The MD5

Umang Beri
University of Rochester

The MD5 (Message Digest) is a hash developed by Ron Livest to create secure signatures on the Internet. This was in 1990 when its predecessor, the MD4, was unceremoniously broken down and shown as a vulnerable algorithm. This paper looks at the weaknesses of the MD5, and why it has been seen, by security experts, not as secure as it seems.

Introduction

The initial motive for this paper was to create a demonstration of how, according to T.A Berson, the MD5's first step can be broken down via Crytoanalysis. Unfortunately, understanding the algorithm itself was a daunting task and this paper was finally stepped down to understanding the different ways of attacking the MD5.

Ron Rivest

Ron Rivest is currently a professor at the MIT (Andrew and Erna Viterbi Professor of Electrical Engineering and Computer Science), and has developed 3 different versions of the Message Digest algorithms. The MD4 was the first of them but was found to be weak and thus came the MD5 and the MD2, both around the same time. The MD5 is the faster and more popular hash used than the MD2, which is more secure, but slower to run.

He is a founder of RSA Data Security which has now merged with Security Dynamics to run as RSA Security.

MD5

The Internet hit the global community to make it part of everyday life. It is the one portal that information can be shared with others through free or paid access. The information could be public, such as open source technology, or even private data such as small as personal email accounts to something as large as running an online Role Playing Game for over 100,000 simultaneous users. To secure certain information, such as images or paid accounts, different methods were implemented.

The MD5 was also created for various languages, some being in-built functions, such as in PHP 4.0 and soon to be released PHP 5.0. The MD5 takes an input of a message of arbitrary length and produces an output of 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 pre-specified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" [1] in a secure manner before being encrypted with a private (secret) key under a public-key crypto-system such as RSA.

License rights to use the MD5:

The MD5, along with the MD2, and MD4 are all licensed and are intellectual properties of the RSA. For permission to use them, the RSA need not be contacted as per the criteria posted on Ron Rivet's personal website at this URL:
http://www.ietf.org/ietf/IPR/RSA-MD-all

The MD5 is a one way hash. That means the hash can only be used for confirming if the string is true and cannot look up what the string's content was after it has been hashed.

Where H is hashing the string, and x is the string input, h is the hash output

$$H(x) = h$$
$$DH(h) \neq x$$

The creation of the MD5 was with five objectives:

i. Security: there are no two hashes with the same value
ii. Direct Security: there are no assumptions
iii. Speed: based on bit manipulations on 32 bit operands
iv. Simplicity: not a complicated program to code, there are no substitution tables
v. Little Endian Architecture: Optimized for microprocessor architecture

The MD5 is a hard hash as defined by Samuel S. Wagstaff, Jr in *Cryptoanalysis of Number Theoretic Ciphers.* It defines the four criteria for being a hard hash:

1. Given M we can compute $h(M)$
2. Given $H(o)$ it is hard to compute any M for $H(M) = H(o)$
3. Given M it is hard to find $M' \neq M$ for which $H(M') = H(M)$
4. There are no two messages where $M' \neq M$ for which $H(M') = h(M)$

The MD5 works via 512-bit blocks, which are divided into 16 32-bit sub blocks.

It starts off by first padding the length so that there is 512-64bits used up, by adding a 1 followed by the required number of 0s. At least one bit, and at most 512 bits are appended. At the end of it, 64 bits are reserved for the length of the input string.

Four 32-bit chaining variables are initialized as well (Rivest refers to them as 'buffers' in his paper *The MD5 Message Digest Algorithm)*. The buffers (A, B, C, and D) are defined as:

A: x01 23 45 67
B: x89 ab cd ef
C: xfe dc ba 98
D: x76 54 32 10

Notice the buffers are reversals of each other. These buffers are also known as chaining variables [2]. The chaining variables are followed by a loop, which has four rounds.

F(X,Y,Z)	*= (X OR Y)*	*AND*	*((NOT X) AND Z)*
G(X,Y,Z)	*= (X AND Z)*	*OR*	*(Y AND (NOT Z))*
H(X,Y,Z)	*= X XOR Y*	*XOR*	*Z*
I(X,Y,Z)	*= Y*	*XOR*	*(X OR (NOT Z))*

The truth table for the above functions are as follows:

X	Y	Z	f(F)	f(G)	f(H)	f(I)
0	0	0	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	1	1	0
0	1	1	1	0	0	1
1	0	0	0	0	1	1
1	0	1	0	0	0	1
1	1	0	1	1	0	0
1	1	1	1	1	1	0

All four functions are independent, they do not rely on each other, and based on truth table, we are not guaranteed a consistent output (i.e. there is no tautology).

