MD5 Is Weaker Than Weak: Attacks on Concatenated Combiners

Florian Mendel, Christian Rechberger, and Martin Schläffer

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
christian.rechberger@iaik.tugraz.at

Abstract. We consider a long standing problem in cryptanalysis: attacks on hash function combiners. In this paper, we propose the first attack that allows collision attacks on combiners with a runtime below the birthday-bound of the *smaller* compression function. This answers an open question by Joux posed in 2004.

As a concrete example we give such an attack on combiners with the widely used hash function MD5. The cryptanalytic technique we use combines a partial birthday phase with a differential inside-out technique, and may be of independent interest. This potentially reduces the effort for a collision attack on a combiner like MD5||SHA-1 for the first time.

Keywords: hash functions, cryptanalysis, MD5, combiner, differential.

1 Introduction

The recent spur of cryptanalytic results on popular hash functions like MD5 and SHA-1 [28,30,31] suggests that they are (much) weaker than originally anticipated, especially with respect to collision resistance. It seems non-trivial to propose a concrete hash function which inspires long term confidence. Even more so as we seem unable to construct collision resistant primitives from potentially simpler primitives [27]. Hence constructions that allow to hedge bets, like concatenated combiners, are of great interest. Before we give a preview of our results in the following, we will first review work on combiners.

Review of work on combiners. The goal of combiners is to have at least some bound on the expected security even if (some of the) hash functions get broken, for various definitions of "security" and "broken". Joux [12] showed (by using multi-collisions) that the collision resistance of a combiner can not be expected to be much higher than the birthday bound of the component (=hash function) with the largest output size.

On the other hand, combiners seem to be very robust when it comes to collision security up to the birthday bound (of the component with the smallest output size): By using techniques similar to Coron *et al.* [3], Hoch and Shamir [11] showed that only very mild assumptions on a compression function are needed to

M. Matsui (Ed.): ASIACRYPT 2009, LNCS 5912, pp. 144–161, 2009.

achieve a collision resistance of at least $O(2^{n/2})$. In fact, using a model proposed by Liskov [15], they show that none of the compression functions need to be collision, nor preimage resistant in the usual sense.

Motivation: cryptanalysis of combiners. Concatenating the output of hash function is often used by implementors to "hedge bets" on hash functions. A combiner of the form MD5||SHA-1 as used in SSL 3.0/TLS 1.0 and TLS 1.1 [7,8] is an example of such a strategy. Let's assume we are given a combiner of the form MD5||SHA-1. Let's further assume that a breakthrough in cryptanalysis of SHA-1 brings down the complexity of a collision search attack to 2^{52}. We know that the best collision search attacks on MD5 are as fast as 2^{15} [29]. So what is the best collision attack on the combiner? The best known method due to Joux is only as good as a birthday attack on the smaller of the two hash functions in the combiner. There is no known method which would allow to reduce the total effort below this bound, *i.e.* 2^{64}:

Currently, the best solution at our disposal is to combine the (hypothetic) SHA-1 attack with Joux's multicollision approach. Find a 2^{64}-multicollision for SHA-1 with effort $2^{52} \cdot 64 = 2^{58}$, and then perform a birthday-type search in this 2^{64} collision to single out a collision which also collides for MD5. The total effort will be 2^{64}. In fact, reductions of the effort for SHA-1 collision search will only marginally improve the attack on the combiner. How to improve upon this? Analyzing the combiner as a whole may by prohibitively complicated. The resistance of two-pipe designs with sufficiently different pipes like RIPEMD-160 [10] against recent collision search attacks also gives hints in this direction.

Preview of our results: We propose a new method that allows a cryptanalyst to focus on the hash functions individually while still potentially allowing attacks on combiners with a runtime below the birthday-bound of the smaller compression function. This also answers an open question by Joux posed in 2004 [12]. For this, we start with definitions in Section 2. In Section 3 we give a high-level description of our attack strategy on a concatenation combiner without going into the details of a particular compression function. Next, we consider as a concrete cryptanalytic example combiners that use MD5. We first give an alternative description of MD5 in Section 4, which will turn out to be beneficial (and in fact as our experiments suggest necessary) in Section 5, where we describe the cryptanalytic techniques we need, to be able to use the high-level attack description.

For the cryptanalysis, we employ a combination of a birthday-style attack and a differential inside-out technique that uses different parts of a collision characteristic at different stages of an attack, both before and after a birthday phase. The differential technique may be of independent interest, also for improving known types of collision attacks on MD5, or for finding one-block collisions. In Section 6 we give practical results which allow us to estimate the actual security MD5 is able to give in a combiner. Finally, we conclude and discuss open problem in Section 7.

2 Definitions

In the reminder of the paper we give a few definitions. We give a classification of collision attacks on compression functions and hash functions. Let an iterated hash function F be built by iterating a compression function $f : \{0,1\}^m \times \{0,1\}^n \rightarrow \{0,1\}^n$ as follows:

- Split the message M of arbitrary length into k blocks x_i of size m.
- Set h_0 to a pre-specified IV
- Compute $\forall x_i : h_i = f(h_{i-1}, x_i)$
- Output $F(M) = h_k$

Classification for compression function collision attacks. Higher numbers mean less degrees of freedom for an attacker and are hence more difficult to obtain cryptanalytically.

