Volume 1, Issue 4, December 2013 ISSN: 2320-9984 (Online)

International Journal of Modern Engineering & Management Research

Website: www.ijmemr.org

A Review Paper of Message Digest 5 (MD5)

Alok Kumar Kasgar
Department of Information Technology
University Institute of Technology, RGPV,
Bhopal (M.P.) [INDIA]
Email :alok_kasgar@Yahoo.com

Mukesh Kumar Dhariwal
Assistant Professor
Department of Computer Science & Engineering
University Institute of Technology, RGPV,
Bhopal (M.P.) [INDIA]
Email : er.mukesh2008@gmail.com

Neeraj Tantubay
Department of Information Technology
University Institute of Technology, RGPV,
Bhopal (M.P.) [INDIA]
Email : neerajtantubay2007@gmail.com

Hina Malviya
Department of VLSI
Ram Krishna Dharmarth Foundation (RKDF)
Bhopal (M.P.) [INDIA]
Email : hinamalviya1@gmail.com

Abstract—*The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "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 a 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.*

Keyword:—Hash Function, Collision Message digest Algorithm.

1. INTRODUCTION

MD stands for 'Message Digest' and describes a mathematical function that can take place on a variable length string. The number 5 simply depicts that MD5 was the successor to MD4. MD5 is essentially a checksum that is used to validate the authenticity of a file or a string and this is one of its most common uses. Let's take a look at a working example. Let's say you have released some software or a program that you want people to freely distribute, this is all good and well but what if someone was to tamper with your application with malicious intent? For example what if they added malware onto your program, how would people know? Well if you had taken an MD5 checksum of your original program and made this information public, then when people downloaded your software could then check their downloaded file and check that the MD5 checksum matches yours. If it does then great! If not then it means your program has been tampered with. MD5, with the full name of the Message-digest Algorithm 5, is the fifth generation on behalf of the message digest algorithm. In August 1992, Ronald L.Rivest submitted a document to the IETF (The Internet Engineering Task Force) entitled "The MD5 Message-Digest Algorithm", which describes the theory of this algorithm. For the publicity and security of algorithm, it has been widely used to verify data integrity in a variety of program languages since the 1990s. MD5 was developed from MD, MD2, MD3 and MD4. It can compress any length of data into an information digest of 128bits while this segment message digest often claims to be a digital fingerprint of the data. This algorithm makes use of a series of non-linear algorithm to do the circular operation, so that crackers cannot restore the original data. In cryptography, it is said that such algorithm as an irreversible algorithm, can effectively prevent data leakage caused by inverse operation. Both the theory and practice have good security, because the use of MD5 algorithm does not require the payment of any royalties, time, and cost less which make it be widely used in the general non-top-secret applications. But even the top-secret area, MD5 may well be an excellent Intermediate technology [1]

2. MD5 ALGORITHM

MD5 stands for Message-Digest Algorithm 5. MD5 algorithm is co-invented by Rivets in MIT Computer Science Laboratory and RSA Data Security Company. MD5 is a non-reversible encryption algorithm [3]. It is widely applied in many aspects, including digital signature, encryption of information in a database and encryption of communication information. It makes large amounts of information to be compressed into a confidential format before signing the private key by digital signature soft (that is, any length byte string is transformed into a certain length of big integer). A brief description of MD5 algorithm as follows: MD5 algorithm divides plaintext input into blocks each which has 512-bit, and each block is again divided into sixteen 32-bit message words, after a series of processing, the outputs of the algorithm consist of four 32-bit message words. After these four 32-bit message words are cascaded, the algorithm generates a 128-bit hash value which is the required cipher text. Specific steps are as follows [3, 4, 5]:

(1) ***Padding-bit:*** Without loss of generality, supposes that the original data at the source has k bits (mk-1, mk-2… m0),where mi_ {0, 1}. For MD5 algorithm, its k bits data must be processed in 512-bit message block, so if the length of source is less than that length, padding is always added until its length in bits is congruent to 448modulo 512 (length≡448 mod 512). The padding consists of a single 1-bit followed by the necessary number of 0-bits.

(2) ***Padding the length of data:*** a 64-bit representation of the length on bits of the original message is appended to the result of above step. It is present by two 32-bitdigits. At this time, the length of message is filled to amultiple of 512.

