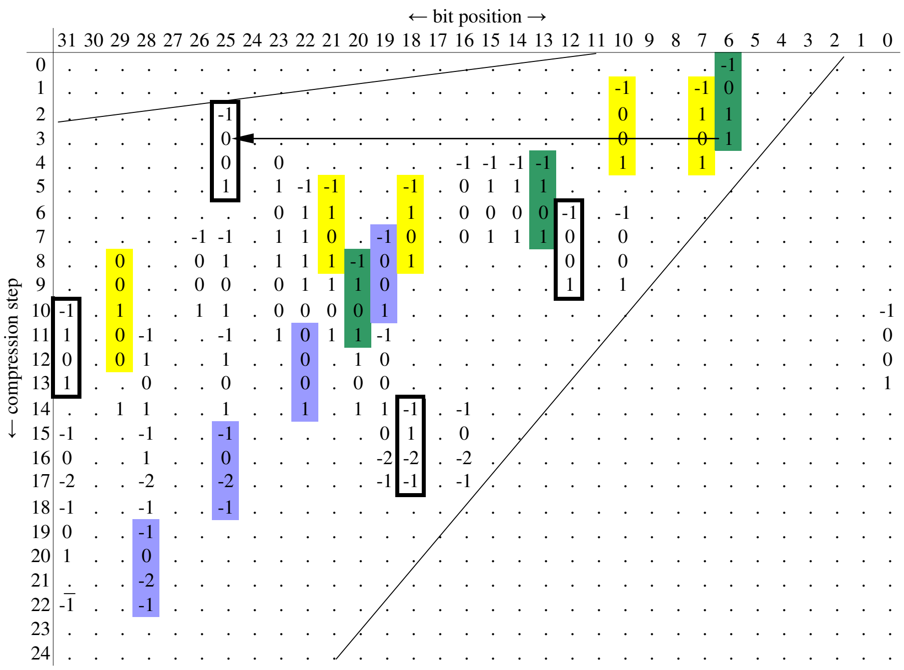

Fig. 3. In our path, differences and conditions are spread by introducing a *dispersion difference* in step $i = 3$ which has a high rotation value ($s_3 = 19$). This dispersion difference causes new conditions, which are marked by a box in this figure. Because of introducing a new difference, the number of conditions is higher (146). However, no contradictions appear.

The techniques that we have used are not very specific for MD4. The forward-backward computation for instance, which is performed in the first part of our algorithm, can be applied in general to determine the target differences. The cancelation search is general in the sense that we try out all carry expansions up to a certain length starting from the simplest one. The use of dispersion differences must be carefully considered for other algorithms where the conditions might be more distributed anyway. In addition, the approach of trying the simplest differences first, and only enlarging the search space if necessary, is also algorithm independent.

Summing up, we have made the first successful step towards an automated search for differential paths, which is the crucial part of Wang *et al.*'s attacks.

7 Acknowledgements

We would like to thank the anonymous referees and the members of IAIK's Krypto group for their helpful comments.

Table 4. One collision of MD4 using our differential path.

M_0	9de70013 4b5611b3 d2ce37bb d3fbfd91 25bb4551 42d059f8 41b1bd57 19ed222e 4c9c5258 20df2cbf d868c1a8 314acd01 e4aca811 5089a823 bb1912b1 2b61d489
M'_0	9de70013 cb5611b3 42ce37bb d3fbfd91 25bb4551 42d059f8 41b1bd57 19ed222e 4c9c5258 20df2cbf d868c1a8 314acd01 e4aba811 5089a823 bb1912b1 2b61d489
H_0	877bd941 14da836a 0af87c2e 143a4028

References

[ABB+05] Daniel Augot, Alex Biryukov, An Braeken, Carlos Cid, Hans Dobbertin, Hakan Englund, Henri Gilbert, Louis Granboulan, Helena Handschuh, Martin Hell, Thomas Johansson, Alexander Maximov, Matthew Parkerand Thomas Pornin, Bart Preneel, Matt Robshaw, and Michael Ward. Ongoing Research Areas in Symmetric Cryptography, January 2005.

