ijecs open access www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3, Issue 7 July, 2014 Page No. 7257-7261

A New Improved and Secure Version of MD5

Sumita Tyagi[1], Seema Luthra[2]

[1]Babu Banarasi Das Institute of Technology, U.P Technical University, Ghaziabad
sumita.tyagi12@gmail.com

[1]Babu Banarasi Das Institute of Technology, U.P Technical University, Ghaziabad
Seema21.bajaj@gmail.com

Abstract: *In cryptography, MD5 known as Message Digest 5 was developed to overcome the cryptanalytic attacks on MD4 algorithm. Most of the hash functions are based on Merkle-Damgard construction, such as MD2, MD4, MD5, and SHA1 etc are often utilized with Digital signature algorithm, Keyed Hash Message Authentication Codes, Key Derivation Functions and Random Number Generators. MD5 takes as input any message of arbitrary length and produces as output a 128 bit message digest. Though MD5 came as a modification of MD4 to overcome its weaknesses but in 2004 some serious flaws were discovered and advances were made in breaking MD5 in 2005, 2006 and 2007[1][2].This paper presents a new algorithm which is a modification of existing MD5 algorithm to overcome the various attacks possible on MD5.*

Keywords:
MD5, SALT, Permutation Boxes, DES, Collision Attacks

1. Introduction

Cryptographic hash functions are building blocks in computer security. In 1991 Ron Rivest developed the MD5 Message Digest algorithm (RFC 1321) at MIT. MD5 was developed to overcome the weaknesses of MD4. MD5 is a widely used hash function for authentication, confidentiality and Digital Signatures of message. It is widely used in the software world to provide some assurance that a transferred file has arrived intact. It takes as input any message of arbitrary length and then process it in blocks of size 512 bits and thereafter producing an output of 128 bits usually called as message digest. The hash output depends on each bit or each character of input, that's why a small change in the message will lead to definite change in the digest value. All the hash functions are expected to satisfy some predefined properties.

1.1 Properties of Hash Function

Each message digest or hash "M" must follow the following properties:

1. M can be applied to a block of data of any size.
2. M produces a fixed-length output.
3. M (a) is relatively easy to compute for any given message a, making both hardware and software implementations easy.
4. For any given value m, it is computationally infeasible to find b such that M (b) =m. This is called as one way property.
5. For any given block a, it is computationally infeasible to find b≠a with M (b) =M (a). This is called as weak collision Property.
6. It is computationally infeasible to find any pair (a, b) such that M (a) =M (b). It is known as strong collision property.

Out of all these properties, the most important ones are 4, 5 and 6.

1.2 Difference between MD5 and MD4

1. MD4 uses three rounds of 16 steps in each round. On the other hand, MD5 uses four rounds of 16 steps each.
2. MD4 uses three primitive functions one for each of the three rounds. Whereas MD5 uses four primitive functions namely F, G, H and I, each used in a single round.
3. MD4 does not use any additive constant in the first round. In second round additive constant is common for all the 16 steps. In the third round a different additive constant is used for all 16 rounds. On the contrary, MD5 uses a single additive constant T[i] for all 64 steps.
4. In MD5, each step adds in the result of the preceding step. The step 1 result updates the content of buffer word A. The step 2 result, which is stored in D, is formed by adding A to the circular left shift result. Similarly, the step 3 result is stored in C and is formed by adding D to the circular left shift result. MD4 did not include this final addition.

1.3 Some applications of MD5

MD5 is widely used to store the passwords.MD5 and other hash functions are also used in the field of electronic document transfer, by providing a unique identifier for each document that is exchanged electronically. When employed in a digital signature application, the hash value of the message is signed instead of message itself; the receiver can use the signature to verify the signer of the message and to authenticate the integrity of the signed message.

2. Algorithm at a glance

2.1 MD5 Algorithm

In this section we will have a look of MD5 and later on we will propose the new algorithm. The algorithm as stated earlier also takes an input of arbitrary length and produces an output of 128 bit message digest. The input message is processed in blocks of 512 bits.