- **Compression collision attacks of type 0**
  Compute h_{i-1}, h^*_{i-1}, m_i and m^*_i s. t. $f(h_{i-1}, m_i) = f(h^*_{i-1}, m^*_i)$. Note that early attacks by den Boer and Bosselaers [1], and Dobbertin [9] on MD5 are of this type.
- **Compression collision attacks of type 1**
  Given h_{i-1}, compute m_i and m^*_i s. t. $f(h_{i-1}, m_i) = f(h^*_{i-1}, m^*_i)$.
- **Compression collision attacks of type 2**
  Given h_{i-1} and h^*_{i-1}, compute m_i and m^*_i s. t. $f(h_{i-1}, m_i) = f(h^*_{i-1}, m^*_i)$
- **Compression collision attacks of type 3**
  Given h_{i-1} and h^*_{i-1}, compute m_i s. t. $f(h_{i-1}, m_i) = f(h^*_{i-1}, m_i)$

Later in the paper, it will be useful to have a weakened version of the collision attack on the compression function of type 3.

- **Compression collision attacks of type 3w**
  Given h_{i-1} and h^*_{i-1} from an efficiently enumerable subset s (of size $|s| = 2^{n-z}$) of all 2^{2n} possible pairs (h_{i-1},h^*_{i-1}), compute m_i s. t. $f(h_{i-1}, m_i) = f(h^*_{i-1}, m_i)$.

Complementing types 1-3 of the compression function attacks, one may define similar attack settings for the hash function as well. For sake of concreteness, we also give examples related to MD5.

- **Hash collision attacks of type 1:** Given m_0, compute m_1 and m^*_1 such that $F(m_0||m_1) = F(m_0||m^*_1)$. This is the most simple way to violate the collision resistance of a hash function. For MD5, see Wang *et al.* [31]. The prefix m_0 may be the string of length 0, or any other message block.
- **Hash collision attacks of type 2:** Given m_0 and m^*_0, compute m_1 and m^*_1 such that $F(m_0||m_1) = F(m^*_0||m^*_1)$. This type of attack is much more demanding from a cryptanalytic view as it needs to cope with arbitrary prefixes and hence arbitrary chaining input differences (Stevens *et al.* [28]). In turn it allows much more powerful attacks, as can be seen by the recent attacks on certificate authorities using MD5 [29].

– **Hash collision attacks of type 3 (new, in this paper):** Given m_0 and m_0^*, compute m_1 such that $F(m_0||m_1) = F(m_0^*||m_1)$. This type of attack is in turn much more difficult than type 2, as it halves the degrees of freedom available to an attacker. The message difference is fixed (to zero), this means that for each MD5 compression function, instead of 1024 degrees of freedom, only 512 degrees of freedom via the message input are available to an attacker.

This leads us to the informal definition of a weak hash function, complementing the concept of a weak compression function from [15]. A weak hash function may be modeled as a random oracle, but offers additionally oracles that allow collision attacks on the hash function of type 1 and type 2, but not of type 3. The purpose of this introduction of a weak hash function is to show that MD5 can not even meet the requirements of a weak hash function, even though no type 3 collision attack on the MD5 compression function are known.

We may define the security of a hash function as a component in a concatenated combiner against collision attacks (concatenated combiner collision security, or simply C^3 security) of an n-bit hash function as the effort to find a collision attack of type 3. For MD5, despite all cryptanalytic advances in recent years, this is 2^{64}. In this paper, we show an attack suggesting that the C^3 security of MD5 is less.

3 Outline of Attack Strategies

In the following we assume it is possible to devise collision attacks of type 3w on the compression function below the birthday bound. These collision attacks will need a suitable differential path, and a method to find message pair which conforms to such a differential path. We will discuss this problem for the case of MD5 in Section 5. This alone is not enough for our attack to work, but based on such a result we propose to continue as follows. We first show how to devise a collision attack of type 3 on a hash function using a combination of birthday techniques and differential shortcut techniques. Then we continue and apply such an attack on a combiner.

3.1 Collision Attack of Type 3

The attack we propose (see Fig. 1 for an illustration) consists of three phases. A preparation phase that computes target differences (1), a birthday phase (using M_1) (2) and a differential phase (using M_2) that performs a type 3w collision attack (3), and is executed in this order.

Before the birthday phase (2), the differential phase needs to be "prepared" as follows (1). We generate a number of 2^x distinct characteristics (also called paths) through the compression function on a heap with the following property: no message difference, an arbitrary input difference (δ_2), and no output difference ($\delta_2 \boxplus \delta_3 = 0$). Let's assume each of them, when given a suitable chaining input pair, results in an effort of 2^w (or less) to find a conforming message pair. Let 2^y

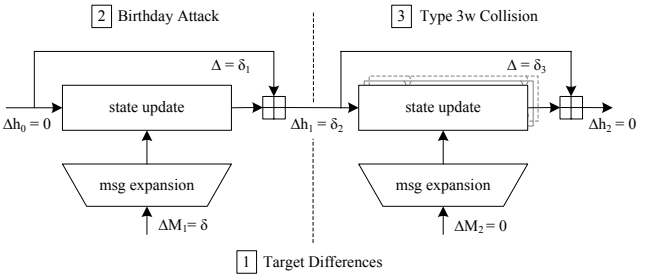


Fig. 1. Outline of attack strategy

be the cost of this path generation in terms of equivalent compression function computations. Let's further assume that each of these paths has an average number of z independent conditions on the chaining input (CI).