[Dau05] Magnus Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-Universität Bochum, May 2005.

[Dob98] Hans Dobbertin. Cryptanalysis of MD4. *Journal of Cryptology*, 11(4):253–271, 1998.

[HPR04] Philip Hawkes, Michael Paddon, and Gregory G. Rose. Musings on the Wang et al. MD5 Collision. Cryptology ePrint Archive, Report 2004/264, 2004.

[NSKO05] Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Attack on MD4. Cryptology ePrint Archive, Report 2005/151, 2005. http://eprint.iacr.org/.

[Sch06] Martin Schläffer. Cryptanalysis of MD4. Master's thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, February 2006.

[WLF+05] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2005.

[WY05] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2005.

[WYY05a] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer, 2005.

[WYY05b] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

A Differential Characteristic of *IF* and *MAJ*

Table 5. Signed differential characteristic of the *IF* and *MAJ* function, with necessary conditions, probability 1 or "-" if the desired output is not possible.

$\delta x \delta y \delta z$	$\delta IF = 0$	$\delta IF = 1$	$\delta IF =$ -1	$\delta MAJ = 0$	$\delta MAJ = 1$	$\delta MAJ =$ -1
0 0 0	1	-	-	1	-	-
0 0 1	$x = 1$	$x = 0$	-	$x = y$	$x \neq y$	-
0 0 -1	$x = 1$	-	$x = 0$	$x = y$	-	$x \neq y$
0 1 0	$x = 0$	$x = 1$	-	$x = z$	$x \neq z$	-
0 1 1	-	1	-	-	1	-
0 1 -1	-	$x = 1$	$x = 0$	1	-	-
0 -1 0	$x = 0$	-	$x = 1$	$x = z$	-	$x \neq z$
0 -1 1	-	$x = 0$	$x = 1$	1	-	-
0 -1 -1	-	-	1	-	-	1
1 0 0	$y = z$	$y = 1, z = 0$	$y = 0, z = 1$	$y = z$	$y \neq z$	-
1 0 1	$y = 0$	$y = 1$	-	-	1	-
1 0 -1	$y = 1$	-	$y = 0$	1	-	-
1 1 0	$z = 1$	$z = 0$	-	-	1	-
1 1 1	-	1	-	-	1	-
1 1 -1	1	-	-	-	1	-
1 -1 0	$z = 0$	-	$z = 1$	1	-	-
1 -1 1	1	-	-	-	1	-
1 -1 -1	-	-	1	-	-	1
-1 0 0	$y = z$	$y = 0, z = 1$	$y = 1, z = 0$	$y = z$	-	$y \neq z$
-1 0 1	$y = 1$	$y = 0$	-	1	-	-
-1 0 -1	$y = 0$	-	$y = 1$	-	-	1
-1 1 0	$z = 0$	$z = 1$	-	1	-	-
-1 1 1	-	1	-	-	1	-
-1 1 -1	1	-	-	-	-	1
-1 -1 0	$z = 1$	-	$z = 0$	-	-	1
-1 -1 1	1	-	-	-	-	1
-1 -1 -1	-	-	1	-	-	1