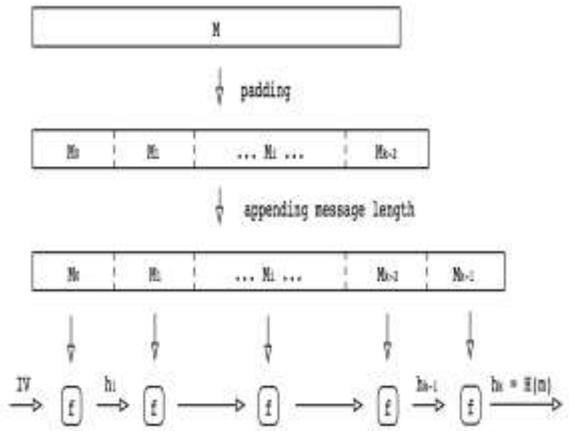

Figure 1: Working principle of an iterated hash function

(3) ***Initialize MD5 Parameters:*** four 32-bit integers A, B, C, D are called chaining variables, used to calculate themes age digest, are initialized by hexadecimal number

A=0x01234567

B=0x89abcdef

C=0xfedcba98

D=0x76543210

(4) ***Bit operation functions:*** We define four bit operation functions, F, G, H and I respectively, in which x, y, z are three 32 -bit integers. The operation is as follows:

$$F(x, y, z) = (x \wedge y) \vee ((\neg x) \wedge z) \ldots\ldots\ldots\ldots\ldots 1$$

$$G(x, y, z) = (x \wedge z) \vee (y \wedge (\neg z)) \ldots\ldots\ldots\ldots\ldots 2$$

$$H(x, y, z) = x^{\oplus} y^{\oplus} z \ldots\ldots\ldots\ldots\ldots 3$$

$$I(x, y, z) = y \oplus (x \vee (\neg z)) \ldots\ldots\ldots\ldots\ldots 4$$

In four functions, if the corresponding bits of x, y and z are independent and uniform, then each bit of the results should be independent and uniform as well. For

$$x = \sum_{i=1}^{32} x_i 2^{i-1} \in \frac{Z}{[(2)^{32})}, x^i \in \{0,1\}$$

We call x^i the i-th bit of x.

(5) ***Main transformation process:*** the number of main loop in this algorithm is the number of 512-bit information groups. The main loop have four rounds, each round carries out 16 operations, so the total of operations are64 steps. The above four chaining variables are assigned to another four chaining values: a0=A, b0=B, c0=C, d0=D. One of the chaining values is updated in each step and computation is continued in sequence. Here we have defined four rounds composite functions of main loop FF, GG, HH, and II respectively, which change from F, G, H and I. the operation is as follows:

FF →a = b + ((a + F (b c d) M_i+ t_i)<< s)……..…5
GG →a = b + ((a + H (b c d))M_i+ t_i<<s)……….6
HH →a = b + ((a + H (b c d))M_i+ t_i<<s)…..…...7
II →a = b + ((a + I (b c d)) M_i+ t_i<<s)…………8

Where, + is addition module2^{32}, Mi (0_i_15) is a 32-bit message word and the 512-bit message block is divided into 16 32-bit message words. x _ s is the left shift rotation of x by s bits. The t_i and s are step-dependent constants, ti has the following options: in i-th step, t_i is the integer part of 4294967296×abs (sin (i)),4294967296 = 2^{32}.

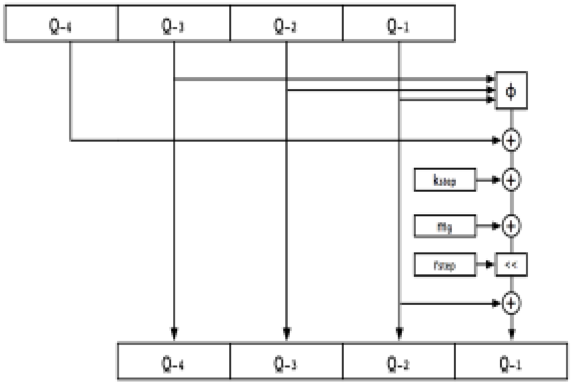

Figure 2: One step of the compression function in MD5

After all of these steps, A, B, C, D add a, b, c, d Respectively, then the algorithm is continued to run the next 512-bit message block, the final output is A, B, C, D of cascading. Application of MD5 algorithm is to generate a message digest of information in order to prevent tampering. We view the entire file as a large text message, and result in a unique MD5 message digest by the irreversible string transform method. In the future, if the contents of file are changed, we only recalculate MD5 message digest of this file, and will find the difference from the original message digest. Thereby, we can sure the checked file is incorrect.[2]

