



AccessData®

WhitePaper

MD5 Collisions

The Effect on Computer Forensics

April 2006



ACCESSDATA, ON YOUR RADAR

MD5 Collisions: The Impact on Computer Forensics

Hash functions are one of the basic building blocks of modern cryptography. They are used for everything from password verification to digital signatures. A hash function has three fundamental properties:

- It must be able to easily convert digital information (i.e. a message) into a fixed length hash value.
- It must be computationally impossible to derive any information about the input message from just the hash.
- It must be computationally impossible to find two files to have the same hash.

A collision is when you find two files to have the same hash. The research published by Wang, Feng, Lai and Yu demonstrated that MD5 fails this third requirement since they were able to generate two different messages that have the same hash.

In computer forensics hash functions are important because they provide a means of identifying and classifying electronic evidence. Because hash functions play a critical role in evidence authentication, a judge and jury must be able trust the hash values to uniquely identify electronic evidence. A hash function is unreliable when you can find any two messages that have the same hash.

Birthday Paradox

The easiest method explaining a hash collision is through what is frequently referred to as the Birthday Paradox. How many people on the street would you have to ask before there is greater than 50% probability that one of those people will share your birthday (same day not the same year)? The answer is 183 (i.e. $365/2$), because you are attempting to find someone who matches one specific date one comparison at a time.

On the other hand, how many people must be in a room at the same time before there is a probability greater than 50% that any two people share the same birthday? This number is surprisingly lower: 23 (hence the paradox). When 23 people are in a room, there are a total of 23 birthdays to match by 253 different possible combinations of dates. The chances of two birthdays being the same increases dramatically, so the amount of people needed to make the chances 50% is lower than most would guess. It's a standard statistical problem: for each additional person n , who enters a room, the number of birthday pairs increases by $n-1$. As more people enter the room, the number of birthday pairs increases rapidly until a matching pair exists.

In the first example, the attempt was to find a person that matched one specific birthday (yours). When matching a specific day, each person has only a $1/365$ chance of being born on your birthday. In cryptography, the first example is analogous to a brute force or exhaustive key space attack. This is the process used in most password recovery attacks. In the second example, any birthday pair will suffice. This second type of attack is the process used by cryptographers to attack hash functions.

If a hash function has a key space of 64 bits, then an exhaustive key space attack would require a computer test up to 2^{64} combinations. If a single computer could process one

million hashes per second, and an advisory could use a distributed network attack to harness the CPU power of 10,000 computers, it could still take up to 58 years to exhaust the key space. However, if the goal was simply to find any hash match, a single computer could find that match in slightly more than an hour. For a more detailed description of the Birthday Paradox see Patterson (1987).

Fortunately MD5 and other common hash functions have substantially larger key lengths than 64-bits. For MD5 the key length is 128 bits, for SHA-1 the key length is 160 bits, SHA-256 the key length is 256 bits. If a cryptographic weakness is discovered in the design of the hash algorithm, however, this weakness can reduce the effective key length of the hash function to be less than the intended design length. In this case, the weakness makes possible the potential for a birthday attack to successfully find a hash collision. Such a weakness was found with the MD5 algorithm.

MD5

The MD5 hash function was developed in 1994 by cryptographer Ron Rivest as a stronger alternative to the MD4 algorithm, developed in 1992. The MD5 algorithm breaks a file into 512 bit input blocks. Each block is run through a series of functions to produce a unique 128 bit hash value for the file. Changing just one bit in any of the input blocks should have a cascading effect that completely alters the hash results. Furthermore, since the key size of 128 bits has 3.4×10^{38} possible combinations, the chance of randomly finding two files that produce the same hash value should be computationally impossible (Schneier, 1996).

Cryptanalysis of MD5

MD5 has been intensely scrutinized by the cryptographic community since its original release. Prior to 2004, most of the research attacks against MD5 demonstrated only minor weaknesses in the design. Two particularly notable exceptions, however, indicated serious design problems.

