"MD5"

Wikipedia

MD5

General	
Designers	Ron Rivest
First published	April 1992
Series	MD2, MD4, MD5, MD6
Detail	
Digest sizes	128 bits
Structure	Merkle–Damgård construction
Rounds	4 [1]
Best public cryptanalysis	
A 2009 attack by Tao Xie and Dengguo Feng breaks MD5 collision resistance using just $2^{20.96}$ time. This attack runs in a few seconds on a regular computer.[2]	

In cryptography, **MD5** (**Message-Digest algorithm 5**) is a widely used cryptographic hash function with a 128-bit (16-byte) hash value. Specified in RFC 1321, MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. However, it has been shown that MD5 is not collision resistant;[3] as such, MD5 is not suitable for applications like SSL certificates or digital signatures that rely on this property. An MD5 hash is typically expressed as a 32-digit hexadecimal number.

MD5 was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5. While it was not a clearly fatal weakness, cryptographers began recommending the use of other algorithms, such as SHA-1 (which has since been found also to be vulnerable). In 2004, more serious flaws were discovered, making further use of the algorithm for security purposes questionable; specifically, a group of researchers described how to create a pair of files that share the same MD5 checksum.[4] [5] Further advances were made in breaking MD5 in 2005, 2006, and 2007.[6] In an attack on MD5 published in December 2008, a group of researchers used this technique to fake SSL certificate validity.[7] [8] US-CERT of the U. S. Department of Homeland Security said MD5 "should be considered cryptographically broken and unsuitable for further use,"[9] and most U.S. government applications will be required to move to the SHA-2 family of hash functions after 2010.[10]

History and cryptanalysis

MD5 is one in a series of message digest algorithms designed by Professor Ronald Rivest of MIT (Rivest, 1994). When analytic work indicated that MD5's predecessor MD4 was likely to be insecure, MD5 was designed in 1991 to be a secure replacement. (Weaknesses were indeed later found in MD4 by Hans Dobbertin.)

In 1993, Den Boer and Bosselaers gave an early, although limited, result of finding a "pseudo-collision" of the MD5 compression function; that is, two different initialization vectors which produce an identical digest.

In 1996, Dobbertin announced a collision of the compression function of MD5 (Dobbertin, 1996). While this was not an attack on the full MD5 hash function, it was close enough for cryptographers to recommend switching to a replacement, such as SHA-1 or RIPEMD-160.

The size of the hash—128 bits—is small enough to contemplate a birthday attack. MD5CRK was a distributed project started in March 2004 with the aim of demonstrating that MD5 is practically insecure by finding a collision using a birthday attack.

MD5CRK ended shortly after 17 August 2004, when collisions for the full MD5 were announced by Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu.[4] [5] [11] Their analytical attack was reported to take only one hour on an IBM p690 cluster.

On 1 March 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger demonstrated[12] construction of two X.509 certificates with different public keys and the same MD5 hash, a demonstrably practical collision. The construction included private keys for both public keys. A few days later, Vlastimil Klima described[13] an improved algorithm, able to construct MD5 collisions in a few hours on a single notebook computer. On 18 March 2006, Klima published an algorithm[14] that can find a collision within one minute on a single notebook computer, using a method he calls tunneling.

In 2009, the United States Cyber Command used an MD5 hash of their mission statement as a part of their official emblem. [15]

On December 24, 2010, Tao Xie and Dengguo Feng announced the first published single-block MD5 collision (two 64-byte messages with the same MD5 hash).[16] Previous collision discoveries relied on multi-block attacks. For "security reasons", Xie and Feng did not disclose the new attack method. They have issued a challenge to the cryptographic community, offering a US$ 10,000 reward to the first finder of a different 64-byte collision before January 1, 2013.

In 2011 an informational RFC was approved to update the security considerations in RFC 1321 (MD5) and RFC 2104 (HMAC-MD5). [17]

Security

The security of the MD5 hash function is severely compromised. A collision attack exists that can find collisions within seconds on a computer with a 2.6Ghz Pentium4 processor (complexity of $2^{24.1}$).[18] Further, there is also a chosen-prefix collision attack that can produce a collision for two chosen arbitrarily different inputs within hours, using off-the-shelf computing hardware (complexity 2^{39}).[19]

These collision attacks have been demonstrated in the public in various situations, including colliding document files[20] [21] and digital certificates.[7]

As of 2009[22] , a theoretical attack also breaks MD5's preimage resistance.

