Analisis Pembuktian Kolisi pada fungsi hash MD5

Muhammad Iqbal faruqi/13507101
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
saidhawwa@itb.ac.id , said_hawwa@yahoo.com, ibnu.musa.khawarizmi@gmail.com

***Abstract*— Dalam era digital seperti saat ini banyak sekali timbul permasalahan-permasalahan baru terkait dengan data-data yang beredar, salahsatunya mengenai integritas data, untuk mengetahui tingkat integritas sebuah data, maka digunkaknlah fungsi hash, MD5 adalah salah satu fungsi hash yang cukup banyak digunakan, pada makalah kali ini penulis akan memaparkan apa itu fungsi hash, algoritma MD5. Lalu akan dibahas pula kolisi pada MD5, mengapa itu bisa terjadi, dan dapatkah itu direkayasa, sehingga kita bisa mengetahui tngkat keamanan algoritma MD5 saat ini dengan melihat kasus-kasus dan scenario yang dapat direkayasa, dan terakhir, penuli akan menyimpulkan apakah algoritma MD5 masih layak digunakan sebagai sebuah metode untuk menguji integritas data atau tidak.**

***Index Terms*—Fungsi Hash, integritas data, kolisi, MD5, rekayasa kolisi, rekayasa scenario.**

I. PENDAHULUAN

Dalam era digital seperti saat ini banyak sekali timbul permasalahan-permasalahan baru terkait dengan data-data yang beredar, salah satu permasalahan yang paling krusial adalah bagaimana kita dapat mengetahui bahwa data atau informasi yang kita dapat dari suatu sumber adalah data yang benar-benar konsisten atau benar. Oleh karena itu muncullah banyak metode untuk mengetahui ke-konsistenan suatu data dan mengetahui apakah benar data yang dikirim tersebut tidak mengalami perubahan dari data aslinya.

Salah satu metode yang ditemukan oleh para ahli informasi untuk menangani masalah konsistensi atau keaslian data adalah menggunakan fungsi hash satu arah. Dengan menggunakan fungsi hash satu arah kita dapat mengetahui apakah data yang kita terima dari sumber adalah data yang benar dan tidak mengalami perubahan.

MD5 adalah salah satu gungsi hash satu arah, yang biasa digunakan untuk menguji keaslian sebuah data atau informasi. Dengan menggunakan MD5 kita bisa mengetahui apakah sebuah data mengalami perubahan dengan melihat fungsi hashnya. Meskipun data hanya mengalami perubahan sebanyak satu bit, namun nilai hash-nya akan berubah secara drastis.

Pada saat ini MD5 menjadi algoritma hash satu arah yang cukup popular pada saat ini, namun pada kenyataannya, ada seseorang yang meng-klaim bahwa algoritma ini sudah tidak aman lagi untuk dipakai, *Xiaoyun Wang, Dengguo Feng, Xuejia Lai dan Hongbo Y* menyatakan mereka telah menemukan kolisi pada MD5 pada tahun 2005. Karena itu penulis ingin melakukan sebuah analisis dan eksplorasi mengenai MD5 untuk mengkaji hal-hal berikut:

1. Pembuktian klaim terjadinya kolisi pada MD5.
2. Mengapa kolisis bisa terjadi?
3. Dapatkah kita merekayasa sebuah kolisi pada MD5?
4. Sejauh mana tingkat keamanan algoritma MD5 saat ini?
5. Terakhir, Apakah MD5 masih layak untuk digunakan sebagai metode pengujian keaslian data atau tidak?

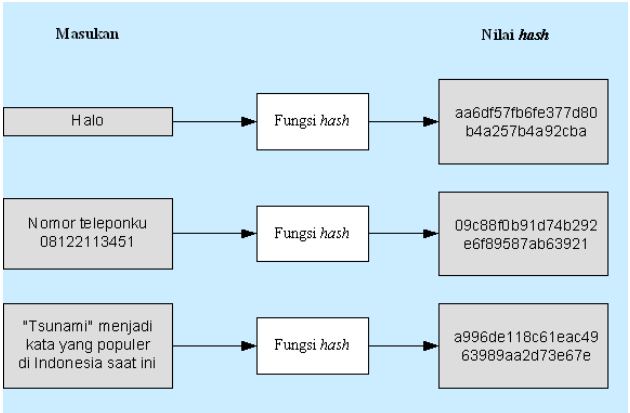
Untuk mengkaji hal-hal yang telah disebutkan di atas penulis akan membahas beberapa teori dasar dan beberapa terminology atau istilah yang akan dibahas pada bab berikutnya.