Phase 1: Append the Padding Bits: The original message is added on by some padding bits starting from 1 with appended 0's. The padding is done so that its length in bits is congruent to 448 modulo 512. Thus, padding makes the original message a multiple of 512 bits.

Phase 2: Append Length: The length of original message before padding is appended in the end of padding bits. This is done so that there could be a difference between padding bits and original message. The length is represented in 64 bit field.

Phase 3: Initialize MD Buffer: We use a 128 bit buffer to hold intermediate and final results of the hash function. This buffer can be represented as four 32-bit registers (A, B, C, D) having hexadecimal values stored in little endian format.
Word A=01 23 45 67
Word B=89 AB CD EF
Word C=FE DC BA 98
Word D=76 54 32 10

Phase 4: Processing the 512 bit blocks: Each 512 bit block is processed through 4 rounds. All the 4 rounds have similar structure but uses different primitive function named as F, G, H and I. Each round takes as input 512 bit message block "Mi" and ABCD 128 bit buffer updated in each round. A 64 element table called "T" is also used where T[i] =integer part of 2^{32} * abs (sin (i)) where i is in radians. Another element used is "X" where Round 1 uses X[i], Round 2 uses X [p_2i], Round 3 uses X [p_3i] and Round 4 uses X [p_4i]. Here, p_2i is (1+5i) mod16, p_3i= (5+3i) mod16 and p_4i=7i mod16.

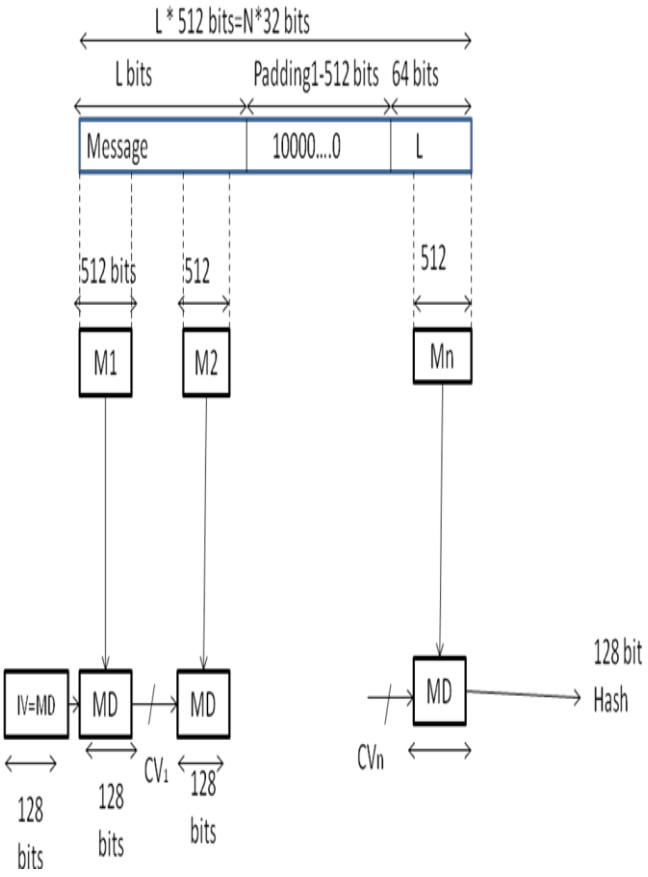

Fig. 1 General Structure of MD5

Each round in turn consists of 16 steps. We call each step as compression function. All 16 steps are similar to each other. The output of 4th round is added to the input of 1st round(CV_q) to produce output CV_{q+1}. The addition is done independently for each of the four words in the buffer with each of the corresponding words in CV_q, using addition modulo 2^{32}

Phase 5: Output: After all the 512 bit blocks M1, M2, M3 …Mn have been processed, the output is the result obtained from the last 512 block which is 128 bit message digest.