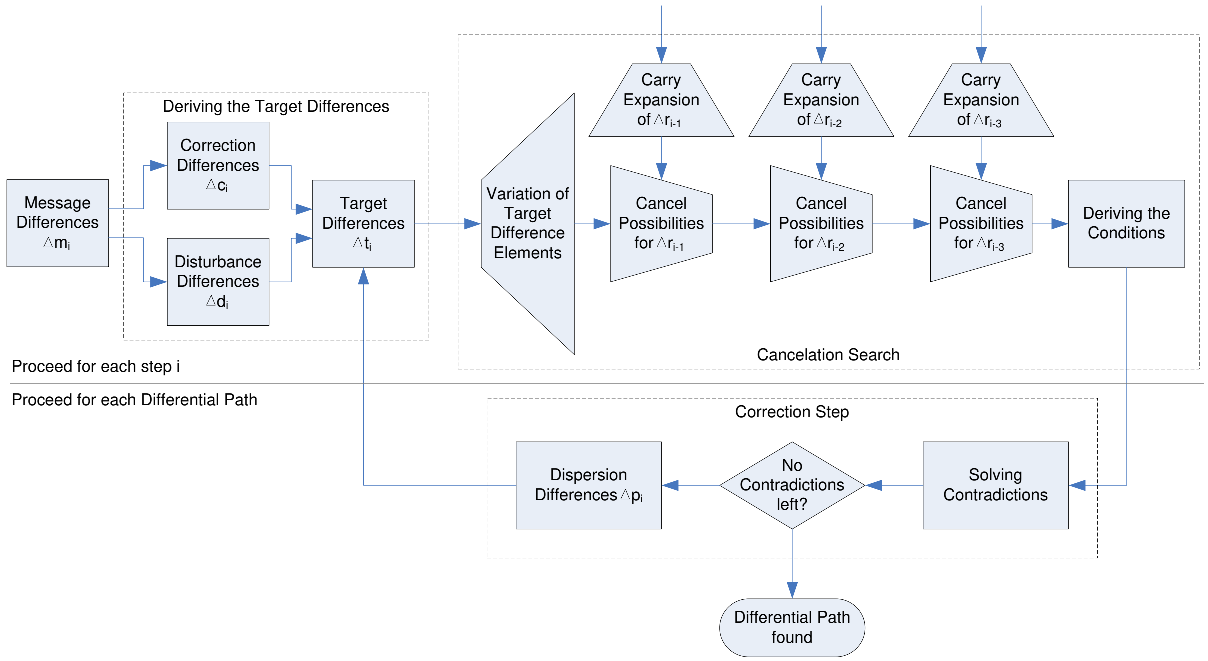

Fig. 4. An overview of the differential path search algorithm. Trapezoids represent expansions and restrictions of the search space.

C The Target Differences

Table 6. Deriving the target differences Δt_i by forward computation of the disturbance differences Δd_i, and backward computation of the correction differences Δc_i for all message differences of step $0 - 24$. The differences caused by m_{12} of step 12 are highlighted.

step	Δm_{w_i}	s_i	Δd_i	Δc_i	Δt_i
0		3		Δ[16, 13, -10, -7]	Δ[-16, -13, 10, 7]
1	Δ[31]	7	Δ[31]		Δ[-31]
2	Δ[31, -28]	11	Δ[31, -28]		Δ[-31, 28]
3		19		Δ[-4]	Δ[4]
4		3		Δ[19, 16, -13, -10]	Δ[-19, -16, 13, 10]
5		7	Δ[6]		Δ[-6]
6		11	Δ[10, -7]		Δ[-10, 7]
7		19		Δ[-23]	Δ[23]
8		3		Δ[22, 19, -16, -13]	Δ[-22, -19, 16, 13]
9		7	Δ[13]		Δ[-13]
10		11	Δ[21, -18]		Δ[-21, 18]
11		19		Δ[-10]	Δ[10]
12	Δ[-16]	3	Δ[-16]	Δ[25, 22, -19]	Δ[-25, -22, 19, 16]
13		7	Δ[20]		Δ[-20]
14		11	Δ[-29, 0]		Δ[29, -0]
15		19		Δ[-29]	Δ[29]
16		3	Δ[-19]	Δ[28, 25, -22]	Δ[-28, -25, 22, 19]
17		5	Δ[27]		Δ[-27]
18		9	Δ[11, -8]		Δ[-11, 8]
19	Δ[-16]	13	Δ[-16]		Δ[16]
20	Δ[31]	3	Δ[31, -22]	Δ[28, -25]	Δ[-31, -28, 25, 22]
21		5	Δ[0]		Δ[-0]
22		9	Δ[20, -17]		Δ[-20, 17]
23		11	Δ[-29]		Δ[29]
24	Δ[31, -28]	3	Δ[31, -28, -25, 2]		Δ[-31, 28, 25, -2]

D Detailed Description of our Path.

