Jayeeta Majumder / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 2, Issue 3, May-Jun 2012, pp.721-724

Dictionary Attack on MD5 Hash

Jayeeta Majumder
Department of Computer Science and Engineering
Haldia Institute of Technology
Haldia, West Bengal

Abstract
MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input .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. Message Digest 5 is commonly used to create hash of passwords that is an encrypted form of password to be sent over network or stored in file system. Through out this paper we have gone through different mechanism of md5. Throughout this thesis we have analyzed several password-cracking techniques and compare them for exploiting MD5 hashed passwords. We have gone through cryptanalysis of MD5 based on dictionary attack.

Keywords : MD5 Hash, Dictionary Attack, MD5_DRIV algorithm, MD5_IMPL algorithm

1. Introduction
A hash function H maps an arbitrary input message m to a fixed length output called hash value h = H (m). Hash functions can basically be divided in two groups, keyed hash functions and
unkeyed hash functions. An example for keyed hash functions are Message Authentication Codes MACs. MACs should ensure the source of a message and its integrity. As the input of these hash functions are the message m and a particular key k, one needs to know the key k to compute the correct hash value h = H (m, k). Hence, without the knowledge of the key k it should be computationally infeasible to find a message m^* such that H (m^*, k) = h.

A subclass of un keyed hash functions are Modification Detection Codes MDCs. Contrary to MACs, we do not need a special key to compute the hash value h = H(m). Nevertheless, it should be computationally infeasible to find two messages $m \neq m^*$ such that H (m) = H (m^*, k) .

A hash function with n bits output yields in 2^n possible output hash values. Due to the birthday paradox, we only have to compute $2^{n/2}$ different input messages to find a collision with a certain probability. If we are able to find a collision in less than $2^{n/2}$ the hash function is considered to be broken. Most hash functions used in practice are designed as iterated hash functions. The arbitrary input message m is split in k fixed length blocks mi. Then, a so called compression function f is applied to every single message block mi and the result of the previous message block mi−1. For the application of the compression function to the first message block m0, we use a constant predefined initial value IV.

$$h_i = f(m_i, h_{i-1})\big|_{i=0}^{k} \text{ with } h_0 = IV,\ H(m) = h_k.$$

An example for an unkeyed hash function is MD5 [2] . As the successor of MD4 it was proposed by Ronald L. Rivestin 1992 because weaknesses were found in MD4 [16]. The hash function MD5 is designed as an iterated hash function and computes a 128-bit hash value out of arbitrary length messages. Today it is widely used in SSL/TLS, IPSec, and on UNIX systems to store passwords [2]. It would be a security weakness if passwords, for instance on servers, were stored in plain text. Therefore just the hash values of the passwords are stored in a public file. If a user wants to log in, the hash function is applied to the entered password. Then the computed hash value is compared to the one in the password file and the user is allowed to log in or not.

2. Overview of MD5
Message Digest 5 (MD5) hash was developed by Rivest as an update to his previous MD4[16] hash and published in 1992 [2]. MD5, like other cryptographic hash algorithms, takes a message of arbitrary size and produces an output of fixed size (128 bits).

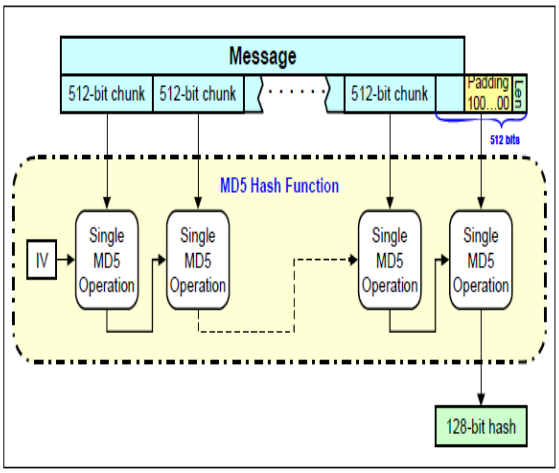

Figure – Illustration of MD5 Hash Function

Fig 1: Illustration of MD5 Hash Function

3. MD5 Hash Algorithm

MD5 is a hash function that takes an arbitrary message as input and generates a 128-bit output digest. The initial values (IV1, IV2, IV3, IV4) and the chaining variables (A, B, C, D) are four 32-bit values. In the following, we describe the algorithm with message pre-processing, the application of the compression function, and the final hash value h.