Table 1: Values of F, G, H and I

Round	***Primitive Function g***	***G(B, C, D)***
1	*F(B, C, D)*	*(B^C)V(B'^D)*
2	*G(B, C, D)*	*(B^D)V(C^D')*
3	*H(B, C, D)*	*B+C+D*
4	*I(B, C, D)*	*C+(BVD')*

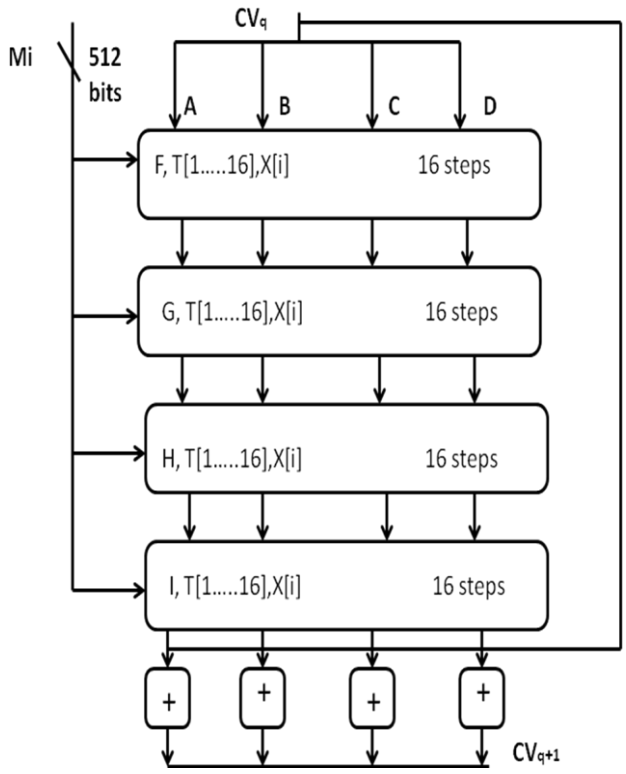

Fig. 2 Details of Compression Function MD

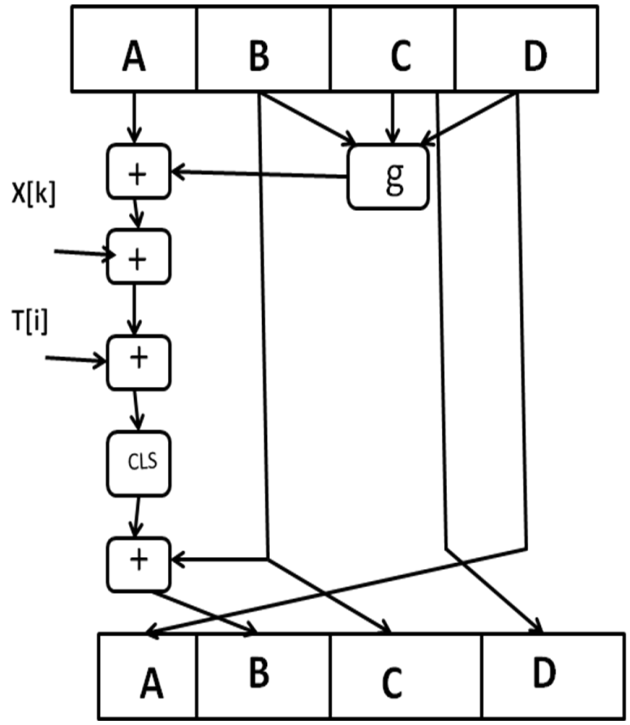

Fig. 3. Single Step of a Round Function

2.2 Attacks Possible on MD5

Various cryptanalysis attacks have been discovered on MD5 [4].In 1993, Den Boer and Bosslears gave "Pseudo collisions attack"on MD5 compression function giving identical message digests on two different initialization vectors. In 1996, Dobbertin announced a collision on compression function, but it was not on full MD5. Later on, in August 2004 attacks on full MD5 were discovered by Xiaoyun Wang. And finally in December 2010, Tao Xie and Dengguo Feng announced the first published single block MD5 collision.

3. Proposed Algorithm

In the proposed algorithm we modify MD5 by using the concept of SALT and Permutation Boxes. Once the existing algorithm is modified we get a new algorithm that produces a more secure and complex hash code as compared to the previous one.