A single path with z conditions on the CI in fact can be used for 2^{n-z} possible pairs of CIs. Since there exist 2^{2n} pairs, 2^{n+z} randomly generated pairs would be needed before one matches the CI described by the path (δ_1 matches δ_2, and the conditions are fulfilled). Using birthday techniques, this is expected to take $2^{(n+z)/2}$ time. Given all 2^x paths, only 2^{n+z-x} randomly generated pairs are needed, which in turn is expected to take $2^{(n-x+z)/2}$ time. Hence, if $x > z$, the runtime is expected to be below the birthday bound.

For obtaining a single hash collision of type 3, the overall method may be seen as a successful cryptanalytic attack, if the sum of the runtimes for the path generation, the birthday phase, and the work to find a conforming message pair using a particular path is below the birthday bound, *i.e.* if $2^y + 2^{(n-x+z)/2} + 2^w < 2^{n/2}$. For obtaining many hash collisions of type 3, the effort to generate the heap of paths (1) may be negligible, hence to goal would be reduced to $2^{(n-x+z)/2} + 2^w < 2^{n/2}$.

3.2 Attack on the Combiner $F_1(M)||F_2(M)$

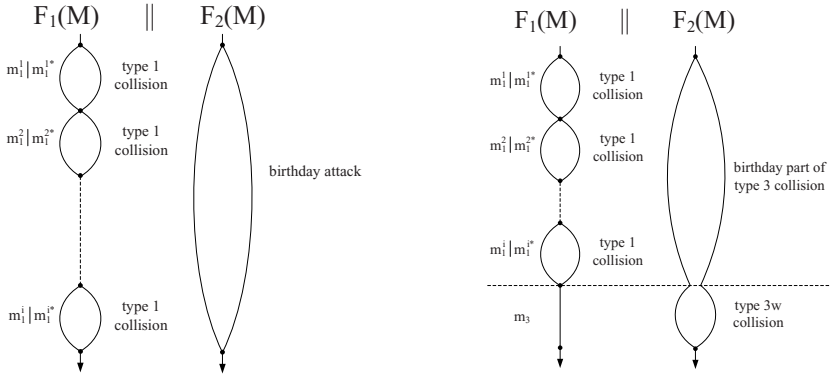
We now discuss how to use a type 3 collision attack on a hash to devise an attack on a combiner of two hash functions using it, where the first of two hash functions suffers from a type 1 collision attack.

The setting: Let $F_1(\cdot)$ and $F_2(\cdot)$ be two hash functions with output size n_1 and n_2. For the sake of simplicity we assume in the following that $n_1 = n_2 = n$. Let's further assume that F_1 suffers from a type 1 collision attack, *i.e.* given m_0, let the effort to find a m_1 and m_1^* such that $F_1(m_0||m_1) = F_1(m_0||m_1^*)$ be $2^{c_1} < 2^{n/2}$. Furthermore, assume that F_2 suffers from a type 3 collision attack, *i.e.* given m_2 and m_2^*, compute m_3 such that $F_2(m_2||m_3) = F_2(m_2^*||m_3)$ be $2^{c_2} < 2^{n/2}$. In more detail, as noted above, 2^{n+z-x} randomly generated pairs (m_2, m_2^*) are needed. The introduced symbols are summarized in Table 1.

We are now ready to formulate the new collision attack on the combiner $F_1(M)||F_2(M)$ that combines both attacks. It is also illustrated in Fig. 2.

Table 1. Symbols used in the description of the attack

symbol	description
n	output size
w	$\log_2$ of the cost of finding a conforming message pair
x	$\log_2$ of the number of distinct characteristics
y	$\log_2$ of the cost of the preparatory path generation
z	number of conditions on the chaining input
c_1	$\log_2$ of the cost of type 1 collision attack on the first hash function
c_2	$\log_2$ of the cost of type 3 collision attack on the second hash function



(a) The known approach due to Joux does not allow to exploit shortcut collision attacks on both hash functions. The lower bound is hence a birthday attack on the "smaller" hash function.

(b) New collision construction using type 3 collisions allows to exploit shortcuts attacks in both hash functions without considering the interaction in the cryptanalysis.

Fig. 2. Comparison of collision attack on a combiner

1. Let m_0 be the string of size zero and perform the type 1 collision attack on F_1 and obtain a (m_1^1,m_1^{1*}) such that $F_1(m_1^1) = F_1(m_1^{1*})$. Note that $F_2(m_1^1)$ does not collide with $F_2(m_1^{1*})$.
2. Repeat the step above while replacing m_0 with the concatenation of all previously found messages $(n + z - x)/2 - 1$ times. This means, for the i-th step (for $i = 2 \ldots (n+z-x)/2$), let $m_0 = m_1^1 || \ldots || m_1^i$ and obtain a (m_1^i,m_1^{i*}) such that $F_1(m_1^i) = F_1(m_1^{i*})$.
3. Note that by using Joux's multicollision method, we have produced a $2^{(n+z-x)/2}$-collision for F_1.
4. Perform the type 3 attack of F_2 as follows. For the birthday-part of the type 3 attacks, use the (n+z-x)/2 collisions in F_1 to obtain the required 2^{n+z-x} pairs of prefixes m_2 and m_2^*.
5. Continue with the differential shortcut part of the type 3 attack as outlined in the previous subsection, *i.e.* find a suffix m_3 such that there is a collision between
$$F_2(m_1^1||m_1^2|| \ldots ||m_1^{(n+z-x)/2}||m_3)$$

