Automatic Search of Differential Path in MD4

Pierre-Alain Fouque, Gaëtan Leurent, Phong Nguyen

Laboratoire d'Informatique de l'École Normale Supérieure,
Département d'Informatique,
45 rue d'Ulm, 75230 Paris Cedex 05, France
{Pierre-Alain.Fouque,Gaetan.Leurent,Phong.Nguyen}@ens.fr

Abstract. In 2004, Wang *et al.* obtained breakthrough collision attacks on the main hash functions from the MD4 family. The attacks are differential attacks in which one closely follows the inner steps of the underlying compression function, based on a so-called *differential path*. It is generally assumed that such differential paths were found "by hand". In this paper, we present an algorithm which automatically finds suitable differential paths, in the case of MD4. As a first application, we obtain new differential paths for MD4, which improve upon previously known MD4 differential paths. This algorithm could be used to find new differential paths, and to build new attacks against MD4.
Key words: MD4, collisions, differential path.

1 Introduction

Hash functions are fundamental primitives used in many cryptographic schemes and protocols. In a breakthrough work, Wang *et al.* recently discovered devastating collision attacks [19,21,22,20] on the main hash functions from the MD4 family, *e.g.* MD4 [19], RIPE-MD [19], MD5 [21], SHA-0 [22] and SHA-1 [20]. Such attacks can find collisions in much less time than the birthday paradox. Despite the efficiency of these new attacks, their impact on the security of existing hash-based cryptographic schemes is unclear, for at least two reasons: the applications of hash functions rely on various security properties which may be much weaker than collision resistance (such as pseudorandomness); Wang *et al.*'s attacks are still not completely understood.

In the past few years, much work [8,18,1,9,13] has been devoted to better understand the new attacks [3,19,21,22,20]. Roughly speaking, attacks *à la Wang* first select a specific *message difference* Δ such that carefully selected message pairs of the form $(M, M+\Delta)$ will collide for the hash function. To do this, one specifies a *differential path*: during the computation of the hash function on respectively M and $M+\Delta$, the internal state of the hash function varies at each step of the compression function, depending on the particular value of M; the differential path specifies a particular variation that guarantees $(M, M+\Delta)$ to be a hash collision. Next, one computes a *set of sufficient conditions* on the internal state (and sometimes on the message) such that if the message M satisfies all the conditions, then the pair $(M, M+\Delta)$ is guaranteed to follow the differential path, and will therefore give a collision. Finally, using *message modifications*, one shows how to satisfy many conditions deterministically, and therefore efficiently find messages M satisfying all the sufficient conditions. The final stages where one computes a set of sufficient conditions and finds suitable message modifications are arguably well-understood now. However, the search for a suitable differential path remains mysterious.

Our Results. This paper focuses on differential paths for the MD4 hash function. We present a new way to search for differential paths, based on a novel internal representation of the path, and we hope that this will give a better understanding of the notion of differential path.

The search algorithm has some applications. It allows to improve previous attacks: namely, we have found better paths for MD4 collisions based on [19,16] and for second-preimage attacks on MD4 on weak messages [23]. More precisely, the new collision paths lead to fewer (or equal) conditions in each of the three rounds of the compression function; and the new second-preimage path decreases the total number of conditions, which therefore increases the success probability.

The search algorithm also allows us to test new message differences, or to search for differential path with some other specific property. We believe this is an interesting tool, and this could led to new kind of attacks against MD4. For instance, Sasaki *et al.* have shown in the rump session of FSE 2007 how to improve an attack against APOP using a new differential path [14]. Since MD4 is believed to be quite weak, it is expected that more powerful attacks than mere collisions are possible, and our algorithm could be used to find differential paths adapted to specific attacks. This is a work in progress: we are trying to find new applications and new differential paths using our algorithm.

Related Work. Wang *et al.* presented two differential paths for MD4: the first one [19] was designed to find collisions efficiently, while the second one [23] was better suited to find second preimages of weak messages. It is usually assumed that such paths have been found "by hand".

At FSE '06, Schläffer and Oswald [16] presented an algorithm to automatically find differential paths in MD4, and interestingly found a new path, which is better suited for collision search than the first path [19]. However, the search algorithm was not fully automatic.

At ASIACRYPT '06, De Cannière and Rechberger [2] proposed a method to find differential paths in SHA-1, and gave a two-block collision for 64-step SHA-1 based on a new characteristic. They use a generalized notion of the differential path, and provide a way to estimate the work-factor to find a collision, using a slightly modified message-finding algorithm.

At FSE '07, Sasaki *et al.* proposed a new message difference that allows a more efficient collision attack on MD4 [15]. He gave a differential path with this message difference and some insight on how he improved the algorithm from Schläffer and Oswald to find this path.

Road Map. This paper is divided in three sections: we will first give some general background and notations about MD4, and Wang's attack, then we will present our algorithm, and in the last section we will show some applications of this algorithm.

2 Background and notation

Unfortunately, there does not seem to be any standard notation in the hash function literature. Here, we will use a notation similar to that of Daum [6].

2.1 MD4

MD4 follows the Merkle-Damgård construction. Its compression function is designed to be very efficient using 32-bit words and operations implemented in hardware in most processors:

- rotation $\lll$;
- addition $\bmod 2^{32}$ $\boxplus$;
- bitwise Boolean operations Φ_i, among:
  - $\mathrm{IF}(x, y, z) \quad = (x \wedge y) \vee (\neg x \wedge z)$ if $0 \leq i < 16$
  - $\mathrm{MAJ}(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ if $16 \leq i < 32$
  - $\mathrm{XOR}(x, y, z) = x \oplus y \oplus z$ if $32 \leq i < 48$

The compression function uses an internal state of four words, and updates them one by one in 48 steps. Here, we will assign a name to every different value of these registers, so the description is different from the standard one: the value changed on step i is called Q_i. Then the MD4 compression function is given by:

$$\boxed{\begin{array}{r l} \text{Step update: } & Q_i = (Q_{i-4} \boxplus \Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) \boxplus m_i \boxplus k_i) \lll s_i \\ \text{Input: } & Q_{-4} || Q_{-1} || Q_{-2} || Q_{-3} \\ \text{Output: } & Q_{-4} \boxplus Q_{44} || Q_{-1} \boxplus Q_{47} || Q_{-2} \boxplus Q_{46} || Q_{-3} \boxplus Q_{45} \end{array}}$$

The security of the compression function was based on the fact that such operations are not "compatible" and mix the properties of the input.

2.2 Wang's Attack against MD4

Wang *et al.* published a very efficient collision attack for MD4 at EUROCRYPT '05 [19]. This attack is a differential one, and is divided in two main parts:

1. A precomputation phase:
   - choose a message difference Δ
   - find a differential path
   - compute a set of sufficient conditions
2. Search for a message M satisfying all the conditions; then $\mathrm{MD4}(M) = \mathrm{MD4}(M + \Delta)$.

The differential path specifies how the computations of $\mathrm{MD4}(M)$ and $\mathrm{MD4}(M + \Delta)$ are related: it tells how the differences introduced in the message will evolve in the internal state Q_i. If we choose Δ with a low Hamming weight, and some extra properties, we can find some differences in the Q_i that are very likely. Then we look at each step of the compression function, and we can express a set of sufficient conditions that will make the Q_i's follow the path. These conditions are on the bits of Q_i, so we can not directly find a message satisfying them (and the probability that a random message fulfills them is too low).

Wang introduced three important ideas to make such an attack possible:

- The path is specified with a signed difference on the bits, which contains both the modular difference and the XOR-difference.
- Once a path is chosen, it is possible to compute a set of conditions on the internal states Q_i which are sufficient for a message M to collide with $M + \Delta$.
- Some of these conditions can be fulfilled deterministically through message modifications, and the rest will be statistical by trial and error; then the number of messages we need to try is low enough for a practical attack.

This first part of Wang's attack is not very well understood, and there are very few papers about it [13,16,2]. It is believed that Wang's paths were found by hand, and all the work on the second part of the attack just uses Wang's path (*eg.* [11,1,18,9]). In this paper we study this first part in the case of MD4, and we give an algorithm to find differential paths.

2.3 Notation

In this paper we use $\delta(x, y) = y \boxminus x$ to denote the modular difference and $\partial(x, y) = \left\langle y^{[31]} - x^{[31]}, y^{[30]} - x^{[30]}, ...y^{[1]} - x^{[1]}, y^{[0]} - x^{[0]} \right\rangle$ to denote Wang's difference. We will use $\blacktriangle$ and $\blacktriangledown$ to represent $+1$ and -1, and we will give a compact representation by omitting the zeroes, and grouping the bits, eg. $\left\langle \blacktriangle^{[0]}, \blacktriangledown\blacktriangle^{[3,4]}, \blacktriangle\blacktriangle^{[30,31]} \right\rangle$ stands for $\langle 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0, 1 \rangle$. We use $x^{[k]}$ to represent the $k+1$-st bit of x, that is $x^{[k]} = (x \ggg k) \bmod 2$ (note that we count bits and steps starting from 0).