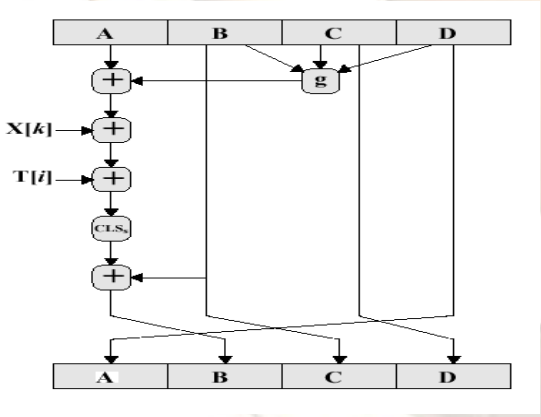

fig 2. : Message pre-processing

At first, we append a binary 1 to the message and then binary 0s unless the message length is 448 (mod 512). Then, we append the 64-bit little endian message length to this message. We now have an input message that has a length of 0 (mod 512). The pre-processed message now looks like $M = (M_0 , M_1 , \ldots , M_{k-1})$ and the compression function can be applied to every message block.

4. Active attacks on MD5

4.1 Brute Force Attack

The attacker tries every possible key on a piece of cipher text [2] until an intelligible translation into plaintext is obtained. On average, half of all possible keys to be tried achieve success.

4.2. Chosen-Prefix Collisions Attack

We present a novel, automated way to find differential paths for MD5. As an application we have shown how, at an approximate expected cost of 2^{29} calls to the MD5 compression function, for any two chosen message prefixes P and P' suffix s and S and S' can be constructed such that the concatenated values P OR S and P′OR S' collide under MD5. The practical attack potential of this construction of chosen-prefix collisions [15] is of greater concern than the MD5-collisions that were published before. This is illustrated by a pair of MD5-based X.509 certificates one of which was signed by a commercial Certification Authority (CA) as a legitimate website certificate, while the other one is a certificate for a rogue CA that is entirely under our control.

Given two arbitrarily chosen messages, we first apply padding to the shorter of the two, if any, to make their lengths equal. This is unavoidable, because Merkle-Damgard strengthening, involving the message length, is applied after the last message block has been compressed by MD5[2]. We impose the additional requirement that both resulting messages are a specific number of bits (such as 64 or 96) short of a whole number of blocks. In principle this can be avoided, but it leads to an efficient method that allows relatively easy presentation. All these requirements can easily be met, also in applications with stringent formatting restrictions.

Given this message pair, we modify a suggestion by Xiaoyun Wang (private communication) by finding a pair of k-bit values that, when appended to the last incomplete message blocks, results in a specific form of difference vector between the IHVs after application of the MD5 compression function to the extended message pair. Finding the k-bit appendages can be done using a birthday procedure.

4.3. Dictionary Attack

A dictionary attack [15] refers to the general technique of trying to guess some secret by running through a list of likely possibilities, often a list of words from a dictionary [5]. It contrasts to a brute force attack in which all possibilities are tried. The attack works because users often choose easy to guess passwords, even after being exhorted against doing so. Dictionary attacks either precompute hash values for a given dictionary file and then compare target password hashes to the pre computed tables for a match, or words in a dictionary file are hashed and compared against a target password hash for a match.

5. Algorithm Specification

In this paper first I have create a hash value of a particular message. After that we have taken all dictionary Word to create all hash value of the corresponding hash value then match hash value with our target hash value.