Also: If *X, Y* and *Z* are independent and unbiased, then so are *F(X,Y,Z), G(X,Y,Z), H(X,Y,Z)* and *I(X,Y,Z).* Each of the chaining variables are linked to a four unique rounds in this loop, specified as *FF, GG, HH, II.*

Key	$FF(a,b,c,d,Mj,s,t_i)$ $\rightarrow a = b + ((a + F(b,c,d)+Mj+ t_i) <<< s)$ $GG(a,b,c,d,Mj,s,t_i)$ $\rightarrow a = b + ((a + G(b,c,d)+Mj+ t_i) <<< s)$ $HH(a,b,c,d,Mj,s,t_i)$ $\rightarrow a = b + ((a + H(b,c,d)+Mj+ t_i) <<< s)$ $II(a,b,c,d,Mj,s,t_i)$ $\rightarrow a = b + ((a + I(b,c,d)+Mj+ t_i) <<< s)$
Mj – where it represents the *j*th sub-lock of the message (0 to 15).	
<<< s – is a left circular shift of *s* bits	
A, b, c, d represent the chaining variables	

Round 1 (FF)

FF (a, b, c, d, Mo, 7, 0xd76aa478)
FF (a, b, c, d, M1, 12, 0xe8c7b756)
FF (a, b, c, d, M2, 17, 0x242070db)
FF (a, b, c, d, M3, 22, 0xc1bdceee)
FF (a, b, c, d, M4, 7, 0xf57c0faf)
FF (a, b, c, d, M5, 12, 0x4787c62a)
FF (a, b, c, d, M6, 17, 0xa8304613)
FF (a, b, c, d, M7, 22, 0xfd469501)
FF (a, b, c, d, M8, 7, 0x698098d8)
FF (a, b, c, d, M9, 12, 0x8b44f7af)
FF (a, b, c, d, M10, 17, 0xffff5bb1)
FF (a, b, c, d, M11, 22, 0x895cd7be)
FF (a, b, c, d, M12, 7, 0x6b901122)
FF (a, b, c, d, M13, 12, 0xfd987193)
FF (a, b, c, d, M14, 17, 0xa679438e)
FF (a, b, c, d, M15, 22, 0x49b40821)

GG (a, b, c, d, M1, 5, 0Xf61e2562)
GG (a, b, c, d, M6, 9, 0xc040b340)
GG (a, b, c, d, M11, 14, 0x265e5a51)
GG (a, b, c, d, M0, 20, 0xe9b6c7aa)
GG (a, b, c, d, M5, 5, 0xd62f105d)
GG (a, b, c, d, M10, 9, 0x02441453)
GG (a, b, c, d, M1, 14, 0xd8a1e681)
GG (a, b, c, d, M4, 20, 0xe7d3fbc8)
GG (a, b, c, d, M9, 5, 0x21e1cde6)
GG (a, b, c, d, M14, 9, 0xc33707d6)
GG (a, b, c, d, M3, 14, 0xf4d50d87)
GG (a, b, c, d, M8, 20, 0x455a14ed)
GG (a, b, c, d, M13, 5, 0xa9e3e905)
GG (a, b, c, d, M2, 9, 0xfcefa3f8)
GG (a, b, c, d, M7, 14, 0x676f02d9)
GG (a, b, c, d, M12, 20, 0x8d2a4c8a)

HH (a, b, c, d, M5, 4, 0xfffa3942)
HH (a, b, c, d, M8, 11, 0x8771f681)
HH (a, b, c, d, M11, 16, 0x6d9d6122)
HH (a, b, c, d, M14, 23, 0xfde5380c)
HH (a, b, c, d, M5, 4, 0xa4beea44)
HH (a, b, c, d, M8, 11, 0x4bdecfa9)

HH (a, b, c, d, M11, 16, 0xf6bb4b60)
HH (a, b, c, d, M14, 23, 0xbebfbc70)
HH (a, b, c, d, M5, 4, 0x289b7ec6)
HH (a, b, c, d, M8, 11, 0xeaa127fa)
HH (a, b, c, d, M11, 16, 0xd4ef3085)
HH (a, b, c, d, M14, 23, 0x04881d05)
HH (a, b, c, d, M5, 4, 0xd9d4d039)
HH (a, b, c, d, M8, 11, 0xe6db99e5)
HH (a, b, c, d, M11, 16, 0x1fa27cf8)
HH (a, b, c, d, M14, 23, 0xc4ac5665)