Collision vulnerabilities

In 1996, collisions were found in the compression function of MD5, and Hans Dobbertin wrote in the RSA Laboratories technical newsletter, "The presented attack does not yet threaten practical applications of MD5, but it comes rather close ... in the future MD5 should no longer be implemented...where a collision-resistant hash function is required."[23]

In 2005, researchers were able to create pairs of PostScript documents[24] and X.509 certificates[25] with the same hash. Later that year, MD5's designer Ron Rivest wrote, "md5 and sha1 are both clearly broken (in terms of collision-resistance),"[26] and RSA Laboratories wrote that "next-generation products will need to move to new algorithms."[27]

On 30 December 2008, a group of researchers announced at the 25th Chaos Communication Congress how they had used MD5 collisions to create an intermediate certificate authority certificate which appeared to be legitimate when checked via its MD5 hash.[7] The researchers used a cluster of Sony Playstation 3s at the EPFL in Lausanne, Switzerland[28] to change a normal SSL certificate issued by RapidSSL into a working CA certificate for that issuer, which could then be used to create other certificates that would appear to be legitimate and issued by RapidSSL. VeriSign, the issuers of RapidSSL certificates, said they stopped issuing new certificates using MD5 as their checksum algorithm for RapidSSL once the vulnerability was announced.[29] Although Verisign declined to revoke existing certificates signed using MD5, their response was considered adequate by the authors of the exploit

(Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger).[7] Bruce Schneier wrote of the attack that "[w]e already knew that MD5 is a broken hash function" and that "no one should be using MD5 anymore."[30] The SSL researchers wrote, "Our desired impact is that Certification Authorities will stop using MD5 in issuing new certificates. We also hope that use of MD5 in other applications will be reconsidered as well."[7]

MD5 uses the Merkle–Damgård construction, so if two prefixes with the same hash can be constructed, a common suffix can be added to both to make the collision more likely to be accepted as valid data by the application using it. Furthermore, current collision-finding techniques allow to specify an arbitrary *prefix*: an attacker can create two colliding files that both begin with the same content. All the attacker needs to generate two colliding files is a template file with a 128-byte block of data aligned on a 64-byte boundary that can be changed freely by the collision-finding algorithm.

Preimage vulnerability

In April 2009, a preimage attack against MD5 was published that breaks MD5's preimage resistance. This attack is only theoretical, with a computational complexity of $2^{123.4}$ for full preimage and $2^{116.9}$ for a pseudo-preimage.[31]

Other vulnerabilities

A number of projects have published MD5 rainbow tables online, that can be used to reverse many MD5 hashes into strings that collide with the original input, usually for the purposes of password cracking.

The use of MD5 in some websites' URLs means that search engines such as Google can also sometimes function as a limited tool for reverse lookup of MD5 hashes.[32]

Both these techniques are rendered ineffective by the use of a sufficiently long salt.

Applications

MD5 digests have been widely used in the software world to provide some assurance that a transferred file has arrived intact. For example, file servers often provide a pre-computed MD5 (known as Md5sum) checksum for the files, so that a user can compare the checksum of the downloaded file to it. Unix-based operating systems include MD5 sum utilities in their distribution packages, whereas Windows users use third-party applications.

However, now that it is easy to generate MD5 collisions, it is possible for the person who created the file to create a second file with the same checksum, so this technique cannot protect against some forms of malicious tampering. Also, in some cases the checksum cannot be trusted (for example, if it was obtained over the same channel as the downloaded file), in which case MD5 can only provide error-checking functionality: it will recognize a corrupt or incomplete download, which becomes more likely when downloading larger files.

MD5 is widely used to store passwords.[33] [34] To mitigate the vulnerabilities mentioned above, one can add a salt to the passwords before hashing them. Some implementations may apply the hashing function more than once—see key stretching.