We will consider two messages M and M', and we use a prime to represent any variable related to the message M' (*eg.* Q'_i, m'_i). As a shortcut, we will sometimes use δX (resp. ∂X) to represent $\delta(X, X')$ (resp. $\partial(X, X')$), and Φ_i for $\Phi_i(Q_{i-1}, Q_{i-2}, Q_{i-3})$.

When we are given a differential path, we will call it ∂_i, and a message follows the path if $\partial Q_i = \partial_i$ holds for every step i. We will also use δ_i as the desired value of δQ_i.

3 Automatic Search of Differential Paths

Before giving the description of the algorithm itself, let us study some useful properties of the different operations used in MD4. MD4 security is based on the interaction of incompatible operations, so we will see how to unify them. Some of these results are already in the literature (eg. [6], [13]), but we give them together using our notations.

3.1 Mathematical Toolbox

The first tool to study MD4 operations is Wang's ∂ difference. It contains both the modular difference δ and the XOR-difference, and we will often have to switch between these representations.

Relation between the modular difference and the ∂-difference. If the value of $\partial(x, y)$ is known, then we know the value of $\delta(x, y)$, but a given $\delta(x, y)$ can be satisfied with different $\partial(x, y)$, with some carry extensions. For instance, if $\delta(x, y) = 2^{27}$, we can have $\partial(x, y) = \left\langle \blacktriangle^{[27]} \right\rangle$ or $\left\langle \blacktriangledown\blacktriangle^{[27,28]} \right\rangle = 2^{28} - 2^{27}$ or $\left\langle \blacktriangledown\blacktriangledown\blacktriangle^{[27,28,29]} \right\rangle = 2^{29} - 2^{28} - 2^{27}$... up to $\left\langle \blacktriangledown\blacktriangledown\blacktriangledown\blacktriangledown\blacktriangle^{[27...31]} \right\rangle$ and $\left\langle \blacktriangledown\blacktriangledown\blacktriangledown\blacktriangledown\blacktriangledown^{[27...31]} \right\rangle$.

However, if $\delta(x, y)$ is written in a way that satisfies some extra conditions, we can compute $\partial(x, y)$:

Theorem 1. *Let $x, y \in \mathbb{Z}_{2^{32}}$. Then:*

$$\partial(x, y) = \langle \varepsilon_{31}, \varepsilon_{30}, ...\varepsilon_0 \rangle \iff \begin{cases} \sum_{j=0}^{31} \varepsilon_j 2^j = \delta(x, y) \\ \forall j, \varepsilon_j \in \{-1, 0, +1\} \\ \forall j : \varepsilon_j = +1 \implies x^{[j]} = 0 \\ \forall j : \varepsilon_j = -1 \implies x^{[j]} = 1 \end{cases}$$

Proof. The "$\Rightarrow$" direction is easy. Reciprocally, let $\langle \varepsilon_j \rangle_{j=0}^{31}$ and $\langle \varepsilon'_j \rangle_{j=0}^{31}$ be two sequences which fulfill the right-hand side conditions. Then we have $\sum (\varepsilon_j - \varepsilon'_j) 2^j = 0$, and every $\varepsilon_j - \varepsilon'_j$ is in $\{-1, 0, +1\}$ because of the last two conditions. By reducing this sum modulo two, one sees that $\varepsilon_0 - \varepsilon'_0 = 0 \bmod 2$, therefore $\varepsilon_0 - \varepsilon'_0 = 0$. By iterating, one sees that $\forall j, \quad \varepsilon_j = \varepsilon'_j$. Hence the sequence is unique, and since $\partial(x, y)$ is one candidate, it is the only one. $\square$

Note that some of the conditions depend on x; if only $\delta(x, y)$ is known, we have many possible $\partial(x, y)$, but when bits of x are known, the set of possible $\partial(x, y)$ gets smaller (if x is completely known, we know y and therefore $\partial(x, y)$).

Interactions between modular difference and rotation. To build a differential path, we need to know how a modular difference is affected by a rotation. This turns out to be rather easy, due to the following result (see Appendix A for a detailed proof):

Theorem 2. *Let $a, b \in \mathbb{Z}_{2^{32}}$, $0 \leq s < 32$ and $\alpha = a \lll s, \beta = b \lll s$. Then we may compute $\upsilon = \delta(\alpha, \beta)$ from $u = \delta(a, b)$ and a, as follows:*

$$\upsilon = \begin{cases} \upsilon_1 = (u \lll s) & \text{if } a + u < 2^{32} \text{ and} \\ & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) < 2^{32-s} \\ \upsilon_2 = (u \lll s) \boxplus 1 & \text{if } a + u < 2^{32} \text{ and} \\ & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) \geq 2^{32-s} \\ \upsilon_3 = (u \lll s) \boxminus 2^s & \text{if } a + u \geq 2^{32} \text{ and} \\ & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) < 2^{32-s} \\ \upsilon_4 = (u \lll s) \boxminus 2^s \boxplus 1 & \text{if } a + u \geq 2^{32} \text{ and} \\ & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) \geq 2^{32-s} \end{cases}$$

So, if we known $\delta(a, b)$, we can choose a value for $\delta(a \lll s, b \lll s)$, and with some extra conditions on a, this will be the correct one.

Remark. In [13], Sasaki *et al.* used the fact that there are only 4 possible values, but did not give a proof of this, and they computed these values by exhaustive search over the 2^{32} inputs.

Interactions between the ∂-difference and the boolean functions. The main advantage of the signed difference ∂ over the modular difference δ is to handle the boolean function. The Φ_i's are bitwise functions and we know for each bit how the input and output are supposed to change between M and M'; if we add some conditions to restrict the inputs, we can make sure the output follows the path. See Table 3 in Appendix C for the full conditions.

Based on these tools, we will first show how to compute a set of sufficient conditions once a differential path is given. As such, it can be used to check a given differential path, and it will be the basis of our differential path search algorithm.

3.2 Computing a Set of Sufficient Conditions

The technique used here is rather simple, and is the same as [13]. This algorithm (referred to as SC algorithm) will take as input a message difference Δ and a differential path $\langle \partial_i \rangle_{i=0}^{48}$.

The SC algorithm will follow the path backwards from Q_{48} to Q_0, and will recursively compute a set of conditions: at each step i, we assume that the current set is sufficient to satisfy the path from step Q_{i+1} to Q_{48}, and we will add some conditions to extend it to step i. If we look at step $i+4$ for messages M and M', we have:

$$Q_{i+4} = (Q_i \boxplus \Phi_{i+4}(Q_{i+3}, Q_{i+2}, Q_{i+1}) \boxplus m_{i+4} \boxplus k_{i+4}) \lll s_{i+4}$$
$$Q'_{i+4} = (Q'_i \boxplus \Phi_{i+4}(Q'_{i+3}, Q'_{i+2}, Q'_{i+1}) \boxplus m'_{i+4} \boxplus k_{i+4}) \lll s_{i+4}$$

We know how to compute $\delta^{\ggg}_{i+4} = \delta(Q_{i+4} \ggg s_{i+4}, Q'_{i+4} \ggg s_{i+4})$ from δ_{i+4} (see Section 3.1), and this will give a first set of conditions on Q_{i+4} (we call these conditions $\lll$-conditions). Then we will add some extra conditions so that the path is followed:

1. If $\Phi'_{i+4} \boxminus \Phi_{i+4} = \delta_i \boxminus \delta^{\ggg}_{i+4} \boxplus \Delta_{i+4}$, then $\delta Q_i = \delta_i$, so we select a $\partial(\Phi_{i+4}, \Phi'_{i+4})$, and we can ensure it is followed by adding a few extra conditions on the inputs Q_{i+1}, Q_{i+2}, Q_{i+3} of Φ_{i+4} (see Table 3). We call these conditions Φ-conditions.
2. Once we have $\delta Q_i = \delta_i$, we only need a few extra conditions on Q_i to get $\partial Q_i = \partial_i$ by Theorem 1. We call these conditions ∂-conditions.

3.3 The Differential Path Search Algorithm

Our algorithm is based on the SC algorithm. The basic idea is to run the SC computation, but since we do not know δ_i nor $\delta\Phi_i$, we will assume that $\delta\Phi_i = 0$, which gives $\delta_i = \delta^{\ggg}_{i+4} \boxminus \Delta_{i+4}$: the differences will only appear every 4 turns, and will not propagate in between. This is possible in the first two rounds, because the boolean functions IF and

MAJ can absorb one input difference. Using this basic idea, we find a path with a non-zero difference in $Q_{-4}...Q_{-1}$, that is, a path leading to pseudo-collisions (this initial path is called ϵ in the algorithm).

Then we will run another pass of the algorithm, but we will try to modify the path so as to lower the number of differences in the IV. In fact, we will have a set of paths $\mathcal{P}$, and every run will select a path, try to enhance it, and insert new paths in this set. This basic structure is described in Algorithm 1: we will make an extensive use of recursivity to explore the path space. This algorithm will be referred to as the DP algorithm.