The first indication that MD5 might have a design flaw was in a paper published by Den Boer and Bosselaers in which it was demonstrated that, given certain different input conditions, it was possible for identical internal states to exist for some of the MD5 computations. Boer and Bosselaers were not able to expand upon these internal anomalies to produce duplicate hashes for different input values (Den Boer and Bosselaers, 1994).

The second indication occurred in 1996 when Dobbertin was able to demonstrate that the MD5 algorithm could produce identical hashes for two different messages if the initialization vector could be chosen (Dobbertin, 1996). The initialization vector is the value to which the MD5 internal variables are initially set before beginning the hashing process. Because MD5, when used in real life, is always set to the same initialization state (IV_0), Dobbertin's result did not present an immediate security concern. His work did, however, demonstrate that an MD5 collision was inevitable.

In the summer of 2004, the cryptographers Wang et al. demonstrated their ability to generate MD5 collisions using the standard initialization vector IV_0 . This research showed that it is possible to create two related 512 bit input blocks and

modify specific bits within these blocks to create two slightly different messages that have the same hash value. The amount of time to create an MD5 message pair was, on average, 1 hour (Wang et al., 2004).

Table 1: MD5 Collision

Message 1	1 st Block	02DD31D1 C4EEE6C5 069A3D69 5CF9AF98 87 B5CA2F AB7E4612 3E580440 897FFBB8 0634AD55 02B3F409 8388E483 5A41 71 25 E8255108 9FC9CDF7 F2 BD1DD9 5B3C3780
	2 nd Block	D11D0B96 9C7B41DC F497D8E4 D555655A C7 9A7335 0CFDEBF0 66F12930 8FB109D1 797F2775 EB5CD530 BAADE822 5C15 CC 79 DDCB74ED 6DD3C55F D8 0A9BB1 E3A7CC35
Message 2	1 st Block	02DD31D1 C4EEE6C5 069A3D69 5CF9AF98 07 B5CA2F AB7E4612 3E580440 897FFBB8 0634AD55 02B3F409 8388E483 5A41 F1 25 E8255108 9FC9CDF7 72 BD1DD9 5B3C3780
	2 nd Block	D11D0B96 9C7B41DC F497D8E4 D555655A 47 9A7335 0CFDEBF0 66F12930 8FB109D1 797F2775 EB5CD530 BAADE822 5C15 4C 79 DDCB74ED 6DD3C55F 58 0A9BB1 E3A7CC35
MD5 Hash		8D5E7019 6324C015 715D6B58 61804E08

Response of the Cryptographic Community to MD5 Collisions

While these results are mathematically significant, they do not present an immediate cause for alarm. Creating two messages that have identical MD5 hashes requires very specific circumstances that would have an extremely rare chance of actually existing in the regular world. Additionally, this research does not provide a hacker with any new technique to break through a firewall, attack a public key encryption system, or fabricate a false digitally signed message. Nevertheless, this research does point out a design weakness in the MD5 algorithm, and the cryptographic community needs to increase the diligence in which it searches for a new hash standard. Bruce Schneier summarized the feelings of many in the cryptographic community with his statement:

“The magnitude of the results depends on who you are. If you’re a cryptographer, this is a huge deal. While not revolutionary, these results are substantial advances in the field. The techniques described by the researchers are likely to have other applications, and we’ll be better able to design secure systems as a result. As a user of cryptographic systems—as I assume most readers are—this news is important, but not particularly worrisome. MD5 and SHA aren’t suddenly insecure; no one is going to be breaking digital signatures or reading encrypted messages anytime soon with these techniques. The electronic world is no less secure after these announcements than it was before.” (Schneier, 2004).