Phase 1: Append SALT: We append or we can say that we add SALT which is variable and randomly decided by sender at the end of message to be hashed. Number of salt bits depends upon original message so as to make its total length a multiple of 512 bits. SALT is just an additional string appended to the password before it is being hashed and store into the database table. The same exact SALT will be required during verification between the user entered password and stored password.

Phase 2: Append Length: Similar to the previous algorithm simply add the length of message at the end of SALT. This is done to distinguish between the original message and SALT.

Phase 3: Initialize MD Buffer: It is also similar to the previous algorithm. We use a 128 bit buffer to hold intermediate and final results of the hash function. This buffer can be represented as four 32-bit registers (A, B, C, D) having hexadecimal values stored in little endian format.
Word A=01 23 45 67
Word B=89 AB CD EF
Word C=FE DC BA 98
Word D=76 54 32 10

Phase 4: Processing the 512 bit blocks: We apply one difference to the previous algorithm. We introduce the concept of Permutation Boxes adopted from DES algorithm. At the end of each round we add permutation boxes to make the structure of each round more complex. The security of this algorithm depends on the construction of Permutation Boxes. We need three permutation boxes. First box is added at the end of Round 1, second at the end of Round 2 and third at the end of 3rd Round. No Permutation Box is required at the end of Round 4.

Rest all is similar to previous algorithm. Each 512 bit block is processed through 4 rounds. All the 4 rounds have similar structure but uses different primitive function named as F, G, H and I. Each round takes as input 512 bit message block "Mi" and ABCD 128 bit buffer updated in each round. A 64 element table called "T" is also used where T[i] =integer part of 2^{32} * abs (sin (i)) where i is in radians. Another element used is "X" where Round 1 uses X[i], Round 2 uses X [p_2i], Round 3 uses X [p_3i] and Round 4 uses X [p_4i]. Here, p_2i is (1+5i) mod16, p_3i= (5+3i) mod16 and p_4i=7i mod16.

Each round in turn consists of 16 steps. We call each step as compression function. All 16 steps are similar to each other.

Phase 5: Output: Similar to the original MD5 the output is obtained from the processing of the last 512 bit block taking input as last message block M_n and the output of the previous block M_{n-1} after the application of compression function.