Algorithm 1 Overview of the differential path search algorithm

function PATHFIND
    $\mathcal{P} \leftarrow \{\epsilon\}$ ▷ ϵ is the path with $\delta\Phi_i = 0$
    loop
        extract P from $\mathcal{P}$
        PATHSTEP(P,ϵ,48)   ▷ start search from last step
function PATHSTEP(P_0,P,i)   ▷ Extend path P to step i, following P_0
    if $i < 0$ **then**
        add P to $\mathcal{P}$
    else
        for all possible choice P' **do**
            PATCHTARGET(P_0,P',i)
function PATCHTARGET(P_0,P,i)   ▷ Modify P to fix IV differences in the end
    for all possible choice P' **do**
        PATCHCARRIES(P_0,P',i)
function PATCHCARRIES(P_0,P,i)   ▷ Extend some carries to help the next steps
    for all possible choice P' **do**
        PATHSTEP(P_0,P',$i-1$)

Path representation. During the computation of a path, we represent the path as $\langle \partial Q_i \rangle_{i=0}^{48}$, where each ∂Q_i is given as 32 values in $\{-1, 0, +1\}$. However, between two passes, this representation is almost useless: when we apply a local modification to a ∂Q_i, the ∂Q_j's for the rest of the path will become quite different.

Therefore we propose a new representation of the path: we will store $\langle \delta\Phi_i \rangle_{i=0}^{48}$. The ∂Q_i's can be efficiently computed from the $\delta\Phi_i$'s, even if there is a little loss of information: a given $\langle \delta\Phi_i \rangle_{i=0}^{48}$ can correspond to many $\langle \partial_i \rangle_{i=0}^{48}$ (for instance using different carry extensions), but the algorithm quickly finds a good one. The main advantage of this representation is that a local modification of $\delta\Phi_i$ will not modify the other $\delta\Phi_j$, and we recompute the full path $\langle \partial Q_i \rangle_{i=0}^{48}$. In fact, since $\partial\Phi_i = 0$ most of the time, this is a much better description of the path: it tells us where we have to do something unusual.

Overview of the algorithm. The function PATHSTEP will extend the path one step further, using the same ideas as the SC algorithm at step $i+4$. It assumes the ∂Q_j's and $\delta\Phi_j$'s are chosen for $j > i$. Then, for every possible choice of $\delta_{i+4}^{\lll}$, it will compute

δQ_i from ∂Q_{i+4} and $\partial \Phi_{i+4}$ and add the $\lll$-conditions and Φ-conditions. It will have to choose a $\partial \Phi_{i+4}$ matching $\delta \Phi_{i+4}$ that is feasible given ∂Q_{i+1}, ∂Q_{i+2}, and ∂Q_{i+3}; if none is available, this branch of the search is aborted. Here we will also set $\delta \Phi_i$ to the value it had in the path P_0, so that the new path is similar the old one.

The function PATCHTARGET will then modify $\partial \Phi_i$ so as to remove some unwanted differences in the IV (trying to turn a pseudo-collision path into a collision path).

To finish the step i, the function PATCHCARRIES will select a ∂Q_i corresponding to δQ_i, and will extend some carries according to the values $\delta \Phi_{i+1}$, $\delta \Phi_{i+2}$ and $\delta \Phi_{i+3}$. This step is important because we need a non-zero bit in a ∂Q_{j-1}, ∂Q_{j-2} or ∂Q_{j-3} for every non-zero bit in $\partial \Phi_j$. Then it will add the ∂-conditions.

Correcting Differences. The critical part of the algorithm is the computation of the bits to modify in step i so as to correct a difference in the IV. To change directly a bit $Q_{i_0}^{[k]}$, we will set a non-zero difference in $\Phi_{i_0}^{[k \boxminus s_{i_0}]}$. However, we detect the differences in the IV, and we can't fix them here; we will have to act on a different step and see how the difference evolves. The simplest way to do so is to keep $\delta \Phi_i$ unmodified in the rest of the path, which is possible if the difference is absorbed by the Φ_i's. So we will try to use bit $Q_{i_0+4}^{[k \boxplus s_{i_0}]}$ to modify bit $Q_{i_0}^{[k]}$, and so on until we find a bit of Q_{i_0+4t} which can be changed using Φ.

When such a modification succeeds, it will remove one difference in the IV. This simple correction method is already useful: it finds the path from [23], but not the one from [19].

Indirect Correction. While searching for more complex paths, we will have some differences in the IV which cannot be dealt with this way. So we will introduce a difference which will not directly cancel the difference in the IV, but which will allow us to remove the target difference using the previous method. More precisely, to fix $Q_{i_0}^{[k]}$, we want a difference in some Q_{i_0+4t}, but we need a difference in the inputs of Φ_{i_0+4t}; so we will try to introduce a difference in Q_{i_0+4t+a}, where $a \in \{1, 2, 3\}$, and this will use $\Phi_{i_0+4t+a+4t'}$. See Algorithm 2 for a pseudo-code description.

When this succeeds, it removes the target difference, but it introduces a new unwanted difference. Hopefully, we may remove this new difference without indirect modifications... This method works rather well, and finds many paths using the message difference from [19].

Impossible paths. As we compute the differential path and the sufficient conditions at the same time, we do not have to deal with impossible path, during the execution of the algorithm: if a modification of the paths leads to an impossibility, we abort the search and look for other modifications. However, if the path with $\delta \Phi_i = 0$ is impossible – and this is the case if there are some differences in the third round[1] – the first pass of the

[1] for Wang's EUROCRYPT path, the differences in the third round form a local collisions, so we can as well run the algorithm only for the first two rounds.

Algorithm 2 Details on the bit correcting part of the algorithm

function PatchTarget(P_0,P,i)
  for all $Q_{i_0}^{[k]}$ bit to fix in P_0 **do** ▷ we try every difference, one by one
    PatchTargetBit(P_0,P,i,i_0,k,η_0)
function PatchTargetBit(P_0,P,i,i_0,k,η) ▷ η indirect modifications allowed
  if $i < i_0$ **then return**
  else if $i = i_0$ **then**
    modify P on bit k of step i
    PatchCarries(P_0, P, i) ▷ next step of the algorithm
  else
    PatchTargetBit($P_0, P, i, i_0 + 4, k + s_{i_0} \bmod 32, \eta$) ▷ Direct correction
  if $\eta > 0$ **then**
    modify P_0 on bit k of step i_0 ▷ Indirect correction
    for $a \in \{1, 2, 3\}$ **do** PatchTargetBit(P_0,P,i,$i_0 + a$,k,$\eta - 1$)

algorithm will abort with an incomplete path. Therefore we also add incomplete paths to the set $\mathcal{P}$, and we correct their errors in the same ways we correct differences in the IV.

Exploring the search space. In order to avoid spending too much time on uninteresting paths, we have to choose an interesting path in the set $\mathcal{P}$. As the indirect corrections are much expensive that direct ones, we only search for them on paths that have already been run without indirect corrections, and we favour runs without indirect corrections. We implemented the set $\mathcal{P}$ as a priority queue, and our priority function is based on:

- the number of difference in the IV
- the number of conditions
- the number of indirect correction allowed
- the depth in the tree (*ie.* the number of run between the first path and the current path)

To restrict the search space, we also set some limits on the path we are looking for. The difficulty here is to keep enough paths to find the good ones, while cutting enough branches in the search tree to finish in reasonable time. In our algorithm, we limit the size of the carries, and the number of total conditions in a path. We also limit the total number of runs of the algorithm, which allows to keep the set $\mathcal{P}$ to a fixed size.

Example. See Table 2 in Appendix B for an example of how the algorithm modifies the paths until it has no IV difference. For this path, there is no need for indirect correction.

3.4 Comparison with Existing Algorithms

Schläffer and Oswald [16]. Our algorithm bears some similarity with the algorithm of Schläffer and Oswald's (*eg.* the computation of the bits to modify when we have some difference in the IV is very similar to their computation of target differences), but we think the basic idea that rules the algorithm is not the same. Schläffer and Oswald basically try to cancel the differences introduced in the message, while we basically try to compute Q_i for $Q_{i+1}...Q_{i+4}$. Our approach computes the path and the sufficient conditions at the same time, whereas Schläffer and Oswald performed these two steps separately, and had to deal with impossible paths. A more important innovation from our algorithm comes from the possibility of indirect corrections: Oswald and Schläffer had to manually introduce *disturbance difference* which seemed to play the same role. It seems that the general structure of their algorithm is not well suited to automate this part. As a result our algorithm finds better ways to choose the indirect corrections, which results in a much better path.