and

$$F_2(m_1^{1*}||m_1^{2*}||\ldots||m_1^{((n+z-x)/2)*}||m_3).$$

6. Also, the collision in F_1 remains.

$$F_1(m_1^1||m_1^2||\ldots||m_1^{(n+z-x)/2}||m_3)$$

collides with

$$F_1(m_1^{1*}||m_1^{2*}||\ldots||m_1^{((n+z-x)/2)*}||m_3),$$

as after the multicollision the message block m_3 without a difference is added.
7. As the same message constitutes a collision for both F_1 and F_2, this in turn results in a collision for the combiner.

The computational complexity of this procedure is as follows. The type 1 collision search on F_1 in step 1 is repeated $(n+z-x)/2$ times, which sums up to an effort of $(n+z-x)/2 \cdot 2^{c_1}$. Afterwards the type 3 collision search in F_2 is performed using the obtained multicollision. This consists of a birthday part and a type 3w compression function attack, in total costing 2^{c_2} computations. Hence, the total complexity is $(n+z-x) \cdot 2^{c_1-1} + 2^{c_2}$, and reusing the calculation for c_2 from Section 3 we arrive at

$$(n+z-x) \cdot 2^{c_1-1} + 2^y + 2^{(n-x+z)/2} + 2^w. \quad (1)$$

4 Alternative Description of MD5

MD5 is an iterative hash function based on the Merkle-Damgård design principle [4,19]. It processes 512-bit input message blocks and produces a 128-bit hash value. If the message length is not a multiple of 512, an unambiguous padding method is applied. For the description of the padding method we refer to [24]. The design of MD5 is similar to the design principles of MD4 [23]. In the following, we briefly describe the compression function of MD5. It basically consists of two parts: message expansion and state update transformation. A detailed description of the MD5 hash function is given in [24].

4.1 Message Expansion

The message expansion of MD5 is a permutation of the 16 message words m_i in each round. For each of the four rounds, a permutation of these 16 message words is used, resulting in 64 32-bit words, denoted by W_i, with $0 \leq i \leq 63$. For the permutation defining the ordering of message words we refer to [24].

4.2 State Update Transformation

The state update transformation of MD5 starts from a (fixed) initial value IV $(A_{-4}, A_{-3}, A_{-2}, A_{-1})$ of four 32-bit registers and updates them in 4 rounds of 16 steps each. The state update transformation of MD5 works on four state

variables. The state update transformation can be written to update one variable only:

$$A_i = A_{i-1} + (A_{i-4} + f(A_{i-1}, A_{i-2}, A_{i-3}) + W_i + K_i) \lll s_i.$$

However, in our case it turned out that a description which updates 2 state variables A_i and B_i is beneficial. In this case, one step is computed as follows (see also Fig. 3):

$$\begin{aligned} B_i &= (A_{i-4} + f(A_{i-1}, A_{i-2}, A_{i-3}) + W_i + K_i) \lll s_i \\ A_i &= A_{i-1} + B_i. \end{aligned}$$

In each step of MD5, different step constants K_i, rotation values s_i and Boolean functions f are used. For the definition of the constants and the rotation values we refer to [24]. The Boolean function f differs for each round of MD5: `IF` is used in the first round, `IF3` is used in round 2, and `XOR` is used in round 3 and `ONX` is used in the last round:

$$\begin{aligned} \texttt{IF}(x, y, z) &= xy \oplus \neg xz \\ \texttt{IF3}(x, y, z) &= zx \oplus \neg zy \\ \texttt{XOR}(x, y, z) &= x \oplus y \oplus z \\ \texttt{ONX}(x, y, z) &= y \oplus (x \vee \neg z) \end{aligned}$$

After the last step of the state update transformation, the initial value and the output values of the last four step are combined, resulting in the final value of one iteration known as Davies-Meyer hash construction (feed forward). The result is the final hash value or the initial value for the next message block.

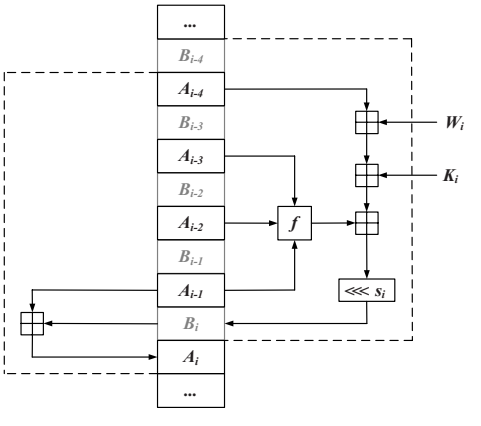


Fig. 3. Alternative description of the step update transformation of MD5 using two state variables A_i and B_i

5 Path Search Technique for MD5 Type 3 Collisions

We now tackle the problem of finding collision attacks on the compression function of MD5 of type 3w. Various automated path search techniques for MD4-like hash functions have been proposed in the past.In this section, we describe the new path search technique we developed to solve the problem. In fact it can be seen as a variation of the fine grained condition propagation originally proposed in [6].