II. TEORI DASAR

Pada bab ini akan dibahas mengenai Teori Dasar tentang fungsi hash, MD5, dan menjelaskan beberapa terminology yang akan digunakan. Bagian teori dasar ini adalah kumpulan dari berbagai sumber Termasuk beberapa artikel dan slide kuliah.

A. Fungsi Hash

. fungsi hash adalah sebuah fungsi yang menerima masukan sebuah string, yang memiliki panjang sembarang atau bebas, lalu mentransformasikannya menjadi sebuah string yang memiliki panjang yang pasti. Untuk lebih mudahnya, mari kita lihat gambar di bawah ini:



Gambar1: ilustrasi fungsi hash, sumber:

http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Fungsi%20Hash.ppt

Secara matematis fungsi hash dapat dituliskan sebagai berikut:

$$md = H(M)$$
$$where:$$
$$md = MessageDigest(HashValue)$$
$$M = PlainText(Message)$$

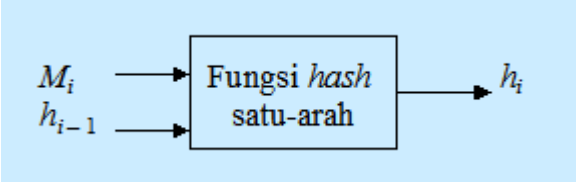
Hash punya beberapa nama lain, yaitu: fungsi kompresi, *fingerprint*, *cryptographic checksum*(MIC), dan *manipulation detection code*(MDC). Pada umumnya hash yang dipakai untuk keperluan konsistensi dan otentikasi adalah fungsi hash satu arah, disebut fungsi hash satu arah karena pesan yang diubah menjadi message digest tidak dapat dikembalikan lagi menjadi pesan semula(*irreversible*).

Ada beberapa sifat fungsi hash yang harus kita perhatikan:

1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nilai (h) dengan panjang tetap (fixed-length output).
3. H(x) mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga H(x) = h. Itulah sebabnya fungsi H dikatakan fungsi hash satu-arah (one-way hash function).
5. Untuk setiap x yang diberikan, tidak mungkin mencari y != x sedemikian sehingga H(y) = H(x).
6. Tidak mungkin mencari pasangan x dan y sedemikian sehingga H(x) = H(y).

Masukan dari fungsi hash adalah blok pesan (M) dan keluaran dari hashing blok pesan sebelumnya untuk lebih jelasnya mari kita lihat gambar berikut ini:

Gambar2: ilustrasi fungsi hash satu arah, sumber: http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Fungsi%20Hash.ppt



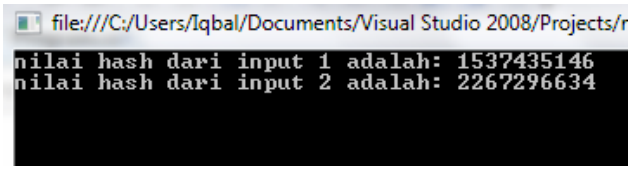
Seperti yang telah dijelaskan sebelumnya fungsi hash satu arah ini berfungsi untuk menguji integritas data. Fungsi hash sangat peka terhadap perubahan pesa, meskipun hanya 1-bit. artinya jika pesan berubah 1-bit, nilai hash akan berubah secara signifikan. Skema penggunaan fungsi hash untuk menguji integritas sebuah data adalah sebagai berikut:

1. Hitung fungsi hash dari data asli, (h1)
2. Hitung fungsi hash dari data yang akan di uji integritasnya (h2)
3. Lalu bandingkan h1 dengan h2
4. Jika h1 = h2, maka data dijamin integritasnya
5. Jika h1 != h2 data telah berubah.

