Implementation of (MD5) Algorithm

م. م. محمد جاسم رضا

الجامعة ألمستنصريه - كلية التربية الأساسية - قسم الرياضيات

Abstract

This paper describes the (MD5) message-digest algorithm. The algorithm takes as input a message of arbitrary length and produces an output of 128-bit as "message digest" of the input . It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

This algorithm is fast on 32-bit machines, and does not require large substitution tables.

Keyword: message-digest algorithm, digital signature compressed large file, encrypted

يصف البحث خوارزمية تلخيص الرسالة مدخل الخوارزمية ذو طول متغير ومخرج الخوارزمية ذو (128 بت) . من المعتقد أنه من غير الممكن احتسابيا إنتاج رسالتين بملخص رسالة واحده , أوأنتاج رسالة لها ملخص رسالة معروف مسبقا . تفيد خوارزمية (MD5) في تطبيقات التوقيع الرقمي حيث يجب ضغط ملف كبير بشكل آمن قبل إخفاءه بواسطة مفتاح سري ضمن منظومة أخفاء مثل (RSA) . تم تنفيذ خوارزمية (MD5) بنجاح حيث هذه الخوارزمية سريعة على حاسبات (32 بت) و لاتحتاج جداول تعويض كبيره.

الكلمات المفتاحية: خوارزمية تلخيص الرسالة، التوقيع الرقمي، حفظ ملف كبير، اخفاء.

Introduction

MD5 (Message-Digest algorithm 5) is a widely used cryptographic hash function with a 128-bit 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; 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. [2]

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. [3][4]

Further advances were made in breaking MD5 in 2005, 2006, and 2007. [5]

In an attack on MD5 published in December 2008, a group of researchers used this technique to fake SSL certificate validity. US-CERT of the U. S. Department of Homeland Security said MD5 "should be considered cryptographically broken and unsuitable for further use," and most U.S. government applications will be required to move to the SHA-2 family of hash functions after 2010. [6][7]

Background

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). [10]

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. [11]

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. Their analytical attack was reported to take only one hour on an IBM p690 cluster. [3][4]

On 1 March 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger demonstrated 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 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 that can find a collision within one minute on a single notebook computer, using a method he calls tunneling. [11]

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

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 Pentium 4 processor (complexity of $2^{24.1}$). 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}).

These collision attacks have been demonstrated in the public in various situations, including colliding document files and digital certificates. [1][6][7]

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

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." [1][2]

In 2005, researchers were able to create pairs of Postscript documents and X.509 certificates with the same hash. [2]

Later that year, MD5's designer Ron Rivest wrote, "md5 and sha1 are both clearly broken (in terms of collision-resistance)," and RSA Laboratories wrote that generation products will need to move to new algorithms." [1] [8]

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. [6]

The researchers used a cluster of Sony Play station 3s at the EPFL in Lausanne, Switzerland to change a normal SSL certificate issued by Rapid SSL 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 Rapid SSL. VeriSign, the issuers of Rapid SSL certificates, said they stopped issuing new certificates using MD5 as their checksum algorithm for Rapid SSL once the vulnerability was announced. [2][4]

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). [6]

Bruce Schneier wrote of the attack that "*we* already knew that MD5 is a broken hash function" and that "no one should be using MD5 anymore." [7]

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." [6]

MD5 makes only one pass over the data, 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.

Imlementation of MD5 Algorithm

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 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.

Algorithm

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. S denotes a left bit rotation by s places; s varies for each operation. Denotes addition modulo 2^{32}.

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 Indian 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.

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.