MD5 and other hash functions are also used in the field of electronic discovery, in order to provide a unique identifier for each document that is exchanged during the legal discovery process. This method can be used to replace the "Bates stamp" numbering system that has been used for decades during the exchange of paper documents.

Algorithm

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit little endian integers); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with a 64-bit integer representing the length of the original message, in bits.

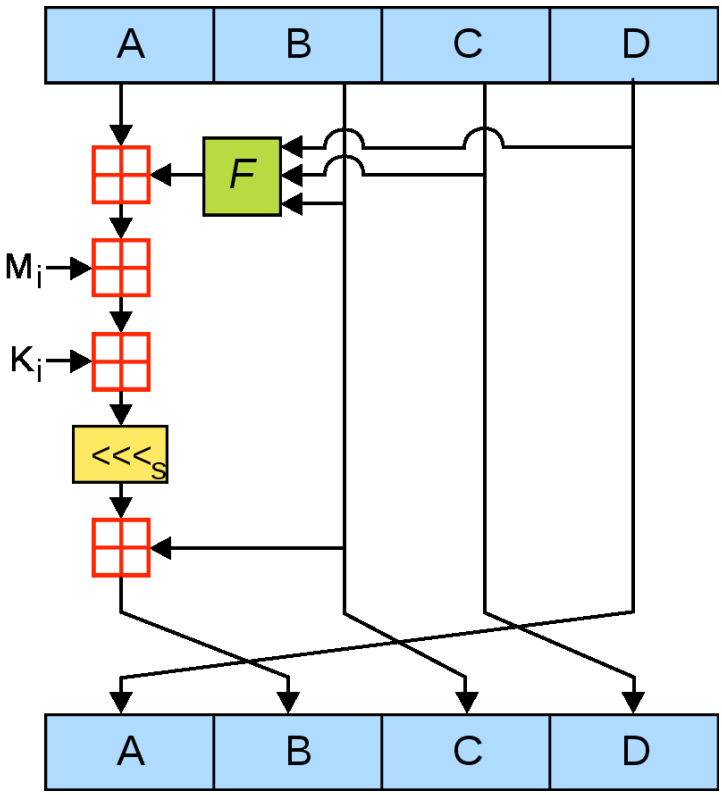

Figure 1. One MD5 operation. MD5 consists of 64 of these operations, grouped in four rounds of 16 operations. F is a nonlinear function; one function is used in each round. M_i denotes a 32-bit block of the message input, and K_i denotes a 32-bit constant, different for each operation. $\lll_s$ denotes a left bit rotation by s places; s varies for each operation. $\boxplus$ denotes addition modulo 2^{32}.

The main MD5 algorithm operates on a 128-bit state, divided into four 32-bit words, denoted *A*, *B*, *C* and *D*. These are initialized to certain fixed constants. The main algorithm then operates on each 512-bit message block in turn, each block modifying the state. The processing of a message block consists of four similar stages, termed *rounds*; each round is composed of 16 similar operations based on a non-linear function *F*, modular addition, and left rotation. Figure 1 illustrates one operation within a round. There are four possible functions *F*; a different one is used in each round:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$\oplus$, $\wedge$, $\vee$, $\neg$ denote the XOR, AND, OR and NOT operations respectively.

Pseudocode

The MD5 algorithm is calculated according to this algorithm:

```
//Note: All variables are unsigned 32 bits and wrap modulo 2^32 when calculating
var int[64] r, k
```

```
//r specifies the per-round shift amounts
r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22,  7, 12, 17, 22}
r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20,  5,  9, 14, 20}
r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23,  4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21,  6, 10, 15, 21}
```

```
//Use binary integer part of the sines of integers (Radians) as constants:
for i from 0 to 63
    k[i] := floor(abs(sin(i + 1)) × (2 pow 32))
```

```
//Initialize variables:
var int h0 := 0x67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0x10325476
```

```
//Pre-processing:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit /* bit (not byte) length of unpadded message as 64-bit little-endian integer to message
```

```
//Process the message in successive 512-bit chunks:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit little-endian words w[j], 0 ≤ j ≤ 15
```

```
    //Initialize hash value for this chunk:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3
```

```
    //Main loop:
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            f := (b and c) or ((not b) and d)
            g := i
        else if 16 ≤ i ≤ 31
            f := (d and b) or ((not d) and c)
            g := (5×i + 1) mod 16
        else if 32 ≤ i ≤ 47
            f := b xor c xor d
            g := (3×i + 5) mod 16
        else if 48 ≤ i ≤ 63
            f := c xor (b or (not d))
            g := (7×i) mod 16
```

```
        temp := d
        d := c
        c := b
        b := b + leftrotate((a + f + k[i] + w[g]) , r[i])
        a := temp
```

```
    //Add this chunk's hash to result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
```

```
    h3 := h3 + d
```

```
var char digest[16] := h0 append h1 append h2 append h3 //(expressed as little-endian)
```

```
  //leftrotate function definition
  leftrotate (x, c)
      return (x << c) or (x >> (32-c));
```