Sebagai contoh, lihatlah dua buah string berikut ini:

```
string input1 = "Set View Distance as low as possible.";
string input2 = "Set View Distance as slow as possible.";
```

String input1 dan string input2 hanya memiliki perbedaan sebuah tambahan karakter, namun nilai hashnya sangatlah berbeda, nila hash dari kedua string itu adalah:

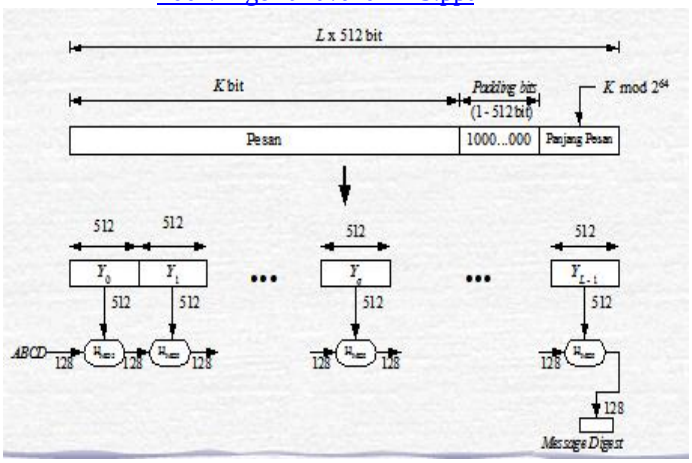


Selain menjadi sebuah metode untuk mengidentifikasi keaslian file fungsi hash juga dapat digunakan untuk menghemat waktu pengiriman yang bertujuan hanya untuk memverifikasi dokumen atau sebagainya, dan juga biasanya fungsi hash dilakukan untuk menyeragamkan panjang password yang beraneka ragam.

B. Hash MD5

MD5 merupakan singkatan dari Message Digest 5, MD5 merupakan algoritma hash satu arah yang didesain oleh Ronald Rivest pada tahun 1991 untuk menyempurnakan fungsi hash sebelumnya yaitu MD4. Message Digest yang dihasilkan oleh MD5 mempunyai panjang 128 bit. Algoritma MD5 adalah sebagai berikut:

Gambar3: ilustrasi Proses MD5, sumber: http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Algoritma%20MD5.ppt



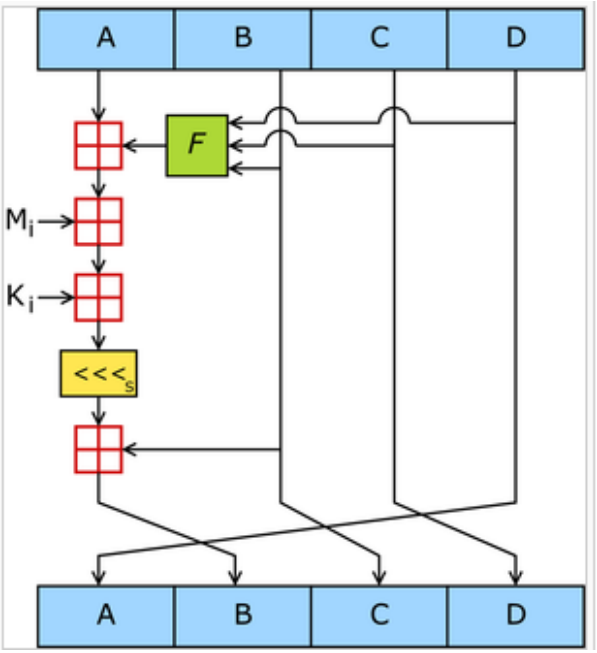
1. **Penambahan bit-bit pengganjal**: Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam

satuan bit) kongruen dengan 448 modulo 512. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

2. **Penambahan Nilai Panjang Pesan**: Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan > 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.
3. **Inisialisasi Penyangga** : *MD5* membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 × 32 = 128 bit. Keempat penyangga ini menampung hasil antara dan hasil akhir. Keempat penyangga ini diberi nama *A*, *B*, *C*, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut: *A* = 01234567, *B* = 89ABCDEF, *C* = FEDCBA98, dan *D* = 76543210
4. **Pengolahan Pesan dalam Blok berukuran 512**: Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}). Setiap blok 512-bit diproses bersama dengan penyangga *MD* menjadi keluaran 128-bit, dan ini disebut proses H_{MD5}.