Fig.1 illustrates one operation within a round. There are four possible functions *F*; a different one is used in each round:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

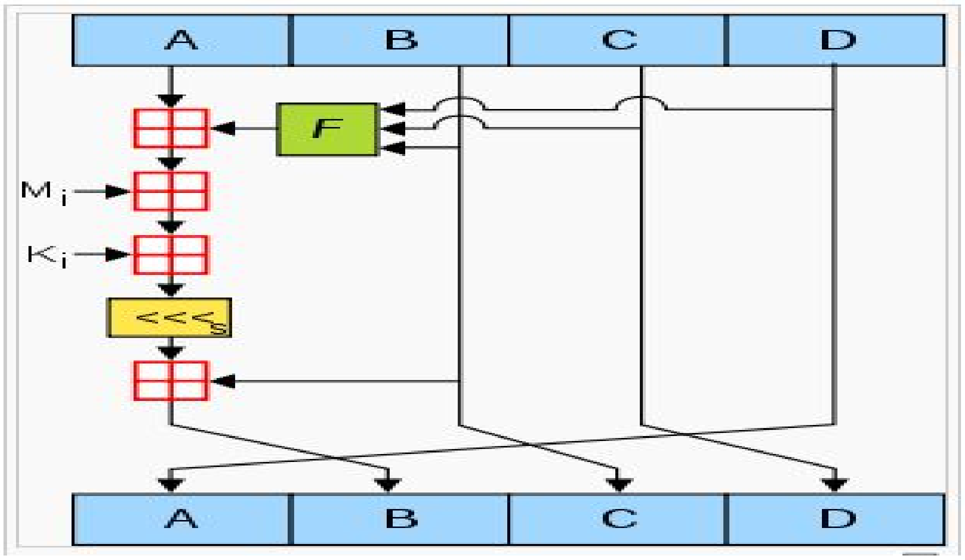


fig 1 : md5 operation within a round

Hash functions are one-way functions with as input a string of arbitrary length (the message) and as output a fixed length string (the hash value). The hash value is a kind of signature for that message. One-way functions work in one direction, meaning that it is easy to compute the hash value from a given message and hard to compute a message that hashes to a given hash value.

They are used in a wide variety of security applications such as authentication, commitments, message integrity checking, digital certificates, digital signatures and pseudo-random generators.

The security of these applications depend on the cryptographic strength of the underlying hash function. Therefore some security properties are required to make a hash function H suitable for such cryptographic uses:

P1. Pre-image resistance: Given a hash value (h) it should be hard to find any message m such that h = H(m).

P2. Second pre-image resistance: Given a message m1 it should be hard to find another message m2 != m1 such that H(m1) = H(m2).

P3. Collision resistance: It should be hard to find different messages m1, m2 such that H(m1) =H(m2).

A hash collision is a pair of different messages m1 != m2 having the same hash value H(m1) =H(m2). Therefore second pre-image resistance and collision resistance are also known as weak and strong collision resistance, respectively.

Since the domain of a hash function is much larger (can even be infinite) than its range, it follows from the pigeonhole principle that many collisions must exist. A brute force attack can find a pre-image or second pre-image for a general hash function with n-bit hashes in approximately 2n hash operations. Because of the birthday paradox a brute force approach to generate collisions will succeed in approximately 2(n/2) hash operations.

Any attack that requires less hash operations than the brute force attack is formally considered a break of a crypto graphical hash function.

Nowadays there are two widely used hash functions: MD5(17) and SHA-1(16).[12]

Both are iterative hash functions based on the Merkle-Damgard construction and using a compression function. [1] [3]

The compression function requires two fixed size inputs, namely a k-bit message block and a n-bit Intermediate Hash Value (internal state between message blocks denoted as IHV), and outputs the updated Intermediate Hash Value. In the Merkle-Damgard construction any message is first padded such that it has bit length equal to a multiple of k and such that the last bits represent the original message length.

The hash function then starts with a fixed IHV called the initial value and then updates IHV by applying the compression function with consecutive k-bit blocks, after which the IHV is returned as the n-bit hash value.

