Generating Multicollisions for MD4 and MD5

Studienarbeit

Christian Knopf
February 2008

Leibniz Universität Hannover
Institut für Theoretische Informatik
Contents

1 Introduction 1

2 Multicollisions for MD4 3
   2.1 Collisions of MD4 3
   2.2 Changes to the Source Code 3
   2.3 Results 4

3 Multicollisions for MD5 5
   3.1 Collisions of MD5 5
   3.2 Changes to the Source Code 5
   3.3 Results 6

4 Conclusion 7

Bibliography 8
1 Introduction

Cryptographic hash functions such as MD4 and MD5 are used for many purposes and are part of many cryptographic protocols. For most purposes, the hash function is required to be collision free. There are many sources on these topics, so I will avoid detailed explanations here. For further information, see, for example, [1] or [2].

A collision for a hash function $H$ is a pair of messages $m$ and $m'$ that have the same hash: $H(m) = H(m')$.

For a secure cryptographic hash function, colliding messages can not be found more efficiently than with a brute force attack. Because of the birthday paradox, such an attack requires about $2^{n/2}$ calculatory steps and memory in the order of $2^{n/2}$.

Hash functions following the Merkle-Damgård construction scheme iterate over all blocks of a message and update an internal state to calculate the hash of the complete message. Once two different messages $m$ and $m'$ are found to lead to the same internal state of a hash function, any number of collisions can be produced by appending any message $k$ to $m$ and $m'$ simultaneously: $H(m \circ k) = H(m' \circ k)$.

Merkle-Damgård strengthened hash functions append a final block containing the length of the message before hashing for security purposes. Therefore, when Merkle-Damgård strengthening is used, the colliding messages are required to have the same length to produce a collision.

MD4 [3] and MD5 [4] both are 128-bit hash functions. Their blocklength is 512 bits. Both hash functions follow the Merkle-Damgård construction scheme, as do all frequently used cryptographic hash functions today.

However, as Antoine Joux first pointed out [5], multicollisions can be found for any iterated hash function if collisions can be found for any internal state of the hash function. The simple construction consists of the concatenation of two or more collision pairs:

$$H(m_1 \circ m_2) = H(m'_1 \circ m_2) = H(m_1 \circ m'_2) = H(m'_1 \circ m'_2)$$

Here, two collision pairs result in a 4-collision. Therefore, a $2^n$-collision
can be found with as little work as finding $n$ collision pairs.

Generating an 8-collision $C_1 - C_8$ out of the three collision pairs $m_1/m'_1$, $m_2/m'_2$, and $m_3/m'_3$. $\rho$ can be any pre-chosen message, and any string $M$ can be appended to the eight colliding messages $c_i^3$. 
2 Multicollisions for MD4

2.1 Collisions of MD4

The MD4 hash function was designed in 1990 by Ronald L. Rivest. It was created as an exceptionally fast hash function in software, especially on x86 processors. Shortly after its publication, the first flaws were discovered. Hans Dobbertin found the first collision in 1996 [6]. In 2005, Xiaoyun Wang, et al., published a very efficient attack [7]. Based on this publication, Patrick Stach implemented an MD4 collision generator and made its sources available [8]. It is able to reliably find MD4 collisions in well under a minute on a standard computer.

While there are slightly faster methods available for collision generation [9], there is no need to incorporate them for this purpose.

2.2 Changes to the Source Code

Little changes had to be made to the program to create a generator for multicollisions. Of course, the input of a new iteration of the collision finding algorithm has to be set to the intermediate hash of the last collision pair. Additionally, the two generated messages are hashed to verify the collision. Therefore, an implementation of the MD4 algorithm had to be added appropriately, an implementation by Chrisophe Devine was chosen [10].

To provide the actual multicollision, all collision pairs are written to separate files. This also eases proving the correctness of the code. The generated files are of the form collision-n.l. Here, n is the number of the collision, l only takes the values a and b for the two messages of each collision.

In order to measure the time spent on each iteration, the clock() function was used. The initial value of the random number generator is printed so that all computations can be repeated and verified.

The maximum number of collisions to be generated can either be provided at compile time, or directly in the source code.
2.3 Results

The resulting code was compiled and run on an average AMD Athlon XP 2000+ system. The program is able to generate a $2^{64}$-collision in well under 10 minutes.

Compiling with optimizations results in about 15% faster calculations.

<table>
<thead>
<tr>
<th>Random Seed</th>
<th>Time [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x46a324a4</td>
<td>498.3</td>
</tr>
<tr>
<td>0x44ff26b3</td>
<td>416.1</td>
</tr>
<tr>
<td>0x43062864</td>
<td>380.7</td>
</tr>
<tr>
<td>0x41fd29f7</td>
<td>409.6</td>
</tr>
<tr>
<td>0x40642ba0</td>
<td>533.1</td>
</tr>
<tr>
<td>0x4e652dd3</td>
<td>522.2</td>
</tr>
<tr>
<td>0x4c472ff2</td>
<td>340.5</td>
</tr>
<tr>
<td>0x4b0b315b</td>
<td>350.2</td>
</tr>
<tr>
<td>0x4a3832c7</td>
<td>330.5</td>
</tr>
<tr>
<td>0x48823425</td>
<td>360.6</td>
</tr>
</tbody>
</table>

Average: 414.18

By randomly concatenating either part a or part b of each collision to combine any of the $2^{64}$ different messages, a short test can be administered. This can be achieved, for example, by a simple shell command. Of course, for smaller values of $n$, all $2^n$ messages can be hashed.
3 Multicollisions for MD5

3.1 Collisions of MD5

The MD5 hash function was designed as a result of security flaws as a strengthened, more secure version of MD4 and published in 1991. Although a few minor flaws were discovered over time, the hash function remained unbroken until a collision was published in 2004. The attacks quickly became more sophisticated. In 2006, Vlastimil Klima showed an improved attack based on several different cryptanalytic methods [11]. He also published a collision finding program along with its source code [12]. Run on an average computer, the program is able to find collisions within minutes.