De Cannière and Rechberger [2]. This work introduces some important new ideas, and some of them could be used to enhance our algorithm (eg. the generalised differential ∇). However their algorithm as such does not seem really suitable for MD4. It seems that they are only doing local modification to the path, and they can't correct a difference far from were it was introduced. This feature is well adapted to MD5, SHA-0 and SHA-1 because the step update function will duplicate a difference in the internal state. In MD4, a difference can be absorbed by the Φ_i's, and we can correct it many steps further. Furthermore, the basic idea of iteratively adding conditions seems incompatible with our indirect modification scheme.

Sasaki [15]. Sasaki introduces an interesting idea in FSE '07 [15]: he combines forward search and backward search with a meet-in-the-middle approach. We believe this can be adapted to our algorithm but we didn't had the time to do it yet. More importantly, Sasaki introduces a new message difference and a correspond path. Our algorithm does not work yet with this message difference, but we are working on it.

4 Applications

The algorithm was implemented in the C language, and we ran it with different message differences on a desktop computer. We used it to check the paths given by Wang *et al.* in [19] and [23].

4.1 Yu *et al.*'s CANS Path [23]

By applying our algorithm to Yu *et al.*'s CANS path [23], we found that

- This path is rather easy to find and does not require any indirect modifications. Our algorithm finds it in about 0.1 s.
- In [23], the authors claim that the path can be rotated and gives 32 similar paths using a message difference on the different bits of Q_4, but only 28 paths are actually correct[2].

[2] it fails if the difference is on bit 17,20,26 or 28.

– If the difference is applied to bit 25 instead of bit 22, the path has only 58 conditions instead of 62. This is good news for applications where one only needs one path with the smallest possible number of conditions, such as attacks against NMAC-MD4 [7,4].

4.2 Wang's EUROCRYPT Path [19]

We also ran our algorithm with the message difference of Wang's EUROCRYPT path [19], and we found many paths with less conditions; the two best are detailed in Path 1 and Path 2. These paths are much harder to find: they need some indirect modifications, and our algorithm takes a few hours to find them (however, a first solution is found in a few minutes, and it already has only 19 conditions on the second round). Our path is also better than the one found by Oswald and Schläffer, see Table 1 for a quick comparison. The number of conditions in a path determines the complexity of the collisions finding phase: conditions in the first round cost almost nothing (because message modification in the first round always succeeds); in the beginning of the second round, they cost a little bit more; and in the end of the second round and in the last round they can only be fulfilled statistically, so they have an exponential cost.

Table 1: Comparison of paths using the same message difference

Number of conditions	round 1	round 2	round 3	total
With Wang's message difference:				
Wang *et al.*'s path [19]	96	25	2	123
Schläffer and Oswald's path [16]	122	22	2	146
Our path	72	16	2	90
With Sasaki's message difference:				
Sasaki *et al.*'s path [15]	167	9	1	177

As far as MD4 collisions are concerned, the best path currently known is due do Sasaki *et al.* [15] and uses another message difference. Unfortunately, we have not yet been able to make our algorithm work with this message difference.

4.3 IV-dependent differential path

Using the DP algorithm, we can search for paths with message differences on the first message word m_0, which is used in the first step of the compression function. In this case, the Φ-conditions for the first step will involve Q_{-1} and Q_{-2}, that is, the IV. This kind of IV-dependent path can be used to recover some bits of the IV: if the condition on the IV is fulfilled, we will find collisions if we try enough message pairs with the prescribed difference; but if the condition is not fulfilled, that will not happen. This gives us a distinguisher which learns one bit of the IV. If we can find enough paths, and we do not need to try too many messages before a collision is found, we can then do an exhaustive search over the remaining IV bits, and we can easily check the validity of the IV using the collisions found.

IV-dependent differential paths can be used to attack some MAC algorithms, in particular NMAC/HMAC.

Our algorithm found 22 IV-dependent paths with a one-bit difference $\Delta_0 = 2^k$. Path 4 in Appendix B shows one of them with $k = 0$, and the other ones are obtained with a bit rotation of the whole path. They have one condition on the IV: $Q_{-1}^{[k \boxplus s_0]} = Q_{-2}^{[k \boxplus s_0]}$, and 79 conditions on the other internal state variables.

Conclusion and outlook

Our algorithm is successful at finding differential paths with some given message differential. Our paths have fewer conditions than the previously known ones, which shows that our algorithm is efficient. Good paths are not really needed for collision search, since collisions are already very cheap, but we believe that new kinds of attack against MD4 or MD4-based constructions could be found thanks to this algorithm. New differential paths could led to new attacks.

We are trying to explore what can be done with various differential paths, and we have already found a full key-recovery attack against NMAC-MD4 based on IV-dependent path.

References

1. John Black, Martin Cochran, and Trevor Highland. A Study of the MD5 Attacks: Insights and Improvements. In Robshaw [12], pages 262–277.
2. Christophe De Cannière and Christian Rechberger. Finding SHA-1 Characteristics: General Results and Applications . In Lai and Chen [10].
3. Anne Canteaut, editor. Ongoing research areas in symmetric cryptography. Technical Report D.STVL.3, ECRYPT, 2005. `http://www.ecrypt.eu.org/documents/D.STVL.3-2.5.pdf`.
4. Scott Contini and Yiqun Lisa Yin. Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In Lai and Chen [10].
5. 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*. Springer, 2005.
6. M. Daum. *Cryptanalysis of Hash Functions of the MD4-Family*. PhD thesis, Ruhr-University of Bochum, 2005.
7. Jongsung Kim, Alex Biryukov, Bart Preneel, and Seokhie Hong. On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In Roberto De Prisco and Moti Yung, editors, *SCN*, volume 4116 of *Lecture Notes in Computer Science*, pages 242–256. Springer, 2006.
8. Vlastimil Klima. Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. Cryptology ePrint Archive, Report 2005/102, 2005. `http://eprint.iacr.org/`.
9. Vlastimil Klima. Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006/105, 2006. `http://eprint.iacr.org/`.
10. Xuejia Lai and Kefei Chen, editors. *12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006.* , volume 4284 of *Lecture Notes in Computer Science*. Springer, 2006.
11. Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Attack on MD4 with Probability Almost 1. In Dongho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2005.

12. Matthew Robshaw, editor. *Fast Software Encryption: 13th InternationalWorkshop, FSE 2006 Graz, Austria, March 15-17, 2006 Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*. Springer, 2006.
13. Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. How to Construct Sufficient Condition in Searching Collisions of MD5. In Phong Q Nguyen, editor, *VIETCRYPT*, volume 4341 of *Lecture Notes in Computer Science*. Springer, 2006.
14. Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. Extended APOP Password Recovery Attack. Presented at the rump session of FSE '07. `http://fse2007.uni.lu/rump.html`.
15. Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New Message Difference for MD4, 2007. To appear in FSE '07 proceedings.
16. Martin Schläffer and Elisabeth Oswald. Searching for Differential Paths in MD4. In Robshaw [12], pages 242–261.
17. 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*. Springer, 2005.
18. Marc Stevens. Fast Collision Attack on MD5. Cryptology ePrint Archive, Report 2006/104, 2006. `http://eprint.iacr.org/`.
19. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Cramer [5], pages 1–18.
20. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Shoup [17], pages 17–36.
21. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Cramer [5], pages 19–35.
22. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Shoup [17], pages 1–16.
23. Hongbo Yu, Gaoli Wang, Guoyan Zhang, and Xiaoyun Wang. The Second-Preimage Attack on MD4. In Yvo Desmedt, Huaxiong Wang, Yi Mu, and Yongqing Li, editors, *CANS*, volume 3810 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2005.

A Interaction between modular difference and rotation

Let $a, b \in \mathbb{Z}_{2^{32}}$, $0 \leq s < 32$ and $\alpha = a \lll s, \beta = b \lll s$. We want to compute $\upsilon = \delta(\alpha, \beta)$ from $u = \delta(a, b)$ and a. We will use the integer addition $+$ (in $\mathbb{Z}$) and the modular addition $\boxplus$ (in $\mathbb{Z}_{2^{32}}$), and we express the rotation in the following way:

$$x \lll s = (2^s x \bmod 2^{32}) + \left\lfloor \frac{x \bmod 2^{32}}{2^{32-s}} \right\rfloor .$$

We will use the following result on integer part:

$$\lfloor x + y \rfloor = \begin{cases} \lfloor x \rfloor + \lfloor y \rfloor & \text{if } x + y < \lfloor x \rfloor + \lfloor y \rfloor + 1 \\ \lfloor x \rfloor + \lfloor y \rfloor + 1 & \text{if } x + y \geq \lfloor x \rfloor + \lfloor y \rfloor + 1 \end{cases}$$