MD5 compression function

The input for the compression function MD5 Compress(IHV,B) is an intermediate hash value IHV = (a, b, c, d) and a 512-bit message block B. There are 64 steps (numbered 0 up to 63), split into four consecutive rounds of 16 steps each. Each step uses a modular addition, a left rotation, and a non-linear function. Depending on the step t, an Addition Constant ACt and a Rotation
Constant RCt are defined as follows, where we refer to Table A-1 for an overview of these values:
F(X, Y,Z) = (X ^ Y) _ (¯X ^ Z) for 0 _ t < 16

We follow the description of the MD5 compression function from [6] because its ‘unrolling’ of the cyclic state facilitates the analysis. For t = 0, 1, . . . , 63, the compression function algorithm maintains a working register with 4 state words Qt, Qt−1, Qt−2 and Qt−3.

These are initialized as (Q0,Q−1,Q−2,Q−3) = (b, c, d, a) and, for t = 0, 1, . . . , 63 in succession, updated as follows:
After all steps are computed, the resulting state words are added to the intermediate hash value and returned as output:

When you select files Fast Sum computes their checksums according to the MD5 checksum algorithm, which can be easily compared with previously computed checksums or stored for future integrity checking. There are two most important MD5 checksum properties for the end-user: the fixed size and the non-discoverability. The first property means that it does not matter how big the source file is, it always has a short 128bit MD5 checksum value that can be compactly stored and easily shared. While the second property implies that a pair of non identical files will have different MD5 checksum value i.e. even a short fixed size value is enough for detecting changes within the file.

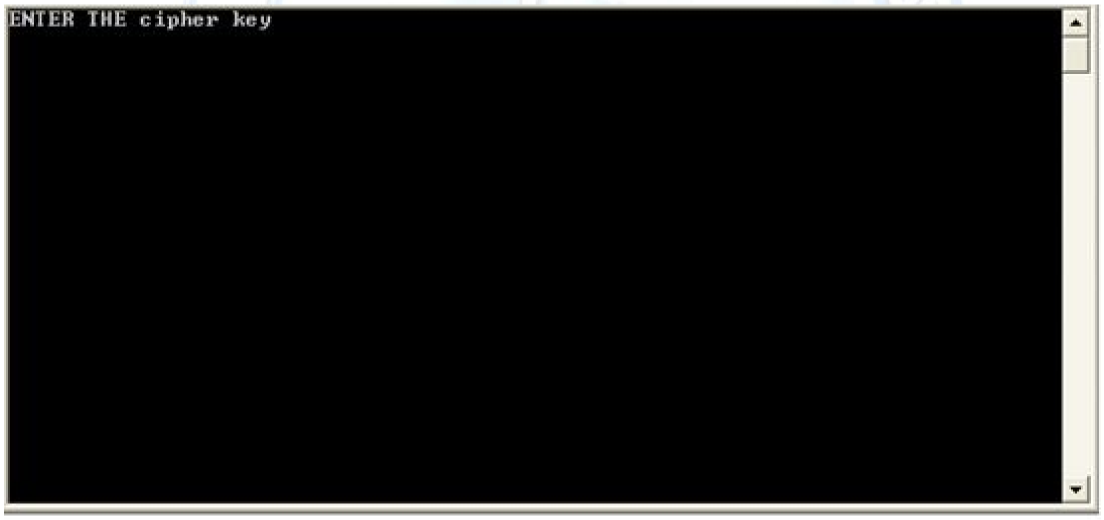
The MD5 checksum (also called as the “MD5 digest”) has been widely used in software systems besides the data integrity checking.

Result

The implementation of MD5 algorithm is done using C++ language.