II (a, b, c, d, Mo, 6, 0xf4292244)
II(a, b, c, d, M7, 10, 0x432aff97)
II(a, b, c, d, M14, 15, 0xab9423a7)
II(a, b, c, d, M5, 21, 0xfc93a039
II (a, b, c, d, Mo, 6, 0x655b59c3)
II(a, b, c, d, M7, 10, 0x8f0ccc92)
II(a, b, c, d, M14, 15, 0xffeff47d)
II(a, b, c, d, M5, 21, 0x85845dd1)
II (a, b, c, d, Mo, 6, 0x6fa87e4f)
II(a, b, c, d, M7, 10, 0xfe2ce6e0)
II(a, b, c, d, M14, 15, 0xa3014314)
II(a, b, c, d, M5, 21, 0x4e0811a1)
II (a, b, c, d, Mo, 6, 0xf7537e82)
II(a, b, c, d, M7, 10, 0xbd3af235)
II(a, b, c, d, M14, 15, 0x2ad7d2bb)
II(a, b, c, d, M5, 21, 0xeb86d391)

Attacking the MD5

The MD5 is a hash, and to date, there have been two ways of attacking it, one of them in a limited capacity via cryptoanalysis and the other through collisions.

Attack using cryptoanalysis

Thomas A. Berson believed he could attack the MD5 via cryptoanalysis. In his report [3], he created a theory for differential cryptanalysis of the circular shift function (>>>s) and attempted it on the four rounds of MD5.

The idea was to find two messages where

$$m \neq m^* \text{ and yet } MD5(m) = MD5(m^*).$$

According to Rivest and Dusse, this would take 2^64 operations. Berson thought it was more feasible to test it on the mod 2^32 as there is no current available computer memory to simulate this (Berson said this back in 1991, but this holds true even today). In fact, M. Leech in his recent journal (Sept, 2003) mentioned that a 64bit MD5 takes an average of 218.04 days to get [4].

Steps:

- *if* $x + c = z$
- *and if* $x^* + c = z^*$
- *then* $z' = z - z^* = x - x^* = x' \rightarrow$ *Probability 1*
- *if* $x + y = z$

- *and if* $x^* + y^* = z^*$
- *then* $z' = z - z^* = x - x^* + y - y^* = x' + y' \rightarrow$ *Probability 1*
- $x' = x - x^* \bmod n\ ;\ 0 \leq x' \leq n-1$

Theorem 1:
- *if i belongs to Z and x belongs to R, then Lbound (I+x) = i + Lbound (x)*

Theorem 2:
- *if a, b, m belongs to Z*
- *Lbound (a/m) – Lbound ((a-b)/m) is*
  - *Lbound (b/m) + 1;* $a \bmod m < b \bmod m$
  - *Lbound (b/m);* $a \bmod m \geq b \bmod m$

x = *32 bit word*
$2^{\wedge}_k x$ = *left shift of x by k places, 0 filled on the right*
Lbound(x/m) = *right shift of x by 32-k places with 0 filled on left*

$CLS_k(x) = 2^{\wedge}_k x + Lbound(x/m) \bmod n$

$z' = CLS_k(x) - CLS_k(x^*) \bmod n = CLS_k(x) - CLS_k(x - x' \bmod n) \bmod n$

$\rightarrow 2^{\wedge}k + Lbound\ (x/m) - 2^{\wedge}k\ (x - x' \bmod n) - Lbound\ ((x - x' \bmod n) / m) \bmod n$
$\rightarrow 2^{\wedge}k + Lbound\ (x/m) - Lbound\ ((x - x' \bmod n) / m) - Lbound\ ((x - x' \bmod n) / m) \bmod n$

The above two are run through two cases:
- $x \geq x' \rightarrow x - x' \geq 0 \rightarrow x - x' \bmod n = x - x'$
  - *occurs when x takes n – x' values* $(x = x', x' +1, x'+2 \ldots\ldots n-1)$
- $x < x' \rightarrow x - x' < 0 \rightarrow x - x' \bmod n = x - x' + n$
  - *occurs when x takes x' values* $(x = 0, 1, 2 \ldots\ldots n-1)$

Case 1:
- $x \geq x'$
- $x = q1m + r1$
- $x' = q2m + r2$
- $0 \leq r1$
- $r2 < m$
- *Lbound (x/m) = q1* *(i)*
- *Lbound (x'/m) = q2* *(ii)*

(i) – (ii) = q1 – q2 + Lbound ((r1 – r2) / m) *{apply Thm. 2}*

- *occurance is: r2 (2^k – q2 – 1) times*

Case 2:
- *x and x' same as in Case 1*
- *Lbound (x/m) – Lbound ((x – x' + n) / m) mod n* *(i)*
- *Where n/m = 2^k*

If (r – r')/ m = 0; then we use q2 – 2^k
If (r – r')/m = -1; then we use q2 – 2^k +1

We need to find the occurrence values of r1 < r2

The above tools are applied to the MD5 as such

We need to find two message blocks from Ma to Ma where FF(Ma) = FF(Ma*).*

Go through each of the steps till step 12, where a'11, b'11,c'11, d'11 = 0. We need to find some difference and remove it so that a'16, b'16, c'16, d'16 = 0 as well.
*This can only be applied to the **FF()** Round.*

We choose x'12, which gets us A'12 = 2^31.
This leads to A'12 = 2^31.
Using the above, run for $CLS_{22}(2^9) = 2^31$ permutations
We use 2^9 as the probability is half for that.