The Impact of MD5 Collision on Using MD5

The recent research on MD5 collision should have little impact on the use of MD5 for evidence authentication in computer forensics for three reasons:

1. MD5 is still secure against a brute force attack—is computationally impossible to modify the contents of a message such that the hash of the new message matches some pre-determined hash value. No one in the cryptographic research community has yet to be able to generate a new file or modify an existing file so that the new file will convey intelligible information and still match a pre-determined MD5 hash from a different file.
2. Changing one bit in the evidence will still cause a cascading effect that dramatically changes the MD5 hash result. A collision similar to the one demonstrated by Wang et al. can be produced only using very specific input blocks. Since these types of input blocks do not occur in the real world, the internal state of the MD5 engine that allowed for the collision will not naturally occur. The MD5 engine does a remarkably good job of generating a cascade effect on all the bits in the hash value even when just a single bit in the input file is changed. The forensics community can still rely upon MD5 to do an excellent job at identifying even the smallest change in electronic data.
3. The chances of a birthday collision from files that are part of the NIST data set or hash keeper project are very remote. The birthday collision that was produced by these cryptographers required a very special set of circumstances within the internal variables of the MD5 engine. To believe that this kind of state would occur naturally when analyzing files that would normally be found on a computer, PDA or similar electronic device is unrealistic. In the real world, the number of files required for a 50% probability for an MD5 collision to exist is still 2^{64} or 1.8×10^{19} . The chance of an MD5 hash collision to exist in a computer case with 10 million files is still microscopically low.

For those who wish to be cautious, electronic evidence using both MD5 and another hash function such as SHA-1 or SHA-256 is still possible. Since these hash functions are linearly independent of each other, the resulting uniqueness of having both hash values would be the sum of the bits from each individual hash. For example, a file that has been hashed with both MD5 (128 bits) and SHA-1 (160 bits) would have an effective uniqueness of 288 bits or 1×10^{86} . Even if a weakness could be found that reduces the effective key size of one of these hash functions, two different data streams that would have the same MD5 and SHA-1 hash is still computationally unrealistic in our life time.

The Future

The struggle to make a perfect crypto-system has long eluded cryptographers. New cryptographic codes are created and broken every day. This challenge is what drives the cryptographic technology advances forward. Cryptographers have new information about how to design hash functions that Ron Rivest did not know back in 1994 when he published his work on the MD5 algorithm. This new announcement does not present a current security threat nor does it make the use of MD5 for evidence authentication any

less trustworthy. Instead, this research gives mathematicians information about how to design hash functions to improve the next generation's codes.

As a result of these developments, a new set of hash algorithms will most certainly emerge in the next several years. These new algorithms will be resistant to the weakness discovered by Wan, et al. One of these new algorithms will rise to the top and for a period of time serve as the world's next hash function standard. Several years afterwards, a brilliant mathematician will discover a weakness in this new algorithm, publish his results, and the process of finding another hash standard will start all over again.

The computer forensics community will want to embrace the new hash technology once it has been thoroughly tested by the cryptographic community. Until then, computer forensics examiners should feel comfortable in their continued, albeit short, term use of MD5. When possible, hashing electronic evidence with both MD5 and a second hash function such as SHA-1 or SHA-256 is always a good idea. The forensics software needs to support multiple hash functions, however, for this to be feasible. Unless new information emerges showing a further weakness in the MD5 hash algorithm, we should continue to use MD5. Forensics examiners should work with the manufacturers of forensic software so that new releases, when possible, will start implementing stronger hash functions, such as SHA-1 or SHA-256, into the forensics process.

References

- Den Boer B, Bosselaers A. "Collisions for the compression function of MD5, Advances in Cryptology" EUROCRYPT '93. LNCS 765; 1994. p. 293-304.
- Dobbertin Hans. "Cryptanalysis of MD5 compress." German Information Security Agency; May 1996.
- Patterson Wayne. Mathematical Cryptology for Computer Scientists and Mathematicians. Rowman & Littlefield, 1987. p 156-158.
- Schneier Bruce. "Protocols, Algorithms and SourceCode." Applied Cryptography Second Edition. C. John Wiley & Sons, Inc.; 1996. p. 436-441.
- Schneier Bruce. "Opinion: cryptanalysis of MD5 and SHA: Time for a new standard." Computerworld; April 19, 2004.
- Wang Xianyan, Feng Dengguo, Lai Xuejia, Yu Hongbo. "Collisions for Hash Functions MD4, MD5 Haval-128 and RIPEMD." CRYPTO '04; Revised August 17, 2004.