If $a + u < 2^{32}$ (in $\mathbb{Z}$), then:

$$\begin{aligned}
\beta - \alpha &= \left(\left\lfloor \frac{a+u}{2^{32-s}} \right\rfloor + 2^s(\cancel{a} + u) \right) - \left(\left\lfloor \frac{a}{2^{32-s}} \right\rfloor + \cancel{2^s a} \right) \\
&= \left\lfloor \frac{a+u}{2^{32-s}} \right\rfloor - \left\lfloor \frac{a}{2^{32-s}} \right\rfloor + 2^s u \\
&= \left\lfloor \frac{u}{2^{32-s}} \right\rfloor + 2^s u \quad \text{or} \quad \left\lfloor \frac{u}{2^{32-s}} \right\rfloor + 2^s u + 1 \\
\beta \boxminus \alpha &= u \lll s \quad \text{or} \quad (u \lll s) \boxplus 1
\end{aligned}$$

Otherwise, $2^{32} \le a + u < 2^{33}$:

$$\begin{aligned}
\beta - \alpha &= \left(\left\lfloor \frac{a+u-2^{32}}{2^{32-s}} \right\rfloor + 2^s(a + u) \right) - \left(\left\lfloor \frac{a}{2^{32-s}} \right\rfloor + 2^s a \right) \\
&= -2^s + \left(\left\lfloor \frac{a+u}{2^{32-s}} \right\rfloor + 2^s(a + u) \right) - \left(\left\lfloor \frac{a}{2^{32-s}} \right\rfloor + 2^s a \right) \\
\beta \boxminus \alpha &= (u \lll s) \boxminus 2^s \quad \text{or} \quad (u \lll s) \boxminus 2^s \boxplus 1
\end{aligned}$$

Furthermore, we can explicit precise conditions for each case:

$$\upsilon = \begin{cases}
\upsilon_1 = (u \lll s) & \text{if } a + u < 2^{32} \text{ and} \\
 & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) < 2^{32-s} \\
\upsilon_2 = (u \lll s) \boxplus 1 & \text{if } a + u < 2^{32} \text{ and} \\
 & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) \ge 2^{32-s} \\
\upsilon_3 = (u \lll s) \boxminus 2^s & \text{if } a + u \ge 2^{32} \text{ and} \\
 & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) < 2^{32-s} \\
\upsilon_4 = (u \lll s) \boxminus 2^s \boxplus 1 & \text{if } a + u \ge 2^{32} \text{ and} \\
 & (a \bmod 2^{32-s}) + (u \bmod 2^{32-s}) \ge 2^{32-s}
\end{cases}$$

B Differential Paths

All the paths given in this section will use the notations defined in the article. Moreover, there are two extra differences with Wang's tables:

- The ∂-conditions are not included in the table, since they can be easily be deduced from ∂_i (eg. $Q_1^{[6]} = 1$ in step 1 of Path 1).
- The Φ-conditions and $\lll$-conditions are listed in the step were they are needed, rather than in the step were the message modification will be done. This makes the paths easier to read, but must be taken into account if one wants to count the conditions in each round.

Path 1: First of the two best paths we found with the same message difference as [19].

step	s_i	δm_i	$\partial \Phi_i$	∂Q_i	Φ-conditions and $\lll$-conditions
0	3				
1	7	$\langle \blacktriangle^{[31]} \rangle$		$\langle \blacktriangledown^{[6]} \rangle$	
2	11	$\langle \blacktriangledown^{[28]}, \blacktriangle^{[31]} \rangle$		$\langle \blacktriangle\blacktriangle\blacktriangle^{[7\ldots9]} \rangle$	$Q_0^{[6]} = Q_{-1}^{[6]}$
3	19				$Q_2^{[6]} = 0,\ Q_1^{[7]} = Q_0^{[7]},\ Q_1^{[8]} = Q_0^{[8]},\ Q_1^{[9]} = Q_0^{[9]}$
4	3		$\langle \blacktriangledown^{[6]} \rangle$	$\langle \blacktriangle\blacktriangledown^{[9,10]} \rangle$	$Q_3^{[6]} = 0,\ Q_3^{[7]} = 0,\ Q_3^{[8]} = 0,\ Q_3^{[9]} = 0$
5	7		$\langle \blacktriangle^{[7]} \rangle$	$\langle \blacktriangle^{[13]} \rangle$	$Q_4^{[7]} = 0,\ Q_4^{[8]} = 1,\ Q_3^{[9]} = 0,\ Q_3^{[10]} = Q_2^{[10]}$
6	11		$\langle \blacktriangledown^{[10]} \rangle$	$\langle \blacktriangledown^{[18]} \rangle$	$Q_5^{[9]} = 0,\ Q_5^{[10]} = 1,\ Q_4^{[13]} = Q_3^{[13]}$
7	19				$Q_6^{[9]} = 1,\ Q_6^{[10]} = 1,\ Q_6^{[13]} = 0,\ Q_5^{[18]} = Q_4^{[18]}$
8	3		$\langle \blacktriangle^{[13]} \rangle$	$\langle \blacktriangledown^{[12]}, \blacktriangledown\blacktriangle^{[16,17]} \rangle$	$Q_7^{[13]} = 0,\ Q_7^{[18]} = 0$
9	7		$\langle \blacktriangledown^{[12]} \rangle$	$\langle \blacktriangle^{[19]} \rangle$	$Q_7^{[12]} = 1,\ Q_6^{[12]} = 0,\ Q_7^{[16]} = Q_6^{[16]},\ Q_7^{[17]} = Q_6^{[17]},\ Q_8^{[18]} = 1$
10	11		$\langle \blacktriangle^{[17]} \rangle$	$\langle \blacktriangledown^{[28]} \rangle$	$Q_9^{[12]} = 0,\ Q_9^{[16]} = 0,\ Q_9^{[17]} = 1,\ Q_8^{[19]} = Q_7^{[19]}$
11	19				$Q_{10}^{[12]} = 1,\ Q_{10}^{[16]} = 1,\ Q_{10}^{[17]} = 1,\ Q_{10}^{[19]} = 0,\ Q_9^{[28]} = Q_8^{[28]}$
12	3	$\langle \blacktriangledown^{[16]} \rangle$	$\langle \blacktriangle^{[19]} \rangle$	$\langle \blacktriangledown^{[15]}, \blacktriangle^{[22]} \rangle$	$Q_{11}^{[19]} = 0,\ Q_{11}^{[28]} = 0$
13	7			$\langle \blacktriangledown\blacktriangledown\blacktriangle^{[26\ldots28]} \rangle$	$Q_{11}^{[15]} = Q_{10}^{[15]},\ Q_{11}^{[22]} = Q_{10}^{[22]},\ Q_{12}^{[28]} = 1$
14	11		$\langle \blacktriangle^{[28]} \rangle$		$Q_{13}^{[15]} = 0,\ Q_{13}^{[22]} = 0,\ Q_{12}^{[26]} = Q_{11}^{[26]},\ Q_{12}^{[27]} = Q_{11}^{[27]},\ Q_{12}^{[28]} = 1,\ Q_{11}^{[28]} = 0$
15	19		$\langle \blacktriangle^{[28]} \rangle$	$\langle \blacktriangle^{[15]} \rangle$	$Q_{14}^{[15]} = 1,\ Q_{14}^{[22]} = 1,\ Q_{14}^{[26]} = 0,\ Q_{14}^{[27]} = 0,\ Q_{14}^{[28]} = 1$
16	3		$\langle \blacktriangle^{[15]} \rangle$	$\langle \blacktriangle^{[25]} \rangle$	$Q_{14}^{[15]} \neq Q_{13}^{[15]},\ Q_{15}^{[26]} = Q_{14}^{[26]},\ Q_{15}^{[27]} = Q_{14}^{[27]},\ Q_{15}^{[28]} = Q_{14}^{[28]}$
17	5			$\langle \blacktriangle^{[31]} \rangle$	$Q_{16}^{[15]} = Q_{14}^{[15]},\ Q_{15}^{[25]} = Q_{14}^{[25]}$
18	9				$Q_{17}^{[15]} = Q_{16}^{[15]},\ Q_{17}^{[25]} = Q_{15}^{[25]},\ Q_{16}^{[31]} = Q_{15}^{[31]}$
19	13	$\langle \blacktriangledown^{[16]} \rangle$		$\langle \blacktriangledown^{[28]} \rangle$	$Q_{18}^{[25]} = Q_{17}^{[25]},\ Q_{18}^{[31]} = Q_{16}^{[31]}$
20	3	$\langle \blacktriangle^{[31]} \rangle$	$\langle \blacktriangledown^{[28]}, \blacktriangle^{[31]} \rangle$	$\langle \blacktriangle^{[28]}, \blacktriangledown^{[31]} \rangle$	$Q_{18}^{[28]} \neq Q_{17}^{[28]},\ Q_{19}^{[31]} \neq Q_{18}^{[31]}$
21	5		$\langle \blacktriangledown^{[31]} \rangle$		$Q_{19}^{[31]} \neq Q_{18}^{[31]}$
22	9				$Q_{21}^{[31]} = Q_{19}^{[31]}$
23	13		$\langle \blacktriangle^{[28]} \rangle$		$Q_{22}^{[28]} \neq Q_{21}^{[28]},\ Q_{22}^{[31]} = Q_{21}^{[31]}$
24	3	$\langle \blacktriangledown^{[28]}, \blacktriangle^{[31]} \rangle$			
25	5				
26	9				
27	13				
28	3				
29	5				
30	9				
31	13				
32	3				
33	9				
34	11				
35	15	$\langle \blacktriangledown^{[16]} \rangle$		$\langle \blacktriangledown^{[31]} \rangle$	
36	3	$\langle \blacktriangledown^{[28]}, \blacktriangle^{[31]} \rangle$	$\langle \blacktriangledown^{[31]} \rangle$	$\langle \blacktriangledown^{[31]} \rangle$	
37	9				
38	11				
39	15		$\langle \blacktriangle^{[31]} \rangle$		
40	3	$\langle \blacktriangle^{[31]} \rangle$			
41	9				
42	11				
43	15				
44	3				
45	9				
46	11				
47	15				