Note: Instead of the formulation from the original RFC 1321 shown, the following may be used for improved efficiency (useful if assembly language is being used - otherwise, the compiler will generally optimize the above code. Since each computation is dependent on another in these formulations, this is often slower than the above method where the nand/and can be parallelised):

```
(0  ≤ i ≤ 15): f := d xor (b and (c xor d))
(16 ≤ i ≤ 31): f := c xor (d and (b xor c))
```

MD5 hashes

The 128-bit (16-byte) MD5 hashes (also termed *message digests*) are typically represented as a sequence of 32 hexadecimal digits. The following demonstrates a 43-byte ASCII input and the corresponding MD5 hash:

```
MD5("The quick brown fox jumps over the lazy dog")
 = 9e107d9d372bb6826bd81d3542a419d6
```

Even a small change in the message will (with overwhelming probability) result in a mostly different hash, due to the avalanche effect. For example, adding a period to the end of the sentence:

```
MD5("The quick brown fox jumps over the lazy dog.")
 = e4d909c290d0fb1ca068ffaddf22cbd0
```

The hash of the zero-length string is:

```
MD5("")
 = d41d8cd98f00b204e9800998ecf8427e
```

Notes

[1] RFC 1321 (http://tools.ietf.org/html/rfc1321), section 3.4, "Step 4. Process Message in 16-Word Blocks", page 5.

[2] Tao Xie and Dengguo Feng (30 May 2009). *How To Find Weak Input Differences For MD5 Collision Attacks* (http://eprint.iacr.org/2009/223.pdf). .

[3] Xiaoyun Wang and Hongbo Yu: How to Break MD5 and Other Hash Functions (http://merlot.usc.edu/csac-f06/papers/Wang05a.pdf). Retrieved December 21, 2009.

[4] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD (http://eprint.iacr.org/2004/199), Cryptology ePrint Archive Report 2004/199, 16 August 2004, revised 17 August 2004. Retrieved July 27, 2008.

[5] J. Black, M. Cochran, T. Highland: A Study of the MD5 Attacks: Insights and Improvements (http://www.cs.colorado.edu/~jrblack/papers/md5e-full.pdf), March 3, 2006. Retrieved July 27, 2008.

[6] Marc Stevens, Arjen Lenstra, Benne de Weger: Vulnerability of software integrity and code signing applications to chosen-prefix collisions for MD5 (http://www.win.tue.nl/hashclash/SoftIntCodeSign/), Nov 30, 2007. Retrieved Jul 27, 2008.

[7] Sotirov, Alexander; Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger (2008-12-30). "MD5 considered harmful today" (http://www.win.tue.nl/hashclash/rogue-ca/). . Retrieved 2008-12-30. Announced (http://events.ccc.de/congress/2008/Fahrplan/events/3023.en.html) at the 25th Chaos Communication Congress.

[8] Stray, Jonathan (2008-12-30). "Web browser flaw could put e-commerce security at risk" (http://news.cnet.com/8301-1009_3-10129693-83.html). CNET.com. . Retrieved 2009-02-24.

[9] "US-CERT Vulnerability Note VU#836068" (http://www.kb.cert.org/vuls/id/836068). Kb.cert.org. . Retrieved 2010-08-09.