3. PSEUDO CODE FOR MD5

```
MD5 (message)

{

// Definition of two arrays with constants

r [ 6 4 ] = {7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21};

For i t = 0 . . . 6 3

 k [ i t ] = floor (abs(sin(it+1))_2ˆ32) ;

// define the initialvales IV

IV1 = 0 x67452301;

IV2 = 0 xefcdab89;

IV3 = 0 x98badcfe;

IV4 = 0 x10325476;
```

```
A = IV1; B = IV2; C = IV3; D = IV3;

// preprocess the message

append1BitToMessage (message);
While(message. length! = (448 mod 512))
AppendBitToTheMessage (message);
append64BitMessageLengthToTheMessage (message);
M[i] = splitMessageIn512BitBlocks (message);

For every M[i]
{
    Q [−4] = A;
    Q [−3] = B;
    Q [−2] = C;
    Q [−1] = D; // define step variables

    m[g] = splitMessageBlockIn16Words ( ) ; //16 x 32−bit words
    For step = 0 . . . 63
    {
        if step == 0 . . . 15 //round 1
        phi = (Q[ step −1] & Q[ step −2]) | (˜Q[step −2] & Q [step −3]) ;
        g = step;

        If step== 16 . . . 31 //round 2
        phi = (Q[step −1] & Q[ step −3]) or (˜Q[step −2] & Q [step −3]) ;
        g = (5 *step + 1) mod 16;

        If step== 32 . . . 47 //round 3
        Phi = (Q [step −1] XOR Q[step −2] XOR Q[step −3] ;
        g = (5 *step + 1) mod 6 ;

        If step==4 8 . . . 63 //round 4
        phi = Q [step −2] XOR (Q[step −1] | ˜Q[step −3] ) ;
        g = (7*step + 1) mod 1 6;

        // update stepvariable

        Q[step] = ((Q [step −4] + phi + k [step] + m[g] ) <<r [step] ) + Q[step −1] ;
    }
    // update chaining variables

    A = A + Q [60];
    B = B + Q [63];
    C = C + Q [62];
    D = D + Q [61];
}

Hash = concatenate e (A, B, C, D); // hash value of the e n t i r e input message
}
```

4. COLLISIONS AND PREIMAGES

Like every cryptographic function, hashes are susceptible to brute-force attacks. The longer L is, the more work an attacker has to do to mount an attack; however, hashes with longer L also are usually slower to computer. There are three important attacks on hashes:

- A "collision attack" allows an attacker to find two messages M1 and M2 that have the same hash value in fewer than $2^{(L/2)}$ attempts.
- A "first-pre image attack" allows an attacker who knows a desired hash value to find a message that results in

that value in fewer than 2^L attempts.

- A "second-pre image attack" allows an attacker who has a desired message M1 to find another message M2 that has the same hash value in fewer than 2^L attempts.

Other attack:

1. Brute force attack:

In cryptograph, a brute force attack or exhaustive key search is a strategy that can in theory be used against any encrypted data[20 by an attacker who is unable to take advantage of any weakness in an encryption system that would otherwise make his/her task easier. It involves systematically checking all possible key until the correct key is found. In the worst case, this would involve traversing the entire search space.

Symmetric key length vs brute force combinations

Key size in bits [2]	**Permuta-tions**	**Brute force time for a device checking 256 permutations per second**
8	28	0 milliseconds
40	240	0.015 milliseconds
56	256	1 second
64	264	4 minutes 16 seconds
128	2128	149,745,258,842,898 years
256	2256	50,955,671,114,250,100,000,000,000,000,000,000,000,000,000,000,000,000 years

2. Rainbow table:

A rainbow table is a precomputed table for reversing cryptographic hash function, usually for cracking password hashes. Tables are usually used in recovering the plain text password, up to a certain length consisting of a limited set of characters. It is a form of time-memory trade of, using less CPU at the cost of more storage. Proper key derivation function employ sal to make this attack infeasible. Rainbow tables are a refinement of an earlier, simpler algorithm by Martin Hellman[21 that used the inversion of hashes by looking up precomputed hash chains

5. WORK DONE ON MD5

MD5 is one in a series of message digest algorithms designed by Professor Ronald Rives of MI (Rivest, 1994). When analytic work indicated that MD5's predecessor MD 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 vector which produce an identical digest. At Crypto '91 Ronald L. Rivest introduced the MD5 MessageDigest Algorithm as a strengthened version of MD4, differing from it on six points. Four changes are due to the two existing attacks on the two round versions of MD4. The other two changes should additionally streng then MD5. However both these changes cannot be described as well-considered. One of them results in an approximate relation between any four consecutive additive constants. The other allows creating collisions for the compression function of MD5.