90 conditions: $72 + 16 + 2$

Path 2: Second of the two best paths found with the same message difference as [19].

step	s_i	δm_i	$\partial\Phi_i$	∂Q_i	Φ-conditions and $\lll$-conditions
0	3				
1	7	$\langle\blacktriangle^{[31]}\rangle$		$\langle\blacktriangle^{[6]}\rangle$	
2	11	$\langle\blacktriangledown^{[28]},\blacktriangle^{[31]}\rangle$		$\langle\blacktriangledown^{[7]},\blacktriangle^{[10]}\rangle$	$Q_0^{[6]} = Q_{-1}^{[6]}$
3	19				$Q_2^{[6]} = 0,\ Q_1^{[7]} = Q_0^{[7]},\ Q_1^{[10]} = Q_0^{[10]}$
4	3		$\langle\blacktriangle\blacktriangledown^{[6,7]}\rangle$	$\langle\blacktriangle\blacktriangle\blacktriangledown^{[9\ldots11]}\rangle$	$Q_3^{[6]} = 0,\ Q_3^{[7]} = 1,\ Q_3^{[10]} = 0$
5	7			$\langle\blacktriangle^{[13]}\rangle$	$Q_4^{[7]} = 1,\ Q_3^{[9]} = Q_2^{[9]},\ Q_3^{[10]} = 0,\ Q_3^{[11]} = Q_2^{[11]}$
6	11		$\langle\blacktriangle\blacktriangledown^{[10,11]}\rangle$	$\langle\blacktriangledown^{[18]}\rangle$	$Q_5^{[9]} = 0,\ Q_5^{[10]} = 1,\ Q_5^{[11]} = 1,\ Q_4^{[13]} = Q_3^{[13]}$
7	19				$Q_6^{[9]} = 1,\ Q_6^{[10]} = 1,\ Q_6^{[11]} = 1,\ Q_6^{[13]} = 0,\ Q_5^{[18]} = Q_4^{[18]}$
8	3		$\langle\blacktriangle^{[13]}\rangle$	$\langle\blacktriangledown^{[12]},\blacktriangle^{[16]}\rangle$	$Q_7^{[13]} = 0,\ Q_7^{[18]} = 0$
9	7		$\langle\blacktriangledown^{[12]}\rangle$	$\langle\blacktriangle^{[19]}\rangle$	$Q_7^{[12]} = 1,\ Q_6^{[12]} = 0,\ Q_7^{[16]} = Q_6^{[16]},\ Q_8^{[18]} = 1$
10	11			$\langle\blacktriangledown^{[29]}\rangle$	$Q_9^{[12]} = 0,\ Q_9^{[16]} = 0,\ Q_8^{[19]} = Q_7^{[19]}$
11	19				$Q_{10}^{[12]} = 1,\ Q_{10}^{[16]} = 1,\ Q_{10}^{[19]} = 0,\ Q_9^{[29]} = Q_8^{[29]}$
12	3	$\langle\blacktriangledown^{[16]}\rangle$	$\langle\blacktriangle^{[19]}\rangle$	$\langle\blacktriangledown^{[15]},\blacktriangle^{[22]}\rangle$	$Q_{11}^{[19]} = 0,\ Q_{11}^{[29]} = 0$
13	7			$\langle\blacktriangledown\blacktriangledown\blacktriangledown\blacktriangle^{[26\ldots29]}\rangle$	$Q_{11}^{[15]} = Q_{10}^{[15]},\ Q_{11}^{[22]} = Q_{10}^{[22]},\ Q_{12}^{[29]} = 1$
14	11		$\langle\blacktriangle^{[29]}\rangle$		$Q_{13}^{[15]} = 0,\ Q_{13}^{[22]} = 0,\ Q_{12}^{[26]} = Q_{11}^{[26]},\ Q_{12}^{[27]} = Q_{11}^{[27]},\ Q_{12}^{[28]} = Q_{11}^{[28]},\ Q_{12}^{[29]} = 1,\ Q_{11}^{[29]} = 0$
15	19		$\langle\blacktriangledown\blacktriangle^{[28,29]}\rangle$	$\langle\blacktriangle^{[15]}\rangle$	$Q_{14}^{[15]} = 1,\ Q_{14}^{[22]} = 1,\ Q_{14}^{[26]} = 0,\ Q_{14}^{[27]} = 0,\ Q_{14}^{[28]} = 1,\ Q_{14}^{[29]} = 1$
16	3		$\langle\blacktriangle^{[15]}\rangle$	$\langle\blacktriangle^{[25]}\rangle$	$Q_{14}^{[15]} \neq Q_{13}^{[15]},\ Q_{15}^{[26]} = Q_{14}^{[26]},\ Q_{15}^{[27]} = Q_{14}^{[27]},\ Q_{15}^{[28]} = Q_{14}^{[28]},\ Q_{15}^{[29]} = Q_{14}^{[29]}$
17	5			$\langle\blacktriangle^{[31]}\rangle$	$Q_{16}^{[15]} = Q_{14}^{[15]},\ Q_{15}^{[25]} = Q_{14}^{[25]}$
18	9				$Q_{17}^{[15]} = Q_{16}^{[15]},\ Q_{17}^{[25]} = Q_{15}^{[25]},\ Q_{16}^{[31]} = Q_{15}^{[31]}$
19	13	$\langle\blacktriangledown^{[16]}\rangle$		$\langle\blacktriangledown^{[28]}\rangle$	$Q_{18}^{[25]} = Q_{17}^{[25]},\ Q_{18}^{[31]} = Q_{16}^{[31]}$
20	3	$\langle\blacktriangle^{[31]}\rangle$	$\langle\blacktriangledown^{[28]},\blacktriangle^{[31]}\rangle$	$\langle\blacktriangle^{[28]},\blacktriangledown^{[31]}\rangle$	$Q_{18}^{[28]} \neq Q_{17}^{[28]},\ Q_{19}^{[31]} \neq Q_{18}^{[31]}$
21	5		$\langle\blacktriangledown^{[31]}\rangle$		$Q_{19}^{[31]} \neq Q_{18}^{[31]}$
22	9				$Q_{21}^{[31]} = Q_{19}^{[31]}$
23	13		$\langle\blacktriangle^{[28]}\rangle$		$Q_{22}^{[28]} \neq Q_{21}^{[28]},\ Q_{22}^{[31]} = Q_{21}^{[31]}$
24	3	$\langle\blacktriangledown^{[28]},\blacktriangle^{[31]}\rangle$			
25	5				
26	9				
27	13				
28	3				
29	5				
30	9				
31	13				
32	3				
33	9				
34	11				
35	15	$\langle\blacktriangledown^{[16]}\rangle$		$\langle\blacktriangledown^{[31]}\rangle$	
36	3	$\langle\blacktriangledown^{[28]},\blacktriangle^{[31]}\rangle$	$\langle\blacktriangledown^{[31]}\rangle$	$\langle\blacktriangledown^{[31]}\rangle$	
37	9				
38	11				
39	15		$\langle\blacktriangle^{[31]}\rangle$		
40	3	$\langle\blacktriangle^{[31]}\rangle$			
41	9				
42	11				
43	15				
44	3				
45	9				
46	11				
47	15				

90 conditions: $72 + 16 + 2$

Path 3: Improved version of the path from Yu *et al.* [23].