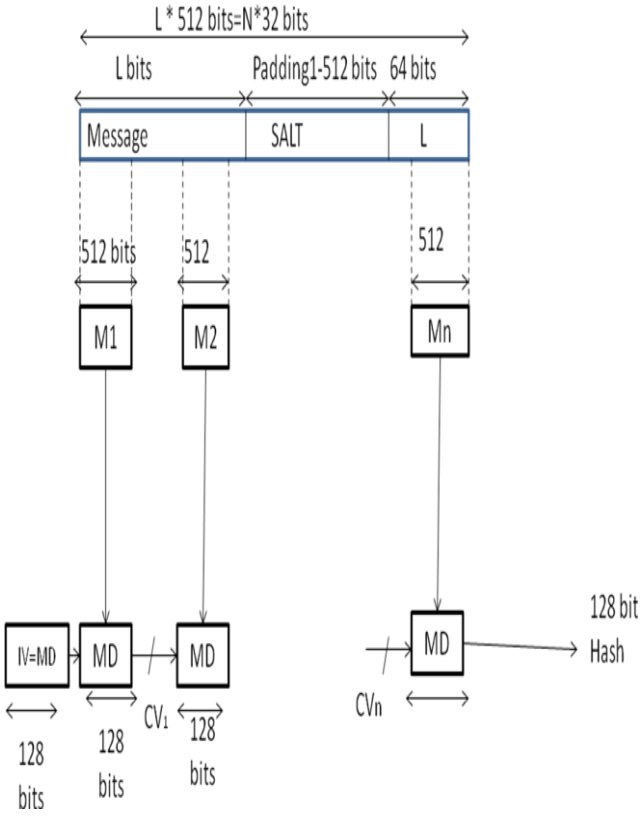

Fig. 4. General Structure of Modified MD5

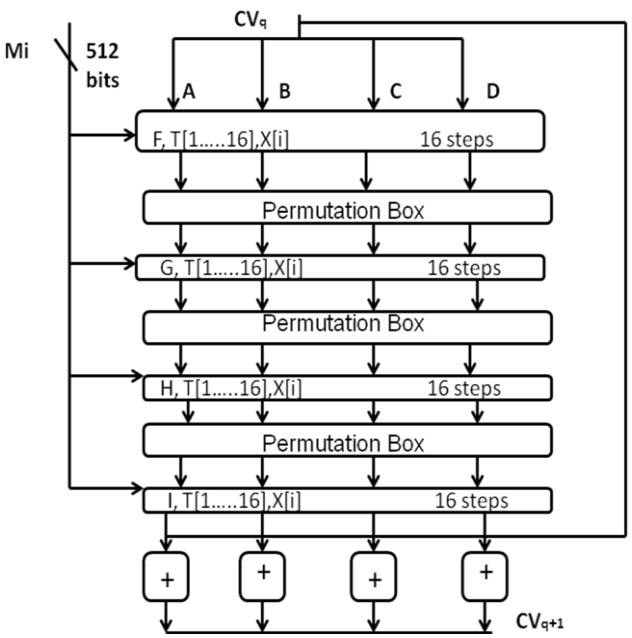

Fig. 5. Compression Function of Modified MD5

4. Conclusions

MD5 and other hash functions are suffering from many security loop holes. These hash functions are prone to Birthday Attack [5] and many other well know collision attacks [2], [4]. Some of the new designs and changes have been implemented to make MD5 and other hash functions more secure [3], [6].MD5 was prone to many attacks and thus to make it more secure we have added the concept of SALT and Permutation Boxes. The security of this new proposed algorithm depends on the construction of Permutation Boxes. SALT is a random generated bits added to the end of text to be hashed. The concept of SALT provides an added advantage of changing the value of message digest by simply making a few changes in the SALT. Adding Permutation Boxes is a concept from DES. We have added Permutation Boxes after every 16 steps. This version of MD5 is more complex and thus more secure than the original MD5 message digest.

References

[1] Xiaoyun Wang, Dengguo Feng, Xeujia Lia, Hongbo Yu: "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", Cryptology ePrint Archive Report 2004/199, 16 august 2004, revised 17 August 2004. Retrieved July 27, 2008.

[2] Tao Xie and Dengguo Feng "How to find weak input Differences for MD5 Collision Attacks", ePrint, IACR, 2009

[3] S.Al Kuwari, J.H. Davenport, R.J. Bradford. Cryptographic Hash Functions: Recent Design Trends and Security Notations. In short paper proceedings of

Inscrypt'10. Science Press of China, 2010, pp- 133-150.
[4] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, Cryptanalysis of the Hash Functions MD4 and RIPEMD, EUROCRYPT, pp. 1–18, 2005.
[5] Erika Batista, Gaël Canal, Karim Ziadeh: The Birthday paradox Operational Research and Optimization, December 2012.
[6] Thulasimani Lakshmanan and Madheswaran Muthusamy, A Novel Secure Hash Algorithm for Public Key Digital Signature Schemes, The International Arab Journal of Information Technology, Vol. 9, No. 3, May 2012

Author Profile

Sumita Tyagi received the B.Tech Degree from U.P Technical University from in 2006 and MBA from IMT in 2009. Currently working as Assistant Professor in BBDIT, Ghaziabad having more than 7 years of academic experience. She has authored a book on "Cryptography & Network Security" and published papers in many national and international journals. Areas of interest are Cryptography, Algorithms and Programming.

Seema Bajaj received M.Tech Degree from Banasthali Vidyapith in 2006. Currently working as Associate Professor in BBDIT, Ghaziabad having more than 8 years of academic experience. She has published papers in many national and international journals. Areas of interest are Cryptography, Programming, Database and Software Engg.