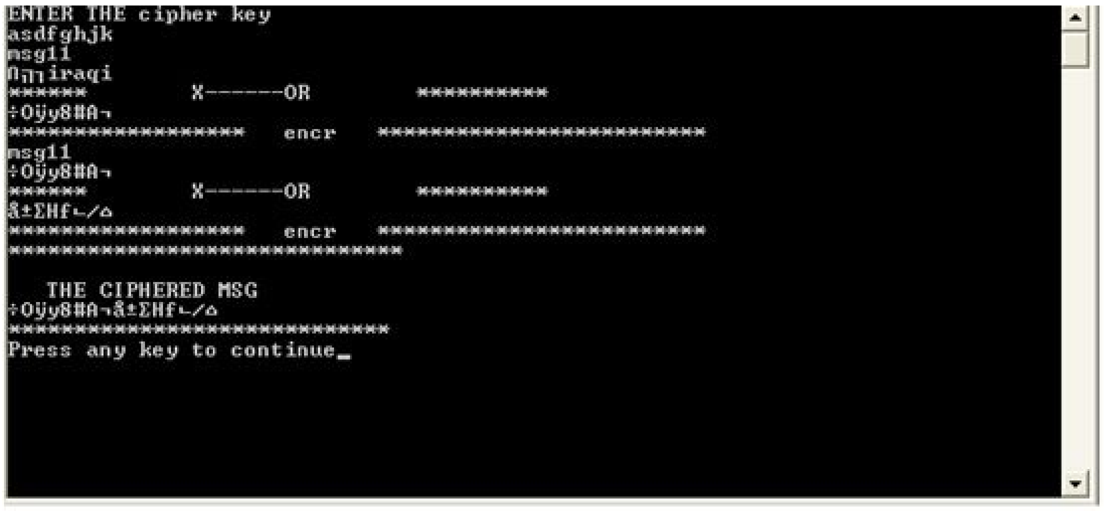
We have two programs one for encryption named md5 encryption and another for decryption named md5 decryption and text file named md5enc.txt .

The result of implementation are shown in figures (2,3,4, and 5).



```
ENTER THE cipher key
```

Fig. (2) First window.



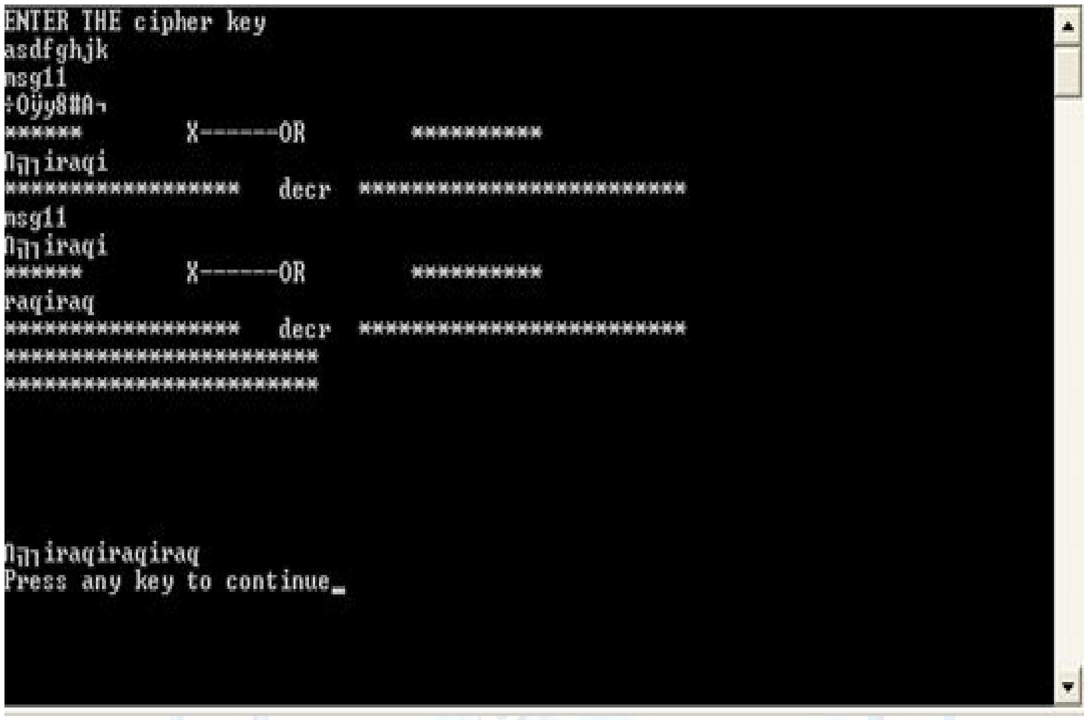
```
ENTER THE cipher key
asdfghjk
msg11
ñ╗╖iraqi
******        X------OR        **********
÷Oÿy8#A¬
******************   encr   *************************
msg11
÷Oÿy8#A¬
******        X------OR        **********
å±ΣHf└/▵
******************   encr   *************************
******************************

   THE CIPHERED MSG
÷Oÿy8#A¬å±ΣHf└/▵
******************************
Press any key to continue_
```