5.1 Overview

As illustrated in Fig. 4, the `MSB`-path of [1] is a building block of our technique. Starting from this `MSB`-path in the middle of the compression function we will study and search for many characteristics which propagate through the `ONX` round in the forward direction, and through the `IF` round in the backward direction in a non-linear way. The constraint is that, despite different rotation values and Boolean functions, resulting differences in both ends of the state update will cancel out after the feed-forward operation.

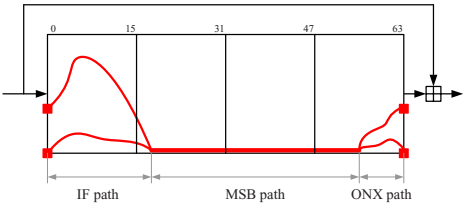


Fig. 4. The outline of the type 3w collision search with `IF`-path, MSB-path and `ONX` path

5.2 Reviewing the Path Search of De Cannière/Rechberger

In 2006, De Cannière/Rechberger [6] propose the concept of generalized conditions. The generalized conditions on a particular pair of words will be denoted by ∇X. ∇X represents as a set the values for which the conditions are satisfied. In order to write this in a compact way, we will reuse the notation listed in Table 2.

In [6], the authors describe a heuristic method to find complex nonlinear characteristics for SHA-1 in an efficient way. Follow-up work directly applied this method in various settings in the context of SHA-0 and SHA-1 [5,13,16,32]. The approach may be described as follows.

1. The starting point is a number of constraints (on the message difference and some target differences in the state) for the characteristic.
2. The basic idea of the algorithm is to randomly pick a bit position which is not restricted yet (*i.e.* , a '`?`'-bit), impose a zero-difference at this position (a '`-`'-bit), and calculate how the condition propagates. This is repeated until all unrestricted bits have been eliminated, or until it runs into an inconsistency, in which case it starts again.

Table 2. Notation for generalized conditions, possible conditions on a pair of bits. The right half is for completeness only, and will not be used in the paper.

(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)	(x_i, x_i^*)	(0, 0)	(1, 0)	(0, 1)	(1, 1)
?	✓	✓	✓	✓	3	✓	✓	-	-
-	✓	-	-	✓	5	✓	-	✓	-
x	-	✓	✓	-	7	✓	✓	✓	-
0	✓	-	-	-	A	-	✓	-	✓
u	-	✓	-	-	B	✓	✓	-	✓
n	-	-	✓	-	C	-	-	✓	✓
1	-	-	-	✓	D	✓	-	✓	✓
#	-	-	-	-	E	-	✓	✓	✓

3. The basic idea was improved by also sometimes picking 'x'-bits once they start to appear, guessing the sign of their differences ('u' or 'n'), and doing a backtracking if this does not lead to a solution.

5.3 The Path Search for MD5

We found that a direct mapping of this strategy to the case of MD5 did not lead to satisfactory results. It was not possible, with significant computational resources, to find a non-linear characteristic for the given setting. There are two main reasons for this difficulty. The first problem is caused by having two modular additions (separated by a rotation operation) within one state update. Fig. 3 shows the iterative step function of MD5 with variables A_i and B_i. Hence, two different carry expansions may occur and by guessing only bits of the state A_i, conditions propagate slowly and contradictions are detected at a very late stage. Table 3 shows an example with many free ('?') bits in B_i due to guessing bits only on A_i.

The second problem are the reduced starting constraints with only a few bit differences set in the chaining input. In the case of the type 3w collision search, there are no input difference in the message and only very few differences in the chaining input and at the chaining output. By guessing even more zero-differences ('-'-bits), the found characteristics tend to get very sparse. In fact, these sparse characteristics are impossible, which is not detected early enough by the path search algorithm. Hence, most of the time is spent with paths whose impossibility should be detected earlier. An example for a sparse (in state variable A_i), but impossible characteristic is given in Table 3.

To avoid these problems, the new MD5 path search strategy works as follows:

1. The starting point are only a small number of constraints (the chaining input difference, no message difference and the MSB path) for the characteristic.
2. Instead of just picking bits of A_i, randomly pick non-restricted bits of the state B_i as well.
3. *Immediately* guess the sign of any unrestricted difference ('x'-bits), as soon as it occurs and do a backtracking if the guess leads to a contradiction.
4. If all 'x'-bits have been determined, continue with randomly guessing zero-differences until the next 'x'-bit occurs.

Table 3. A sparse but impossible characteristic due to guessing too many zero-differences in A_i. Further, conditions do not quickly propagate into B_i and contradictions are detected at a very late stage.

i	$\nabla B_i, \nabla A_i$	∇W_i
-4	A: n----------u-----	
-3	A: n0----------B?------n-----	
-2	A: n1----------n-?-?-?-?n-----	
-1	A: un----------0?x-----?x-----	
	B: ??????????????????????------?x-----	
0	A: -n-----0----------Du----------	W: ----------
	B: ???????----------	
1	A: 7n--x-?0----------Du------0-----	W: ----------
	B: ???????x-------????x----------	
2	A: -B-?7--n----------	W: ----------
	B: ????x----------	
3	A: --D?E?#u----------0----------	W: ----------
	B: ----------	
4	A: -------n----------	W: ----------
	B: ???????x----------	
5	A: ----------	W: ----------
	B: ----------	
6	A: 0------0----------	W: ----------
	B: ----------#----------	
7	A: 0----------	W: ------#----------
	B: x----------	
8	A: n----------	W: ----------

Whenever a contradiction occurs, a simple backtracking strategy (depth first search) is applied. Using this improved strategy, global contradictions (impossible characteristics) are found at an earlier stage and impossible paths are less likely. The disadvantage of this strategy is that long carry expansions are more likely to occur and the resulting characteristic are less sparse. However, since we apply the path search mostly in the first round of MD5, even a high number of conditions can be fulfilled using simple message modification techniques [31].