3.2 Changes to the Source Code

The code of the collision generator had to be heavily modified to give the desired results. Because of czech variable names and comments, and especially due to of poor indentation, the code is somewhat hard to understand and modify.

The generator was made to compile in windows environments (for example, with the gcc of the cygwin environment), so different header files and function calls had to be used for time measurement. However, they were partly included as comments.

Additionally, several of the changes described above were also used here. I applied the same method of writing the collision pairs to files, as well as the modifications for the intermediate hash value to create multicollisions as in the MD4 code. Also, the MD5 collision generator now prints similar information and incorporates the same method of providing the maximum number of desired collisions.

The time measurement was already part of the program, as was a verification of the hash value of the generated message pair. However, the algorithm evidently does not generate a collision under all circumstances, the probability of a failure is higher than 0. Therefore, I modified the verification step included code to restart the calculation of the current collision as needed.
3.3 Results

On the same system, a $2^{64}$-collision of MD5 can be calculated in roughly 40 minutes on average. Optimizations done by gcc are able to deliver a speedup of almost 2.

With an average number of almost 15 non-colliding messages calculated by the program in each run, the probability is almost at 20%. However, it is easy to circumvent any such problems, as is shown in the code.

<table>
<thead>
<tr>
<th>Random Seed</th>
<th>Time [seconds]</th>
<th>non-collisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x347efabc</td>
<td>2679.8</td>
<td>10</td>
</tr>
<tr>
<td>0x79db7349</td>
<td>2413.5</td>
<td>14</td>
</tr>
<tr>
<td>0x6dd48f98</td>
<td>2774.1</td>
<td>9</td>
</tr>
<tr>
<td>0xd4ff54c7</td>
<td>2276.9</td>
<td>16</td>
</tr>
<tr>
<td>0x7dff21b4</td>
<td>2802.4</td>
<td>18</td>
</tr>
<tr>
<td>0x9fd33b05</td>
<td>1963.6</td>
<td>13</td>
</tr>
<tr>
<td>0x3d700392</td>
<td>2179.2</td>
<td>17</td>
</tr>
<tr>
<td>0x8f7fd3da</td>
<td>2239.3</td>
<td>17</td>
</tr>
<tr>
<td>0xb75af3e4</td>
<td>2528.3</td>
<td>19</td>
</tr>
<tr>
<td>0xbffe9e42</td>
<td>2318.9</td>
<td>12</td>
</tr>
<tr>
<td>0xed777c44</td>
<td>2401.4</td>
<td>19</td>
</tr>
<tr>
<td>0xf4b9d0de</td>
<td>2438.8</td>
<td>15</td>
</tr>
<tr>
<td>Average:</td>
<td>2418.0</td>
<td>14.92</td>
</tr>
</tbody>
</table>

A randomized test similar to that of the MD4 multicollisions showed the accuracy of the program.
4 Conclusion

Compilation of the original sources, as well as making small changes to the code is fairly simple. By concatenating many suitable colliding message pairs, an exponentially large number of collisions can be easily generated, as I have shown in the presented code and in this document. The end result is that generating $n$ collisions can be done in $\ln(n)$ time and space.

The most important implication is that the concatenation of two hash functions of length $n$ does not result in a security equal to a hash function of length $2n$.

In $2^{64}$ random messages, two will have the same 128-bit hash value with high probability, regardless of the hash function used. Therefore, out of a $2^{64}$-collision of MD4, two messages will likely also collide under MD5 and vice versa.

Unfortunately, it is a very computationally demanding task to hash $2^{64}$ messages of 1024 bytes each: With a hashing speed of about 1 Gbit/second and $2^{20}$ hashing processors, the calculations alone take $2^{24}$ seconds, or roughly half a year. Additionally, $2^{64}$ hash values, or 256 exabytes would have to be saved into a database, and each newly generated hash would have to be compared against all existing values. Especially these memory requirements can be only considered at the far edge of feasibility.

Of course, a much more important result would be the generation of several single blocks that form an $n$-collision. This is possible for hash functions with a compression function which has a longer input than output. In MD4 and MD5, it is theoretically possible to find $2^{384}$ input blocks that all have the same hash value.

Moreover, calculating a collision for MD4 $\circ$ MD5 would be a small breakthrough in hashing cryptanalysis. However, because of small differences of the two hash functions, especially the added round in MD5, makes combining cryptanalytic results of both hashes very complicated.
References


http://www.stud.uni-hannover.de/~cknopf/Diplomarbeit.pdf

Also: Ronald L. Rivest, **The MD4 Message-Digest Algorithm**, RFC 1320, April 1992.  


http://www.infosec.sdu.edu.cn/paper/md4-ripped-attck.pdf
http://www.stachliu.com/md4coll.c


http://xyssl.org/code/source/md4/


http://cryptography.hyperlink.cz/2006/web_version_1.zip

http://www.jonelo.de/java/jacksum/
Digital Data

Contents of the CD:

- an electronic version of this document.
- Jacksum, a java program to verify MD4 hashes, among others [13].
- the MD4 multicollision generator, along with a bash script to randomly verify collisions.
- the MD5 multicollision generator, along with a similar bash script for verification of the results.
- a $2^{64}$-multicollision each for MD4 and MD5.