[10] "NIST.gov - Computer Security Division - Computer Security Resource Center" (http://csrc.nist.gov/groups/ST/hash/policy.html). Csrc.nist.gov. . Retrieved 2010-08-09.

[11] Philip Hawkes and Michael Paddon and Gregory G. Rose: Musings on the Wang et al. MD5 Collision (http://eprint.iacr.org/2004/264), 13 Oct 2004. Retrieved July 27, 2008.
[12] Arjen Lenstra, Xiaoyun Wang, Benne de Weger: Colliding X.509 Certificates (http://eprint.iacr.org/2005/067), Cryptology ePrint Archive Report 2005/067, 1 March 2005, revised 6 May 2005. Retrieved July 27, 2008.
[13] Vlastimil Klima: Finding MD5 Collisions – a Toy For a Notebook (http://eprint.iacr.org/2005/075), Cryptology ePrint Archive Report 2005/075, 5 March 2005, revised 8 March 2005. Retrieved July 27, 2008.
[14] Vlastimil Klima: Tunnels in Hash Functions: MD5 Collisions Within a Minute (http://eprint.iacr.org/2006/105), Cryptology ePrint Archive Report 2006/105, 18 March 2006, revised 17 April 2006. Retrieved July 27, 2008.
[15] http://www.wired.com/dangerroom/2010/07/code-cracked-cyber-command-logos-mystery-solved/Code Cracked! Cyber Command Logo Mystery Solved
[16] http://eprint.iacr.org/2010/643
[17] RFC 6151, Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms
[18] M.M.J. Stevens (June 2007). *On Collisions for MD5* (http://www.win.tue.nl/hashclash/On Collisions for MD5 - M.M.J.Stevens.pdf). . "[...] we are able to find collisions for MD5 in about $2^{24.1}$ compressions for recommended IHV's which takes approx. 6 seconds on a 2.6GHz Pentium 4.".
[19] Marc Stevens, Arjen Lenstra, Benne de Weger (2009-06-16). *Chosen-prefix Collisions for MD5 and Applications* (https://documents.epfl.ch/users/l/le/lenstra/public/papers/lat.pdf). .
[20] Magnus Daum, Stefan Lucks. "Hash Collisions (The Poisoned Message Attack)" (http://th.informatik.uni-mannheim.de/People/lucks/HashCollisions/). *Eurocrypt 2005 rump session*. .
[21] Max Gebhardt, Georg Illies, Werner Schindler. *A Note on the Practical Value of Single Hash Collisions for Special File Formats* (http://csrc.nist.gov/groups/ST/hash/documents/Illies_NIST_05.pdf). .
[22] Construction of the Initial Structure for Preimage Attack of MD5 | http://www.computer.org/portal/web/csdl/doi/10.1109/CIS.2009.214
[23] Dobbertin, Hans (Summer 1996), "The Status of MD5 After a Recent Attack" (ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf) (pdf), *RSA Laboratories CryptoBytes* **2** (2): 1, , retrieved 2010-08-10, "The presented attack does not yet threaten practical applications of MD5, but it comes rather close. ....[sic] in the future MD5 should no longer be implemented...[sic] where a collision-resistant hash function is required."
[24] "Schneier on Security: More MD5 Collisions" (http://www.schneier.com/blog/archives/2005/06/more_md5_collis.html). Schneier.com. . Retrieved 2010-08-09.
[25] "Colliding X.509 Certificates" (http://www.win.tue.nl/~bdeweger/CollidingCertificates/). Win.tue.nl. . Retrieved 2010-08-09.
[26] "[Python-Dev] hashlib - faster md5/sha, adds sha256/512 support" (http://mail.python.org/pipermail/python-dev/2005-December/058850.html). Mail.python.org. . Retrieved 2010-08-09.
[27] (http://www.rsa.com:80/rsalabs/node.asp?id=2834). The quote refers to moving away from SHA-1, the de facto successor to MD5.
[28] "Researchers Use PlayStation Cluster to Forge a Web Skeleton Key" (http://blog.wired.com/27bstroke6/2008/12/berlin.html). Wired. 2008-12-31. . Retrieved 2008-12-31.
[29] Callan, Tim (2008-12-31). "This morning's MD5 attack - resolved" (https://blogs.verisign.com/ssl-blog/2008/12/on_md5_vulnerabilities_and_mit.php). Verisign. . Retrieved 2008-12-31.
[30] Forging SSL Certificates (http://www.schneier.com/blog/archives/2008/12/forging_ssl_cer.html)
[31] Yu Sasaki, Kazumaro Aoki (2009-04-16). *Finding Preimages in Full MD5 Faster Than Exhaustive Search* (http://www.springerlink.com/content/d7pm142n58853467/). Springer Berlin Heidelberg. .
[32] Steven J. Murdoch: Google as a password cracker (http://www.lightbluetouchpaper.org/2007/11/16/google-as-a-password-cracker/), Light Blue Touchpaper Blog Archive, Nov 16, 2007. Retrieved July 27, 2008.
[33] FreeBSD Handbook, Security - DES, Blowfish, MD5, and Crypt (http://www.freebsd.org/doc/en/books/handbook/crypt.html)
[34] Solaris 10 policy.conf(4) man page (http://docs.sun.com/app/docs/doc/816-5174/policy.conf-4?l=en&a=view)