Now we want the possible values for
w' = F(X,Y,Z) – F(X,Y,Z) where w' must equal 0. This takes us to the third step of z'13 = 2^31 permutations and the probabality is 1/2 for this as well. This cycle is repeated for w'14 as well, where z'14 should lead to 0 with probability 1, and the same happens for w'15 as it did in w'14.*

We need to find x'16, such that A'16 = 0 (and it should be trivially with probability of 1).

This works for all four rounds individually, but never been successful as a group.

Attack using collisions

The MD5 is not collision resistant. That was a weakness seen by Bert den Boer and Antoon Bosselaers (two analysts who attacked the MD4 successfully). The MD5 had the following changes from MD4:

1. added a fourth round
2. a unique additive constant was added
3. function G was changed (Originally ((X OR Y) AND (X AND Z) OR (Y AND Z))
4. the previous result is added
5. the order in which input words are accessed in rounds 2 and 3 is changed
6. usage left circular shift has been changed from before

The second step is the only one that has serious implications as it allows creating collisions for the compression function of MD5.

As it stands, the pseudo-collision arises from initializing the four-word buffer at the start of MD5 to two different values. These values differ only in the MSB of each of the four words. The same message is used for both sets of buffer values and the same message digest is obtained.

A far more serious flaw would be if it were possible to choose one initial starting value for the buffer, not necessarily the one given in the algorithm, and then choose two different messages, perhaps differing in only a few bits of one word, so that the same message digest is obtained. – [5]

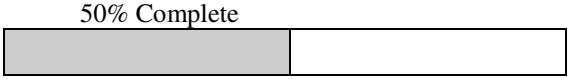
Steps

Initially, the idea was to convey the algorithm in a simplified form as it was done for the crypta-analysis, but going through the journal [6], showed the basic form of attacking the hash which seemed a lot more feasible than the previous method. Unfortunately, I discovered this source too late to implement it at all, as it was a lot more doable than the previous method, though running the code would take a while. The source is straightforward and I could not re-create the steps without directly copying the instructions. According to some sources the it would take over a hundred days for MD5 to be violated via collisions, which would explain why they are building a super computer to just attack the MD5. The cost of this computer is estimated to be $10 million [7].

Conclusion

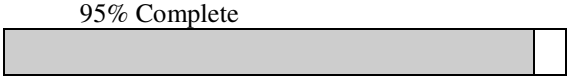
Initial Aim

50% Complete



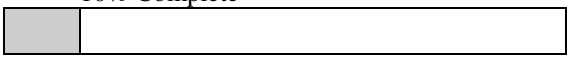
Understanding of the project

95% Complete



Implementation of the project

10% Complete



Unfortunately, I did not achieve my primary goal, which was to attack a portion of the MD5 successfully. I did however end up understanding how to attack it via cryptoanalysis and learned of a collision attack on it. Both seem rather appealing and would do that in my free time. The MD5 is useful in a limited fashion. It will not be long when the MD5 is broken down, as processors get more and more powerful. But as the MD5 becomes more vulnerable, more powerful hashes will come out. The speed for the hashes would be trivially slower, as processors would be fast for them too.

Bibliography

1: http://theory.lcs.mit.edu/~rivest/

2: Bruce Schneir
"Applied Cryptography: Protocols, Algorithms, and Source Code in C"; 18.5 MD5 (pp. 436-441)

3: Thomas A. Berson
"Eurocrypt '92"; Differential Cryptanalysis Mod 2^32 with Applications to MD5, (pp. 71-80)

4: M. Leech
"Chinese Lottery Cryptoanalysis Revisited", Sept. 2003.

5: Hans Dobbertin
"Cryptoanalysis of MD5 Compress", May 2, 1996

6: Bert den Boer, Antoon Bosselaers
"Eurocrypt '93"; Collisions for the compression function of MD5, (pp. 293-305)

7: http://www.rsasecurity.com/

8: http://pajhome.org.uk/crypt/md5/