5.1 Algorithm :

Algorithm : MD5_DRIV

1. **Open source file** SOURCE.TXT
2. **Argument passing from source file to MD5_IMPL .**
3. **Hash Generation .**
4. **Hash code store in RESULT.TXT file as output .**

Algorithm: MD5_IMPL

MD5 hash function:
m=m0m1m2….ms-1

1. Construct M=M[0]M[1]…M[N-1]
2. A←67452301
3. B←EFCDAB89
4. C←98BADCFE
5. D←10325476
6. For i=0 to N/16 -1 Do
7. For j=0 to 15 do
8. X[j]=M[i*16+j]
9. A′ ← A
10. B′← B
11. C'← C
12. D′← D
13. Round1(Algo1)
14. Round1(Algo2)
15. Round1(Algo3)
16. Round1(Algo4)
17. A← A+ A'
18. B← B+ B′
19. C← C+ C'
20. D← D+ D′
21. end for;
22. end for;
23. h(m)= A OR B OR C OR D
24. end

5.2 Complexity Analysis

Algorithm : MD5_DRIV

1. **Open source file** SOURCE.TXT
2. **Argument passing from source file to MD5_IMPL .**
3. **Hash Generation .**
4. **Hash code store in RESULT .TXT file as output .**

⟹ **O(1)**

Algorithm : MD5_IMPL

MD5 hash function:
m=m0m1m2….ms-1

1. Construct M=M[0]M[1]…M[N-1]
2. A←67452301
3. B←EFCDAB89
4. C←98BADCFE
5. D←10325476

⟹ **O(1)**

6. For i=0 to N/16 -1 Do
7. For j=0 to 15 do
8. X[j]=M[i*16+j]

⟹ **16n +c**

9. A′ ← A
10. B′← B
11. C'← C
12. D′← D

⟹ **O(1)**

13. Round1(Algo1) ⟹ **16*c1**
14. Round1(Algo2) ⟹ **16*c2**
15. Round1(Algo3) ⟹ **16*c3**
16. Round1(Algo4) ⟹ **16*c4**
17. A← A+ A'
18. B← B+ B′
19. C← C+ C'
20. D← D+ D′

O(1)

21. end for;
22. end for;
23. h(m)= A OR B OR C OR D
24. end

Total Complexity = 0(1) + 0(1) + (16n+c) +(16*c1+16*c2+16*c3+16*c4) + 0(1)
= **O(n)**

6. References

[1] B.Schneier. *Applied Cryptography*. John Wiley and Sons, second edition, 1996
[2] William Stallings, *Cryptography and Network Security-Principles and Practice*, Third edition, Prentice Hall publications 2004.
[3] Wikipedia http://www.wikipedia.org
[4] WordIQ http://www.wordiq.com/
[5] Trappe and Washington, University of Maryland, *Introduction to Cryptography with Coding Theory*, Prentice Hall, 2002.
[6] Sun Microsystems http://research.sun.com/projects/crypto/
[7] Certicom Corp. http://www.certicom.com/
[8] S. Lucks and M. Daum, *The Story of Alice and her Boss*. Presented at the rump session of Eurocrypt '05, May 2005
[9] X. Wang and H. Yu, *How to Break MD5 and Other Hash Functions.*
[10] Barry M. Leiner and Vinton G. Cerf and David D. Clark and Robert E. Kahn and Leonard Kleinrock and Daniel C. Lynch and Jon Postel and Larry G. Roberts and Stephen Wolff. http://www.isoc.org/internet/history/brief. 9 January, 2006.

[11] National Institute of Standards and Technology (NIST) , Computer Systems Laboratory. Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB) 180-2, August 2002.

[12] Praveen Gauravaram and Adrian McCullagh and Ed Dawson. The legal and practical implication of recent attacks on 128-bit hash functions. First MondayJournal,volume11.Number12,January,2006.http://www.firstmonday.org/issues/issue11/gauravaram/index.html.

[13] National Institute of Standards and Technology (NIST) Computer Systems Laboratory. Secure hash standard. Federal Information Processing Standards Publication (FIPS PUB) 180-1, April 1995.

[14] Adrian McCullagh and William Caelli. Non-Repudiation in the Digital Environment. First Monday–Peer-Reviewed Journal On the Internet, 5, August 2000. This article is available at http://www.firstmonday.dk/issues/issue5 8/mccullagh/ Last access date: 5 January 2006.

[15] Ronald Rivest. The MD5 Message-Digest Algorithm. RFC 1321, MIT, RSA Data Security, April 1992.

[16] Hans Dobbertin. Cryptanalysis of md4. In Fast Software Encryption, pages 53–69,1996. http://world.std.com/~franl/crypto.html