Bert den Boer and Antoon Bosselaers implemented a C program algorithm that establishes a work load offinding about 216 collisions for the first two rounds of the MD5 compression function to find a collision for the entire four round functions. On a 33MHz 80386 based PC the mean run time of this program is about 4 minutes.[9]

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- or

RIPEMD-16Hans Dobbert in proposed an attack on the compress function of MD5, which is based on similar methods as previous attacks on RIPEMD, MD4and the 256-bit extension of MD4 (see [18], [19]). Below we give a collision1of the compress function of MD5. [10]

Since the first feasible collision differential was given for MD5 in 2004 by Wang et al, a lot of work has been concentrated on how to improve it, but the researches on how to select weak input deference for MD5 collision attack are only sporadically scattered in literature. Collision resistance of several hash functions was broken by Wang et al. The strategy of determining message differential is the most important part of collision attacks against hash functions. So far, there are only three other message differentials attack published, one of which is 6 bits difference and two are 1 bit difference. [4] Wang et al. revealed the first MD5 collision with 6 bits message differences in two-block message. Later, a great many of researches greatly improved the complexity of computations and the best result is 2^{30}MD5 operations. However, no other message. Differentials are found to generate collision before 2008.

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]

The size of the hash—128 bits—is small enough to contemplate a birthday attack. MD5CR 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 collision for the full MD5 were announced by Xiaoyun Wan, Dengguo Feng, Xuejia La, and Hongbo Yu.[4][5][11] Their analytical attack was reported to take only one hour on an IBM p69 cluster.

On 1 March 2005, Arjen Lenstr, Xiaoyun Wan, and Benne de Weger demonstrated [12] construction of two X.50 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 Klimdescribed [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 RF was approved to update the security considerations in RFC 132 (MD5) and RFC 210 (HMAC-MD5). HMAC-MD5 describes a keyed-MD5 mechanism (called HMAC-MD5) for use as a message authentication code (or, integrity check value). It is mainly intended for integrity verification of information transmitted over open networks (e.g., Internet) between parties that share a common secret key. The proposed mechanism combines the (key-less) MD5 hash function [RFC-132] with a shared secret key. [17]

REFERENCES:

[1] R. Rivest. The MD5 Message-Digest Algorithm [rfc1321]

[2] Tao Xie and Dengguo Feng (30 May 2009). How to Find Weak Input Differences for MD5 Collision Attack.

[3] Rivest R L. The MD5 message digest algorithm [EB/OL].

[4] Xiaoyun Wang, Dengguo, k., m., m, HAVAL-128 and RIPEMD], Cryptology ePrint Archive Report 2004/199, 16 August 2004,

[5] J. Black, M. Cochran, T. Highland: A Study of the MD5 Attacks: Insights and Improvement, March 3, 2006

[6] Marc Stevens, Arjen Lenstra, Benne de Weger: Vulnerability of software integrity and code signing applications to chosen-prefix collisions for MD, Nov 30, 2007.

[7] Sotirov, Alexander; Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger (2008-12-30). "MD5 considered harmful today. Announce at the 25th Chaos Communication Congress.

[8] Stray, Jonathan (2008-12-30). "Web browser flaw could put e-commerce security at risk.

[9] Bert den Boer, Antoon Bosselaers. Collisions for the compression function on Md5.

[10] Hans Dobbertin. Cryptanalysis of MD5 Compress

[11] Philip Hawkes and Michael Paddon and Gregory G. Rose: Musings on the Wang et al. MD5 Collision, 13 Oct 2004.

[12] Arjen Lenstra, Xiaoyun Wang, and Benne de Weger: Colliding X.509 Certificate, Cryptology ePrint Archive Report 2005/067, 1 March 2005.

[13] Vlastimil Klima: Finding MD5 Collisions – a Toy for a Noteboo, Cryptology ePrint Archive Report 2005/075, 5 March 2005, revised 8 March 2005.

[14] Vlastimil Klima: Tunnels in Hash Functions: MD5 Collisions within a Minute, Cryptology ePrint Archive Report 2006/105, 18 March 2006, revised 17 April 2006.

[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/2010/64

[17] RFC 615, Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms

[18] H. Dobbertin, RIPEMD with two-round compress function is not collision-free, Journal of Cryptology, to appear.

[19] H.Dobbertin, Cryptanalysis of MD4, Fast Software Encryption, LNCS 1039, D.Gollmann, Ed., Springer-Verlag, 1996, pp. 53{69.

[20] Christof Paar, Jan Pelzl, Bart Preneel (2010). Understanding Cryptography: A Textbook for Students and Practitioner. Springer. p. 7. ISBN 364204100.

[21] M.E. Hellman, H.R. Amirazizi, "A Cryptanalytic Time - Memory Trade-Of," IEEE Transactions on Information Theory, vol. 34-3, pp. 505-512, 1988