6 Practical Realization and Results

We now describe implementations of several parts of the attack. This illustrates and details the method, and also serves are a validity check of the attack. To recapitulate our earlier description, the practical implementation of a type 3 collision is divided into three steps:

- **Preparatory phase.** Many special paths are searched and put on a heap.
- **Birthday phase.** Looking through possible pairs of prefixes, a pair needs to be found that matches one of the paths on the heap.
- **Differential attack phase.** Search for a conforming message pair using one of the characteristics generated earlier.

An optimization that is important in practice, is as follows. Starting form the `MSB` path in the middle of the MD5 compression function, it suffices to compute many paths through the last round (`ONX` part). The last steps of this path will impose conditions (of type 'n' and 'u') on the chaining input. This information is enough for the birthday phase. The result of the birthday phase is a prefix

pair that is compatible with a particular path on the heap. It remains to finish the characteristic, the IF part, to connect to the MSB part in the middle (see Fig. 4 for an illustration of the different parts). Having to deal with an actual chaining input pair in this phase of the attack imposes more constraints on the path search. However, as we detail in Section 6.1 and also illustrate with the characteristic in the table in Appendix A, these constraints can be dealt with in practice and do not impose any limitation on the attack.

6.1 Runtime for IF Path Search

In experiments involving the equivalent of about 2000 hours on a single core, we have verified the average runtime to find a single IF path is about 36 hours on a single core, which is about 2^{17} seconds in which about 2^{38} MD5 computations[1] could be done. For these experiments, we not only generated paths for a particular starting point, as the choice of a particular starting point has unpredictable consequences for a particular heuristic (this was also observed in [6]). Instead we generated many (about 30) starting points (*i.e.* different sets of conditions on the chaining input) in a random way to derive meaningful average runtime estimates. This suggests that, using the proposed strategy, we can expect to find a path for every set of constraints, albeit with somewhat varying runtime. In turn, this allows us to estimate the workfactor for a type 3w collision attack on MD5.

We found that the runtime for the search for IF-path does not depend on the number of differences in the CV[2]. The generation of the corresponding IF-paths can be delayed until after the birthday phase, contributes to the final search complexity only in an additive way, and is hence negligible.

6.2 A Type 3 Collision Attack Based on Actually Generated Paths

For the practical generation of type 3w collision attacks on the compression function of MD5, that in turn lead to a type 3 collision attack on the MD5 hash, we constrain ourselves to differential paths which result in runtimes for finding a colliding message pair below 2^{58}. For the preparatory step, it suffices to generate useful ONX paths. An ONX path is useful if it has a high probability, as the probability of a collision characteristic in the last round affects the resulting effort for finding a conforming message pair in a direct way. In order to give a bound on the allowable probability for the ONX path, we argue as follows. Among the four rounds (consisting of 16 steps each) the first round can easily be dealt with via simple message modification. The second round is an MSB-path and contains 16 conditions (the Boolean function needs to behave as expected at every step once, see also [1]), the third round contains no conditions as the Boolean function is an XOR, and the fourth round contains the more complex ONX-path. Improvements upon the original type 1 collision attack on MD5 by

[1] Each of our 2.0 GHz AMD Opteron(tm) cores performs about 2^{21} MD5 computations per second using OpenSSL 0.9.8g.

[2] We tested a range between 1 and 20.

Wang *et al.* concentrated on fulfilling more conditions in round 2. In a work from 2005 [25], 14 conditions could already be fulfilled. Subsequent work by Klima [14] and Stevens *et al.* [29] significantly improved upon this. Conservatively assuming to be able to only fulfill 14 conditions suggests that round number four should not have more than $58 - 16 + 14 = 56$ conditions. In Section 6.4 we give several reasons why this is a very conservative assumption.

Another important parameter of ONX paths is the number and position of differences it has in the last four steps, as this determines (except for carries via the feed-forward operation) the uniqueness of the set of allowed pairs of chaining inputs that can be canceled.

Inhere, we report on empirical findings using an actual implementation of parts of the attack. In total we spent an equivalent of about 15000 hours on a single core. The number of distinct paths for type 3w compression function attacks on MD5 we found together with their number of conditions on the IV is as follows:

number of conditions on IV	1	2	3	4	5	6	7	8	9	10
number of paths	0	0	10	130	1216	6556	21523	49293	87116	127018

Not all found paths may be of use. Let p_i be the number of distinct paths with i conditions on the IV, we want to find a j such that $(\sum_{i=1}^{j} p_i) - 2^j$ is maximal. Using the actually generated paths as described above, we found about $2^{17.34}$ paths with distinct constraints (with at most 9 relevant conditions) on the chaining input. Including also all found paths with 10 conditions would only improves the attack only if more than $2^{17.34}$ paths would be added, which is not the case.