References

- Berson, Thomas A. (1992). "Differential Cryptanalysis Mod 2^{32} with Applications to MD5". *EUROCRYPT*. pp. 71–80. ISBN 3-540-56413-6.
- Bert den Boer; Antoon Bosselaers (1993). *Collisions for the Compression Function of MD5*. Berlin; London: Springer. pp. 293–304. ISBN 3-540-57600-2.
- Hans Dobbertin, Cryptanalysis of MD5 compress. Announcement on Internet, May 1996. "CiteSeerX" (http://citeseer.ist.psu.edu/dobbertin96cryptanalysis.html). Citeseer.ist.psu.edu. Retrieved 2010-08-09.
- Dobbertin, Hans (1996). "The Status of MD5 After a Recent Attack" (ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf). *CryptoBytes* **2** (2).
- Xiaoyun Wang; Hongbo Yu (2005). "How to Break MD5 and Other Hash Functions" (http://www.infosec.sdu.edu.cn/uploadfile/papers/How to Break MD5 and Other Hash Functions.pdf). *EUROCRYPT*. ISBN

3-540-25910-4.

External links

- RFC 1321 *The MD5 Message-Digest Algorithm*
- RFC 6151 *Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms*
- W3C recommendation on MD5 (http://www.w3.org/TR/1998/REC-DSig-label/MD5-1_0)
- REXX MD5 test suite (http://purl.net/xyzzy/src/md5.cmd) covers MD5 examples in various RFCs (incl. errata)