Table 7. Differential characteristic of our differential path for MD4.

Step	r_i	s_i	m_{w_i}	Δm_{w_i}	Δf_i	Δr_i
0	r_0	3	m_0			
1	r_1	7	m_1	$\Delta[31]$		$\Delta[6]$
2	r_2	11	m_2	$\Delta[31,\text{-}28]$		$\Delta[10,\text{-}7]$
3	r_3	19	m_3		$\Delta[6]$	$\Delta[25]$
4	r_4	3	m_4			
5	r_5	7	m_5			$\Delta[16,\text{-}15,\text{-}14,\text{-}13]$
6	r_6	11	m_6			$\Delta[23,\text{-}22,\text{-}21,\text{-}18]$
7	r_7	19	m_7		$\Delta[23]$	$\Delta[12,10]$
8	r_8	3	m_8		$\Delta[23,\text{-}22,16]$	$\Delta[26,\text{-}25,19]$
9	r_9	7	m_9			$\Delta[23,\text{-}22,\text{-}21,\text{-}20]$
10	r_{10}	11	m_{10}		$\Delta[\text{-}21]$	$\Delta[\text{-}29]$
11	r_{11}	19	m_{11}			$\Delta[\text{-}31,29,0]$
12	r_{12}	3	m_{12}	$\Delta[\text{-}16]$	$\Delta[\text{-}22]$	$\Delta[29,\text{-}28,\text{-}25,22,\text{-}20,19]$
13	r_{13}	7	m_{13}		$\Delta[\text{-}20]$	
14	r_{14}	11	m_{14}		$\Delta[29]$	
15	r_{15}	19	m_{15}			$\Delta[19,\text{-}18,16]$
16	r_{16}	3	m_0		$\Delta[19]$	$\Delta[31,\text{-}28,25]$
17	r_{17}	5	m_4			
18	r_{18}	9	m_8			
19	r_{19}	13	m_{12}	$\Delta[\text{-}16]$		$\Delta[31]$
20	r_{20}	3	m_1	$\Delta[31]$		$\Delta[\text{-}31,28]$
21	r_{21}	5	m_5			
22	r_{22}	9	m_9			
23	r_{23}	13	m_{13}		$\Delta[\text{-}31]$	
24	r_{24}	3	m_2	$\Delta[31,\text{-}28]$		
...	...					
35	r_{35}	15	m_{12}	$\Delta[\text{-}16]$		$\Delta[\text{-}31]$
36	r_{36}	3	m_2	$\Delta[31,\text{-}28]$	$\Delta[\text{-}31]$	$\Delta[\text{-}31]$
37	r_{37}	9	m_{10}			
38	r_{38}	11	m_6			
39	r_{39}	15	m_{14}			
40	r_{40}	3	m_1	$\Delta[31]$		

Table 8. Conditions for our differential path.