step	s_i	δm_i	$\partial\Phi_i$	∂Q_i	conditions
0	3				
1	7				
2	11				
3	19				
4	3	$\langle\blacktriangle^{[25]}\rangle$		$\langle\blacktriangle^{[28]}\rangle$	
5	7				$Q_3^{[28]} = Q_2^{[28]}$
6	11				$Q_5^{[28]} = 0$
7	19				$Q_6^{[28]} = 1$
8	3			$\langle\blacktriangle^{[31]}\rangle$	
9	7				$Q_7^{[31]} = Q_6^{[31]}$
10	11		$\langle\blacktriangle^{[31]}\rangle$	$\langle\blacktriangledown^{[10]}\rangle$	$Q_9^{[31]} = 1$
11	19				$Q_9^{[10]} = Q_8^{[10]}$, $Q_{10}^{[31]} = 1$
12	3			$\langle\blacktriangle^{[2]}\rangle$	$Q_{11}^{[10]} = 0$
13	7				$Q_{11}^{[2]} = Q_{10}^{[2]}$, $Q_{12}^{[10]} = 1$
14	11			$\langle\blacktriangledown^{[21]}\rangle$	$Q_{13}^{[2]} = 0$
15	19				$Q_{14}^{[2]} = 1$, $Q_{13}^{[21]} = Q_{12}^{[21]}$
16	3			$\langle\blacktriangle^{[5]}\rangle$	$Q_{15}^{[21]} = Q_{13}^{[21]}$
17	5	$\langle\blacktriangle^{[25]}\rangle$	$\langle\blacktriangle^{[5]}\rangle$	$\langle\blacktriangle^{[10]},\blacktriangle^{[30]}\rangle$	$Q_{15}^{[5]} \neq Q_{14}^{[5]}$, $Q_{16}^{[21]} = Q_{15}^{[21]}$
18	9			$\langle\blacktriangledown^{[30]}\rangle$	$Q_{17}^{[5]} = Q_{15}^{[5]}$, $Q_{16}^{[10]} = Q_{15}^{[10]}$, $Q_{16}^{[30]} = Q_{15}^{[30]}$
19	13				$Q_{18}^{[5]} = Q_{17}^{[5]}$, $Q_{18}^{[10]} = Q_{16}^{[10]}$
20	3			$\langle\blacktriangle^{[8]}\rangle$	$Q_{19}^{[10]} = Q_{18}^{[10]}$
21	5		$\langle\blacktriangledown^{[30]}\rangle$	$\langle\blacktriangle^{[15]}\rangle$	$Q_{19}^{[8]} = Q_{18}^{[8]}$, $Q_{20}^{[30]} \neq Q_{19}^{[30]}$
22	9			$\langle\blacktriangle\blacktriangledown^{[7,8]}\rangle$	$Q_{21}^{[8]} = Q_{19}^{[8]}$, $Q_{20}^{[15]} = Q_{19}^{[15]}$
23	13				$Q_{21}^{[7]} = Q_{20}^{[7]}$, $Q_{22}^{[15]} = Q_{20}^{[15]}$
24	3		$\langle\blacktriangledown^{[8]}\rangle$		$Q_{23}^{[7]} = Q_{21}^{[7]}$, $Q_{23}^{[8]} \neq Q_{21}^{[8]}$, $Q_{23}^{[15]} = Q_{22}^{[15]}$
25	5			$\langle\blacktriangle^{[20]}\rangle$	$Q_{24}^{[7]} = Q_{23}^{[7]}$, $Q_{24}^{[8]} = Q_{23}^{[8]}$
26	9			$\langle\blacktriangledown^{[16]}\rangle$	$Q_{24}^{[20]} = Q_{23}^{[20]}$
27	13				$Q_{25}^{[16]} = Q_{24}^{[16]}$, $Q_{26}^{[20]} = Q_{24}^{[20]}$
28	3				$Q_{27}^{[16]} = Q_{25}^{[16]}$, $Q_{27}^{[20]} = Q_{26}^{[20]}$
29	5			$\langle\blacktriangle^{[25]}\rangle$	$Q_{28}^{[16]} = Q_{27}^{[16]}$
30	9			$\langle\blacktriangledown^{[25]}\rangle$	$Q_{28}^{[25]} = Q_{27}^{[25]}$
31	13				
32	3				
33	9		$\langle\blacktriangledown^{[25]}\rangle$		$Q_{32}^{[25]} = Q_{31}^{[25]}$
34	11	$\langle\blacktriangle^{[25]}\rangle$			
35	15				
36	3				
37	9				
38	11				
39	15				
40	3				
41	9				
42	11				
43	15				
44	3				
45	9				
46	11				
47	15				

58 conditions: $20 + 37 + 1$

Path 4: An IV-dependent path with the message difference on the first word.

step	s_i	δm_i	$\partial\Phi_i$	∂Q_i	Φ-conditions and $\lll$-conditions
0	3	$\langle\blacktriangle^{[0]}\rangle$		$\langle\blacktriangle^{[3]}\rangle$	
1	7				$Q^{[3]}_{-1} = Q^{[3]}_{-2}$
2	11				$Q^{[3]}_{1} = 0$
3	19				$Q^{[3]}_{2} = 1$
4	3			$\langle\blacktriangledown\blacktriangle^{[6,7]}\rangle$	
5	7				$Q^{[6]}_{3} = Q^{[6]}_{2}$, $Q^{[7]}_{3} = Q^{[7]}_{2}$
6	11				$Q^{[6]}_{5} = 0$, $Q^{[7]}_{5} = 0$
7	19		$\langle\blacktriangle^{[7]}\rangle$	$\langle\blacktriangle^{[26]}\rangle$	$Q^{[6]}_{6} = 1$, $Q^{[7]}_{6} = 0$
8	3		$\langle\blacktriangledown^{[26]}\rangle$	$\langle\blacktriangle^{[9]}, \blacktriangledown^{[29]}\rangle$	$Q^{[26]}_{5} = 1$, $Q^{[26]}_{6} = 0$
9	7				$Q^{[9]}_{7} = Q^{[9]}_{6}$, $Q^{[26]}_{8} = 0$, $Q^{[29]}_{7} = Q^{[29]}_{6}$
10	11				$Q^{[9]}_{9} = 0$, $Q^{[26]}_{9} = 1$, $Q^{[29]}_{9} = 0$
11	19			$\langle\blacktriangle^{[13]}\rangle$	$Q^{[9]}_{10} = 1$, $Q^{[29]}_{10} = 1$
12	3			$\langle\blacktriangledown^{[0]}, \blacktriangle^{[12]}\rangle$	$Q^{[13]}_{10} = Q^{[13]}_{9}$
13	7				$Q^{[0]}_{11} = Q^{[0]}_{10}$, $Q^{[12]}_{11} = Q^{[12]}_{10}$, $Q^{[13]}_{12} = 0$
14	11		$\langle\blacktriangledown^{[0]}\rangle$	$\langle\blacktriangle\blacktriangle\blacktriangledown^{[11\ldots13]}\rangle$	$Q^{[0]}_{13} = 1$, $Q^{[12]}_{13} = 0$, $Q^{[13]}_{13} = 1$
15	19		$\langle\blacktriangledown^{[13]}\rangle$		$Q^{[0]}_{14} = 1$, $Q^{[11]}_{13} = Q^{[11]}_{12}$, $Q^{[12]}_{13} = 0$, $Q^{[13]}_{13} = 1$, $Q^{[13]}_{12} = 0$
16	3	$\langle\blacktriangle^{[0]}\rangle$	$\langle\blacktriangle\blacktriangledown^{[12,13]}\rangle$		$Q^{[11]}_{15} = Q^{[11]}_{13}$, $Q^{[12]}_{15} \neq Q^{[12]}_{13}$, $Q^{[13]}_{15} \neq Q^{[13]}_{13}$
17	5				$Q^{[11]}_{16} = Q^{[11]}_{15}$, $Q^{[12]}_{16} = Q^{[12]}_{15}$, $Q^{[13]}_{16} = Q^{[13]}_{15}$
18	9			$\langle\blacktriangle\blacktriangle\blacktriangle\blacktriangledown^{[20\ldots23]}\rangle$	
19	13				$Q^{[20]}_{17} = Q^{[20]}_{16}$, $Q^{[21]}_{17} = Q^{[21]}_{16}$, $Q^{[22]}_{17} = Q^{[22]}_{16}$, $Q^{[23]}_{17} = Q^{[23]}_{16}$
20	3		$\langle\blacktriangledown^{[23]}\rangle$	$\langle\blacktriangledown^{[26]}\rangle$	$Q^{[20]}_{19} = Q^{[20]}_{17}$, $Q^{[21]}_{19} = Q^{[21]}_{17}$, $Q^{[22]}_{19} = Q^{[22]}_{17}$, $Q^{[23]}_{19} \neq Q^{[23]}_{17}$
21	5				$Q^{[20]}_{20} = Q^{[20]}_{19}$, $Q^{[21]}_{20} = Q^{[21]}_{19}$, $Q^{[22]}_{20} = Q^{[22]}_{19}$, $Q^{[23]}_{20} = Q^{[23]}_{19}$, $Q^{[26]}_{19} = Q^{[26]}_{18}$
22	9			$\langle\blacktriangledown^{[29]}\rangle$	$Q^{[26]}_{21} = Q^{[26]}_{19}$
23	13				$Q^{[26]}_{22} = Q^{[26]}_{21}$, $Q^{[29]}_{21} = Q^{[29]}_{20}$
24	3			$\langle\blacktriangle\blacktriangledown^{[29,30]}\rangle$	$Q^{[29]}_{23} = Q^{[29]}_{21}$
25	5				$Q^{[30]}_{23} = Q^{[30]}_{22}$
26	9		$\langle\blacktriangle^{[29]}\rangle$		$Q^{[29]}_{25} \neq Q^{[29]}_{23}$, $Q^{[30]}_{25} = Q^{[30]}_{23}$
27	13				$Q^{[29]}_{26} = Q^{[29]}_{25}$, $Q^{[30]}_{26} = Q^{[30]}_{25}$
28	3			$\langle\blacktriangledown^{[0]}\rangle$	
29	5				$Q^{[0]}_{27} = Q^{[0]}_{26}$
30	9				$Q^{[0]}_{29} = Q^{[0]}_{27}$
31	13				$Q^{[0]}_{30} = Q^{[0]}_{29}$
32	3	$\langle\blacktriangle^{[0]}\rangle$			
33	9				
34	11				
35	15				
36	3				
37	9				
38	11				
39	15				
40	3				
41	9				
42	11				
43	15				
44	3				
45	9				
46	11				
47	15				