Article Sources and Contributors

MD5 *Source*: http://en.wikipedia.org/w/index.php?oldid=417589317 *Contributors*: 129.128.164.xxx, Aaboelela, Aayush214, Adashiel, Ahy1, Akamad, Aleenf1, Alerante, Alex Shih, AlexanderKlink, AlistairMcMillan, Allmightyduck, Amr Ramadan, Andre Engels, AndyKali, Antaeus Feldspar, Anthony, ArchonMagnus, Arvindn, Aurumax, Az1568, BBar, Bbatsell, Bdesham, Ben-Zin, Bender235, Beta16, Bevo, Biblbroks, Bigdok, Biggins, Bkell, Blowdart, Bmschulman, BonzoESC, Boredzo, Brendandonhue, Brianhe, BrokenSegue, Bryan Derksen, CBM, CRGreathouse, CambridgeBayWeather, Canadian-Bacon, CanisRufus, CesarB, Cfeet77, Ciphergoth, Cit helper, Coffee2theorems, Comperr, Conversion script, Coolhead33, Cophus, Corti, Crobichaud, Cryptic, Css, Damieng, DancingPhilosopher, Darrien, David Eppstein, David-Sarah Hopwood, Davidgothberg, Dawnseeker2000, Dcoetzee, Deagle AP, DerekMorr, Dimawik, Djordjes, Dmccarty, Dngrogan, DocWatson42, Dolphin51, Draicone, Drake Wilson, Drichards2, Dwheeler, DéRahier, Elsendero, Elvey, EncMstr, Evercat, Evil Monkey, Eyreland, F, FQuist, Face, Faithtear, Fant, Feezo, Fr33ke, FrYGuY, Frankk74, Frazzydee, Frecklefoot, Fredrik, Garas, Gary63, Gavia immer, Gcm, Gearoid murphy, Gerbrant, Giftlite, Graham87, Graue, Greg Tyler, GregorB, GreyCat, Guoxue, Gwilbur, Haakon, Hadal, Haikz, Hash hasher, Hede2000, HonoreDB, Hoovernj, HorsePunchKid, HubertTrzewik, IRbaboon, Ikh, Imran, Inkling, Intgr, Ironmanxsl, Isidore, Islamsalah, Ivan, Ivan Akira, Ixfd64, J-Wiki, J.delanoy, J44xm, Janetmck, Janufist, Jaredwf, Jbanes, Jberkes, Jebus989, Jeffz1, Jesse Viviano, Jewbacca, Jj137, John Vandenberg, Jonelo, Jpkotta, Karada, Kelly Martin, Kernoz, KevinJr42, Kigali1, Kirachinmoku, Kirill Zinov, Knowhow, Kragen, Kricxjo, Kylu, Largesock, Lipis, LittleDan, Loadmaster, Lolden, Lowellian, Lumingz, Lunkwill, M4gnum0n, MER-C, MITalum, MageDealer, Magioladitis, Maian, Makavelimx, Manishearth, MarekMahut, Mariushm, Martin Hinks, Martin.cochran, Materialscientist, Matt Crypto, Mdd4696, Mendaliv, Michael Hardy, Michael miceli, Michaelll, Mike Rosoft, Mike1242, MikeCapone, Mikeblas, Mipadi, Mmernex, Moonrat506, Moriori, Mpnolan, Mrand, Myria, Mzacarias, Nakon, NawlinWiki, Nealmcb, Netkinetic, Nicolas1981, Nikai, Not Kenton Archer, Ntsimp, Olathe, Oli Filth, Omegatron, Omniplex, OverlordQ, Ozuma, Padsquad43, Pako, Palica, PatrikR, Paulschou, Peterl, Pgan002, Piriax, Pjf, Platonides, Pol430, Powerslide, Psychotic Spoon, Pvasudev, Quelrod, Quietbritishjim, Qwu1777, R00723r0, Rabbler, RainbowOfLight, RandyHassan, Raraoul, Rasmus Faber, Rawling, Rdancer, Relaxing, Rettetast, Rich Farmbrough, Richardcavell, S3000, SDC, SMC, ST47, STOriginaL, Sadads, Sameervijaykar, Saqib, Sasquatch, Scienceandpoetry, Sciyoshi, Sdschulze, SeL, Seanqtx, Serpentes, Seyen, Shabbirbhimani, Shaddack, ShaunMacPherson, Silsor, Simetrical, Skalman, Slash, Sligocki, Sliwers, Soku, SpeedyGonsales, Spinal007, StarBuG, Stevo2001, Strom, Superm401, Surachit, Suruena, Sus scrofa, Taimy, Tbsdy lives, Tcsetattr, Tedickey, Tero, Tetracube, The Anome, The Inedible Bulk, The Thing That Should Not Be, Theone256, Thereverendeg, Thomas Larsen, Tim Starling, Timwi, Tom harrison, TomViza, Tracer9999, Travis Evans, Unkamunka, WISo, Wellithy, WereSpielChequers, WikHead, Wikilolo, Wikinaut, WojPob, Wolftengu, Ww, Yaronf, Yurik, ZanderZ, Zarano, Zchenyu, Zeruski, Zivi03, Ztobor, Zundark, 769 anonymous edits

Image Sources, Licenses and Contributors

Image:lll.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Lll.png *License*: Public Domain *Contributors*: Matt Crypto

Image:Boxplus.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Boxplus.png *License*: Public Domain *Contributors*: Grafite, Maksim

Image:MD5.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:MD5.svg *License*: Creative Commons Attribution-Sharealike 3.0 *Contributors*: User:Surachit

License