Step	Conditions for r_i
0	$r_{0,6} = r_{-1,6}$
1	$r_{1,6} = 0, r_{1,7} = r_{0,7}, r_{1,10} = r_{0,10}$
2	$r_{2,6} = 1, r_{2,7} = 1, r_{2,10} = 0, r_{2,25} = r_{1,25}$
3	$r_{3,6} = 1, r_{3,7} = 0, r_{3,10} = 0, r_{3,25} = 0$
4	$r_{4,7} = 1, r_{4,10} = 1, r_{4,13} = r_{3,13}, r_{4,14} = r_{3,14}, r_{4,15} = r_{3,15}, r_{4,16} = r_{3,16},$ $r_{4,23} = 0, r_{4,25} = 0$
5	$r_{5,13} = 1, r_{5,14} = 1, r_{5,15} = 1, r_{5,16} = 0, r_{5,18} = r_{4,18}, r_{5,21} = r_{4,21}, r_{5,22} = r_{4,22},$ $r_{5,23} = 1, r_{5,25} = 1$
6	$r_{6,10} = r_{5,10}, r_{6,12} = r_{5,12}, r_{6,13} = 0, r_{6,14} = 0, r_{6,15} = 0, r_{6,16} = 0, r_{6,18} = 1,$ $r_{6,21} = 1, r_{6,22} = 1, r_{6,23} = 0$
7	$r_{7,10} = 0, r_{7,12} = 0, r_{7,13} = 1, r_{7,14} = 1, r_{7,15} = 1, r_{7,16} = 0, r_{7,18} = 0, r_{7,19} = r_{6,19},$ $r_{7,21} = 0, r_{7,22} = 1, r_{7,23} = 1, r_{7,25} = r_{6,25}, r_{7,26} = r_{6,26}$
8	$r_{8,10} = 0, r_{8,12} = 0, r_{8,18} = 1, r_{8,19} = 0, r_{8,20} = r_{7,20}, r_{8,21} = 1, r_{8,22} = 1, r_{8,23} = 1,$ $r_{8,25} = 1, r_{8,26} = 0, r_{8,29} = 0$
9	$r_{9,10} = 1, r_{9,12} = 1, r_{9,19} = 0, r_{9,20} = 1, r_{9,21} = 1, r_{9,22} = 1, r_{9,23} = 0, r_{9,25} = 0,$ $r_{9,26} = 0, r_{9,29} = 0$
10	$r_{10,0} = r_{9,0}, r_{10,19} = 1, r_{10,20} = 0, r_{10,21} = 0, r_{10,22} = 0, r_{10,23} = 0, r_{10,25} = 1,$ $r_{10,26} = 1, r_{10,29} = 1, r_{10,31} = r_{9,31}$
11	$r_{11,0} = 0, r_{11,19} = r_{10,19}, r_{11,20} = 1, r_{11,21} = 1, r_{11,22} = 0, r_{11,23} = 1,$ $r_{11,25} = r_{10,25}, r_{11,28} = r_{10,28}, r_{11,29} = 0, r_{11,31} = 1$
12	$r_{12,0} = 0, r_{12,19} = 0, r_{12,20} = 1, r_{12,22} = 0, r_{12,25} = 1, r_{12,28} = 1, r_{12,29} = 0,$ $r_{12,31} = 0$
13	$r_{13,0} = 1, r_{13,19} = 0, r_{13,20} = 0, r_{13,22} = 0, r_{13,25} = 0, r_{13,28} = 0, r_{13,31} = 1$
14	$r_{14,16} = r_{13,16}, r_{14,18} = r_{13,18}, r_{14,19} = 1, r_{14,20} = 1, r_{14,22} = 1, r_{14,25} = 1,$ $r_{14,28} = 1, r_{14,29} = 1$
15	$r_{15,16} = 0, r_{15,18} = 1, r_{15,19} = 0, r_{15,25} = r_{14,25}, r_{15,28} = r_{14,28}, r_{15,31} = r_{14,31}$
16	$r_{16,16} = r_{14,16}, r_{16,18} = r_{14,18}, r_{16,19} = r_{14,19}, r_{16,25} = 0, r_{16,28} = 1, r_{16,31} = 0$
17	$r_{17,16} = r_{16,16}, r_{17,18} = r_{16,18}, r_{17,19} = r_{16,19}, r_{17,25} = r_{15,25}, r_{17,28} = r_{15,28},$ $r_{17,31} = r_{15,31}$
18	$r_{18,25} = r_{17,25}, r_{18,28} = r_{17,28}, r_{18,31} = r_{17,31}$
19	$r_{19,28} = r_{18,28}, r_{19,31} = 0$
20	$r_{20,28} = 0, r_{20,31} = 1$
21	$r_{21,28} = r_{19,28}$
22	$r_{22,28} = r_{21,28}, r_{22,31} = \overline{r_{18,31}}$
23	
24	