80 conditions: (1+) $42 + 37 + 0$

			Initial path		Path 1		Path 2		Path 3		Final Path	
step	s_i	δm_i	$\partial\Phi_i$	∂Q_i	$\partial\Phi_i$	∂Q_i	$\partial\Phi_i$	∂Q_i	$\partial\Phi_i$	∂Q_i	$\partial\Phi_i$	∂Q_i
4	3	⟨▲[22]⟩	⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩			⟨▲[25]⟩		⟨▲[25]⟩
5	7		⟨▼[1] ▲[13]⟩	⟨▼[8], ▲[20]⟩	⟨▲[13]⟩	⟨▲[20]⟩	⟨▲[13]⟩	⟨▲[20]⟩	⟨▲[13]⟩	⟨▲[20]⟩		
6	11		⟨▼[17]⟩	⟨▼[28]⟩	⟨▼[17]⟩	⟨▼[28]⟩						
7	19											
8	3									⟨▼▲[28,29]⟩		⟨▼▲[28,29]⟩
9	7			⟨▼[15], ▲[27]⟩		⟨▲[27]⟩		⟨▼▲[27,28]⟩		⟨▲[27]⟩		
10	11			⟨▼[7]⟩		⟨▼[7]⟩	⟨▼[28]⟩	⟨▼[7]⟩	⟨▼[28]⟩	⟨▼[7]⟩	⟨▼[28]⟩	⟨▼[7]⟩
11	19											
12	3									⟨▲[31]⟩		⟨▲[31]⟩
13	7			⟨▲[2], ▼[22]⟩		⟨▲[2]⟩		⟨▲[2]⟩		⟨▲[2]⟩		
14	11			⟨▼[18]⟩		⟨▼[18]⟩		⟨▼[18]⟩		⟨▼[18]⟩		⟨▼[18]⟩
15	19											
16	3									⟨▲[2]⟩		⟨▲[2]⟩
17	5	⟨▲[22]⟩		⟨▲[7]⟩		⟨▲[7], ▲[27]⟩		⟨▲[7], ▲[27]⟩		⟨▲[7], ▲[27]⟩	⟨▲[2]⟩	⟨▲[7], ▲[27]⟩
18	9			⟨▼[27]⟩		⟨▼[27]⟩		⟨▼[27]⟩		⟨▼[27]⟩		⟨▼[27]⟩
19	13											
20	3									⟨▲[5]⟩		⟨▲[5]⟩
21	5			⟨▲[12]⟩	⟨▼[27]⟩	⟨▲[12]⟩	⟨▼[27]⟩	⟨▲[12]⟩	⟨▼[27]⟩	⟨▲[12]⟩	⟨▼[27]⟩	⟨▲[12]⟩
22	9			⟨▼[4]⟩		⟨▼[4]⟩		⟨▼[4]⟩		⟨▲▼[4,5]⟩		⟨▲▼[4,5]⟩
23	13											
24	3								⟨▼[5]⟩		⟨▼[5]⟩	
25	5			⟨▲[17]⟩		⟨▲[17]⟩		⟨▲[17]⟩		⟨▲[17]⟩		⟨▲[17]⟩
26	9			⟨▼[13]⟩		⟨▼[13]⟩		⟨▼[13]⟩		⟨▼[13]⟩		⟨▼[13]⟩
27	13											
28	3											
29	5			⟨▲[22]⟩		⟨▲[22]⟩		⟨▲[22]⟩		⟨▲[22]⟩		⟨▲[22]⟩
30	9			⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩
31	13											
32	3											
33	9		⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩		⟨▼[22]⟩	
34	11	⟨▲[22]⟩										

Table 2: Example run on the path from [19]. We show intermediate steps to ease comprehension, but the algorithm actually finds final directly from the original one.

C Conditions for Φ

Table 3: Conditions to get the right output difference in Φ_i.
✓: no extra conditions are needed; ✗: input and output requirements are incompatible.

	$F(x,y,z)=\mathrm{IF}(x,y,z)$			$G(x,y,z)=\mathrm{MAJ}(x,y,z)$			$H(x,y,z)=x\oplus y\oplus z$			$I(x,y,z)=y\oplus(x\vee\neg z)$		
$\partial x\ \partial y\ \partial z$	$\partial F=0$	$\partial F=1$	$\partial F=-1$	$\partial G=0$	$\partial G=1$	$\partial G=-1$	$\partial H=0$	$\partial H=1$	$\partial H=-1$	$\partial I=0$	$\partial I=1$	$\partial I=-1$
$0\ 0\ 0$	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗
$0\ 0\ {+1}$	$x=1$	$x=0$	✗	$x=y$	$x\neq y$	✗	✗	$x=y$	$x\neq y$	$x=1$	$x,y=0,1$	$x,y=0,0$
$0\ 0\ {-1}$	$x=1$	✗	$x=0$	$x=y$	✗	$x\neq y$	✗	$x\neq y$	$x=y$	$x=1$	$x,y=0,0$	$x,y=0,1$
$0\ {+1}\ 0$	$x=0$	$x=1$	✗	$x=z$	$x\neq z$	✗	✗	$x=z$	$x\neq z$	✗	$x,z=0,1$	$x,y\neq 1,0$
$0\ {-1}\ 0$	$x=0$	✗	$x=1$	$x=z$	✗	$x\neq z$	✗	$x\neq z$	$x=z$	✗	$x,y\neq 1,0$	$x,z=0,1$
${+1}\ 0\ 0$	$y=z$	$y,z=1,0$	$y,z=0,1$	$y=z$	$y\neq z$	✗	✗	$y=z$	$y\neq z$	$z=0$	$y,z=0,1$	$y,z=1,1$
${-1}\ 0\ 0$	$y=z$	$y,z=0,1$	$y,z=1,0$	$y=z$	✗	$y\neq z$	✗	$y\neq z$	$y=z$	$z=0$	$y,z=1,1$	$y,z=0,1$
$0\ {+1}\ {+1}$	✗	✓	✗	✗	✓	✗	✓	✗	✗	$x=0$	✗	$x=1$
$0\ {-1}\ {+1}$	✗	$x=0$	$x=1$	✓	✗	✗	✓	✗	✗	$x=0$	$x=1$	✗
$0\ {+1}\ {-1}$	✗	$x=1$	$x=0$	✓	✗	✗	✓	✗	✗	$x=0$	✗	$x=1$
$0\ {-1}\ {-1}$	✗	✗	✓	✗	✗	✓	✓	✗	✗	$x=0$	$x=1$	✗
${+1}\ 0\ {+1}$	$y=0$	$y=1$	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
${-1}\ 0\ {+1}$	$y=1$	$y=0$	✗	✓	✗	✗	✓	✗	✗	✗	$y=1$	$y=0$
${+1}\ 0\ {-1}$	$y=1$	✗	$y=0$	✓	✗	✗	✓	✗	✗	✗	$y=0$	$y=1$
${-1}\ 0\ {-1}$	$y=0$	✗	$y=1$	✗	✗	✓	✓	✗	✗	✓	✗	✗
${+1}\ {+1}\ 0$	$z=1$	$z=0$	✗	✗	✓	✗	✓	✗	✗	$z=1$	✗	$z=0$
${-1}\ {+1}\ 0$	$z=0$	$z=1$	✗	✓	✗	✗	✓	✗	✗	$z=1$	✗	$z=0$
${+1}\ {-1}\ 0$	$z=0$	✗	$z=1$	✓	✗	✗	✓	✗	✗	$z=1$	$z=0$	✗
${-1}\ {-1}\ 0$	$z=1$	✗	$z=0$	✗	✗	✓	✓	✗	✗	$z=1$	$z=0$	✗
${+1}\ {+1}\ {+1}$	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓
${-1}\ {+1}\ {+1}$	✗	✓	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
${+1}\ {-1}\ {+1}$	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
${-1}\ {-1}\ {+1}$	✓	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗
${+1}\ {+1}\ {-1}$	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
${-1}\ {+1}\ {-1}$	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓
${+1}\ {-1}\ {-1}$	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗
${-1}\ {-1}\ {-1}$	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