Fig. (3) Second window.

```
ENTER THE cipher key
```

Fig.(4) Third window.



```
ENTER THE cipher key
asdfghjk
msg11
÷Oÿy8#A¬
******        X------OR        **********
ᒉ╖╕iraqi
******************   decr   *************************
msg11
ᒉ╖╕iraqi
******        X------OR        **********
raqiraq
******************   decr   *************************
************************
************************




ᒉ╖╕iraqiraqiraq
Press any key to continue_
```

Fig. (5) Fourth window.

Conclusion

There are many means of implementing an encryption algorithm in your applications, and MD5 is one of them. You may implement a very straightforward, simple MD5 hashing algorithm directly from the user input, or you may choose to utilize a more complex salted approach. Whichever type of encryption you choose to implement, it's advised to utilize a fairly strong algorithm for sensitive data such as passwords or social security numbers when decrypting the value isn't necessary. There are times when an encrypted value must be decrypted. Under those circumstances, alternative methods are available such as RSA, SHA1, Blowfish, Rijndael (pronounced "Rain Doll" and also known as AES), and Triple DES.

References

1. Rivets, R., "The MD4 Message Digest Algorithm", RFC 1320, MIT and RSA Data Security, Inc., April 1992.
2. Rivets, R., "The MD4 message digest algorithm", in A.J. Menezes and S.A. Vanstone, editors, Advances in Cryptology - CRYPTO '90 Proceedings, pages 303-311, Springer-Verlag, 1991.
3. Colin I' Anson, Chris Mitchell, CCITT Recommendation X.509 (1988), "The Directory - Authentication Framework."
4. Jeff Doyle and Jennifer Carroll, 2005. CCIE, Professional Development Routing TCP/IP. 2nd Eden, Published by Cisco Press, pp: 936. ISBN: 1587052024
5. Johnny S.K. Wong ,Iowa State University , Benjamin D. Up off ,Los Alamos National Laboratory, ,"Securing network traffic and intrusion Detection data"
6. Axelsson. "Research in intrusion-detection systems: A survey. Technical Report 98–17", Dept. of Computer Eng. Chalmers Univ. of Tech, SE-412 96, Goteborg, Sweden.
7. Marc Myers, Intrusion Detection Preliminaries: Sanitizing Your Ecommerce Web Servers," FOCUS on Intrusion Detection: Intrusion De...: Sanitizing Your E-Commerce Web Server"
8. Casey, E, „Case study: Network intrusion investigation – lessons in forensic preparation", Digital Investigation vol 2, issue 4. December 2005. pp 254-260
9. Baker and R. Atkinson, RIP-2: MD5 Authentication. http:// portal.acm.org /citation.cfm?id=RFC2082
10. Mkyong , "Java MD5 Hashing Example" Published: February 23, 2010 , Updated: February 23, 2010
11. Benne de Wager, TU/e "Design and analysis of hash functions", 2011
12. Xiaoyun Wang and Hongbo Yu "How to Break MD5 and Other Hash Functions" Shandong University, Jinan 250100, China.