Using the notation of Section 3, this means x=17.3, $w < 58$, and $z \leq 8$. Based on this, a type 3 collision has a runtime of $2^{(128-17.34+9)/2}(+2^{58}) = 2^{60.19}$, which is faster than the expected 2^{64} for an ideal hash function of this size. Hence, MD5 offers a C^3 security of no more than 60 bits.

Note however, that in this calculation, there is a gross imbalance between time spent on generating paths (15000 CPU hours are about 2^{47} MD5 computations) and the total runtime of the attack. Assuming to spend *e.g.* 2^7 times more computational resources in the path generation might well lead to an increase. from $x = 17$ to 24, which in turn would decrease the runtime of the overall type 3 collision attack on MD5 to 2^{57}, and would lead to an attack on the combiner MD5||SHA-1 with complexity less than 2^{59} (assuming the type 1 collision attack on SHA-1 is fast enough).

6.3 On Memory Requirements

Both, the generic method due to Joux and the new approach using a type 3 collision attack, can be implemented without requiring access to large memory. For both cases, this results in a runtime loss of about a factor $n/2$, hence the relativ advantage of the new approach over the generic method remains. Memory requirements of the attack (birthday phase and differential shortcut phase) are as follows.

Birthday phase. A naive implementation of the birthday phase would require a table of size $2^{(n-x+z)/2}$ in order to generate enough pairs to find a match with one of the 2^x paths. However, distinguished point methods may be used on a truncated version of the output of the compression function.[3]

Let t be the size of the subset of bits that is needed to represent all 2^x paths. A lower bound for t is $2x/3$, since every bit that is truncated leaves three possibilities for a path ('n', 'u', or '-'). In practice, t is higher. A memory-less method will find a partially suitable pair in time $2^{(n-t)/2}$, which would need to be repeated $2^{(t-x+z)}$ times if done independently (and hence impose the additional condition $x - z > t/2$ on the attack to be more efficient than a generic attack).

However, as described in [21,22], the distinguished points method can be used to take advantage of the birthday effect also for generating more collisions (or suitable pairs), by keeping the entries in the list of each of the distinguished points. A parallelizable version with linear speed gain is described in [20]. Hence the search needs to be repeated only $2^{(t-x+z)/2}$ times. As a result, a "memoryless" version of the birthday phase for the dedicated combiner attack behaves to a large extend as a "memoryless" version of a generic birthday attack. What is needed is memory to store 2^z candidate pairs which are the outcome of the birthday phase. In all practical settings, z is small.

Differential shortcut phase. Storing the precomputed paths for the shortcut attacks: in the order of a kilobyte per path. For practical values of x between 10 and 20, storage costs are negligible and access to this memory is only needed once.

6.4 On Conservative Estimates

There are several reasons our estimates can be considered to be very conservative:

- Basing assumption on speed-up methods (message modification, tunnels) is very conservative for the following reason. The lack of message differences, and the very simple `MSB` path in round 2 gives more freedom to apply speed-up methods as is the case in type 1 collision search attacks in earlier work.
- Also, early stop methods which further speed-up collision search are not considered.
- Runtime of various path search scenarios are measurements of actual implementations, whose runtime may be optimized by some constant factor.
- For our calculations, we use the highest possible allowed value for w (worst case). The expected value is in fact lower.

7 Conclusions and Open Problems

We proposed a new attack that allows collision attacks on combiners with a runtime below the birthday-bound of the smaller compression function when

[3] We will use the term "memoryless" to refer to these techniques, although they do in fact require some memory, albeit much less than a naive table-based approach.

the smaller compression function is MD5, potentially reducing a collision attack on a combiner like MD5||SHA-1 for the first time. This also answers an open question by Joux posed in 2004. The cryptanalytic technique we proposed for this is a combination of a birthday-style attack and a differential inside-out technique that uses different parts of a collision characteristic at different stages of an attack, both before and after a birthday phase. This technique may be of independent interest. Based on only the characteristics we generated in practical experiments with limited computational resources, a collision attack on the combiner with MD5 would already be around 2^{60} (if the "normal" collision attack on the other hash functions is fast enough), however we argued that such an estimate is *very conservative* for various reasons.

This illustrates that the MD5 hash function can not meet the requirements of a "weak hash function" as informally defined in this paper. Various open questions arise from this work: In a vein similar to concatenated combiners, or the Zipper construction [15], is it possible to come up with other collision resistant constructions that can use MD5, even though our results can be interpreted as showing that MD5 is *"weaker than weak"*? Another open problem is related to the application of our new cryptanalytic method to hash function constructions that use two or more parallel streams, like RIPEMD-160 [10], as well as several SHA-3 candidates[4]. So far it proved difficult to obtain results on RIPEMD-160, even for interesting reduced variants [17].

Acknowledgements. The authors wish to thank the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II) and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
2. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
3. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup [26], pp. 430–448
4. Damgård, I.: A Design Principle for Hash Functions. In: Brassard [2], pp. 416–427
5. De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C.M., Miri, A., Wiener, M.J. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)
6. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
7. Dierks, T., Allen, C.: The TLS Protocol Version 1.0. IETF Request for Comments: 2246 (1999)

[4] In fact we can already refer to a round-1 candidate of the SHA-3 competition which was broken [18] by techniques related to the new ideas presented in this paper.

8. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.1. IETF Request for Comments: 4346 (2006)
9. Dobbertin, H.: Cryptanalysis of MD5 Compress (1996)
10. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
11. Hoch, J.J., Shamir, A.: On the Strength of the Concatenated Hash Combiner When All the Hash Functions Are Weak. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 616–630. Springer, Heidelberg (2008)
12. Joux, A.: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In: Franklin, M. K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
13. Joux, A., Peyrin, T.: Hash Functions and the (Amplified) Boomerang Attack. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 244–263. Springer, Heidelberg (2007)
14. Klima, V.: Tunnels in hash functions: Md5 collisions within a minute. Cryptology ePrint Archive, Report 2006/105 (2006), `http://eprint.iacr.org/`
15. Liskov, M.: Constructing an Ideal Hash Function from Weak Ideal Compression Functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
16. Manuel, S., Peyrin, T.: Collisions on SHA-0 in One Hour. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 16–35. Springer, Heidelberg (2008)
17. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the Collision Resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 101–116. Springer, Heidelberg (2006)
18. Mendel, F., Schläffer, M.: On Free-start Collisions and Collisions for TIB3. In: Proceedings of ISC, Springer, Heidelberg (2009)
19. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard [2], pp. 428–446
20. van Oorschot, P.C., Wiener, M.J.: Parallel Collision Search with Cryptanalytic Applications. J. Cryptology 12(1), 1–28 (1999)
21. Quisquater, J.-J., Delescaille, J.-P.: How Easy Is Collision Search? Application to DES. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990)
22. Quisquater, J.J., Delescaille, J.P.: How Easy is Collision Search. New Results and Applications to DES. In: Brassard [2], pp. 408–413
23. Rivest, R.L.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
24. Rivest, R.L.: The MD5 Message-Digest Algorithm. IETF Request for Comments: 1321 (1992)
25. Sasaki, Y., Naito, Y., Kunihiro, N., Ohta, K.: Improved Collision Attack on MD5. Cryptology ePrint Archive, Report 2005/400 (2005), `http://eprint.iacr.org/`
26. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
27. Simon, D.R.: Findings Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998)
28. Stevens, M., Lenstra, A.K., de Weger, B.: Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)

29. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D.A., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 55–69. Springer, Heidelberg (2009)
30. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup [26], pp. 17–36
31. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
32. Yajima, J., Iwasaki, T., Naito, Y., Sasaki, Y., Shimoyama, T., Peyrin, T., Kunihiro, N., Ohta, K.: A Strict Evaluation on the Number of Conditions for SHA-1 Collision Search. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences E92-A(1), 87–95 (2009)

A Supplementary Material for Obtained Results

A particular low-weight input chaining difference becomes the MSB-path in the course of 10 steps. The following table contains the full characteristics illustrating a candidates for a type 3w compression function attack. As a proof-of-concept, we provide a representative example of a conforming message pair in Table 4.

i	∇A_i	∇W_i
-4	A: u----------u---------	
-3	A: u----------	
-2	A: n----------0----------	
-1	A: 1----------0----------	
0	A: 0----------0u111110000000010-	W: ----------
1	A: ----------1u000000000000000-	W: ----------
2	A: ----------unnnnnnnnnnnnnnnn1	W: ----------
3	A: --unnnnn-----0110001011100000000	W: ----------
4	A: --000000---n-u11u00000000110111-	W: ----------1----00001000111--
5	A: --011111---0-n11n0----------	W: ----------0000-----1100-----
6	A: ----------nuuuuuu----------	W: ----------
7	A: ----------0110000----------	W: ----------
8	A: 0----------1101101----------	W: ----------
9	A: 0----------	W: ----------
10	A: u----------	W: ----------
11	A: n----------	W: ----------
12	A: n----------	W: ----------
13	A: n----------	W: ----------
	...	...
54	A: n----------	W: ----------
55	A: n----------0---------	W: ----------
56	A: u----------1---------	W: ----------
57	A: n----------0n---------	W: ----------
58	A: u----------n---------	W: ----------
59	A: n----------nu---------	W: ----------
60	A: n----------n---------	W: ----------
61	A: n----------11---------	W: ----------
62	A: n----------0---------	W: ----------
63	A: ----------	W: ----------
	FF: ----------	
	FF: ----------	
	FF: ----------	
	FF: ----------	

Table 4. A conforming message pair for the first 16 steps

H_1	C4F12702 D25873C9 5B88CE47 9A8EBB1D
H_1^*	44F12502 525873C9 DB88CE47 9A8EBB1D
ΔH_1	80000200 80000000 80000000 00000000
M_1	D830883A AA2456AA 24B9260C D2F17AE9 F893211E 08F4298C 8A0C7756 3492552F C7CB7D9D 7FB6804C 9336A183 44256E0D 6D095FCF 08D8D9EA 5D79C0BA 0F2CD7C5
M_1^*	D830883A AA2456AA 24B9260C D2F17AE9 F893211E 08F4298C 8A0C7756 3492552F C7CB7D9D 7FB6804C 9336A183 44256E0D 6D095FCF 08D8D9EA 5D79C0BA 0F2CD7C5
ΔM_1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_2	156733C4 4A05644B 20E6A26E 7718EBA4
H_2^*	956733C4 CA05644B A0E6A26E F718EBA4
ΔH_2	80000000 80000000 80000000 80000000