Proses dalam MD5 secara garis besar dapat dilihat pada gambar berikut:

Gambar4: ilustrasi operasi hash MD5, sumber: http://id.wikipedia.org/wiki/MD5



s menunjukkan perputaran bit kiri oleh s; s bervariasi untuk tiap-tiap operasi. menunjukan tambahan modulo 232. MD5 memproses variasi panjang pesan kedalam keluaran 128-bit dengan panjang yang tetap. Pesan masukan dipecah menjadi dua gumpalan blok 512-bit; Pesan ditata sehingga panjang pesan dapat dibagi 512. Penataan bekerja sebagai berikut: bit tunggal pertama, 1, diletakkan pada akhir pedan. Proses ini diikuti dengan serangkaian nol (0) yang diperlukan agar panjang pesan lebih dari 64-bit dan kurang dari kelipatan 512. Bit-bit sisa diisi dengan 64-bit integer untuk menunjukkan panjang pesan yang asli. Sebuah pesan selalu ditata setidaknya dengan 1-bit tunggal, seperti jika panjang pesan adalah kelipatan 512 dikurangi 64-bit untuk informasi panjang (panjang mod(512) = 448), sebuah blok baru dari 512-bit ditambahkan dengan 1-bit diikuti dengan 447 bit-bit nol (0) diikuti dengan panjang 64-bit.

Algortima MD5 yang utama beroperasi pada kondisi 128-bit, dibagi menjadi empat word 32-bit, menunjukkan A, B, C dan D. Operasi tersebut di inisialisasi dijaga untuk tetap konstan. Algoritma utama kemudian beroperasi pada masing-masing blok pesan 512-bit, masing-masing blok melakukan pengubahan terhadap kondisi.Pemrosesan blok pesan terdiri atas empat tahap, batasan putaran; tiap putasan membuat 16 operasi serupa berdasar pada fungsi non-linear F, tambahan modular, dan rotasi ke kiri. Gambar satu mengilustrasikan satu operasi dalam putaran. Ada empat macam kemungkinan fungsi F, berbeda dari yang digunakan pada tiap-tiap putaran:

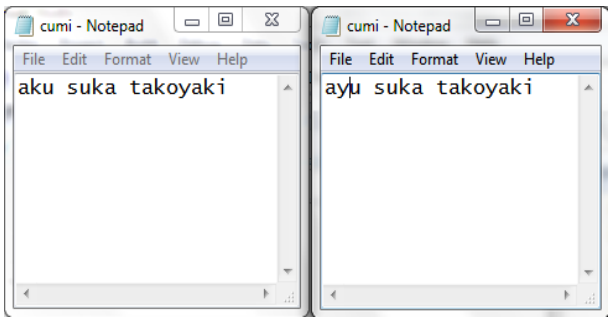
$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Untuk lebih jelasnya, berikut ini adalah pseudo code dari algoritma MD5:

```
  //Catatan: Seluruh variable pada unsigned integer 32-
bit dan dan
  //wrap modulo 2^32 saat melakukan perhitungan

  //Mendefinisikan r sebagai berikut
  var int[64] r, k
  r[ 0..15] := {7, 12, 17, 22,  7, 12, 17, 22,  7, 12,
17, 22,  7, 12, 17, 22}
  r[16..31] := {5,  9, 14, 20,  5,  9, 14, 20,  5,  9,
14, 20,  5,  9, 14, 20}
  r[32..47] := {4, 11, 16, 23,  4, 11, 16, 23,  4, 11,
16, 23,  4, 11, 16, 23}
  r[48..63] := {6, 10, 15, 21,  6, 10, 15, 21,  6, 10,
15, 21,  6, 10, 15, 21}

  //Menggunakan bagian fraksional biner dari integral
sinus sebagai konstanta:
  for i from 0 to 63
      k[i] := floor(abs(sin(i + 1)) × 2^32)

  //Inisialisasi variabel:
  var int h0 := 0x67452301
  var int h1 := 0xEFCDAB89
  var int h2 := 0x98BADCFE
  var int h3 := 0x10325476
```

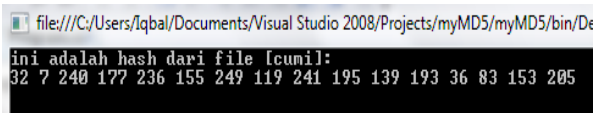
```
   //Pemrosesan awal:
   append "1" bit to message
   append "0" bits until message length in bits = 448
(mod 512)
   append bit length of message as 64-bit little-endian
integer to message

   //Pengolahan pesan paada kondisi gumpalan 512-bit:
   for each 512-bit chunk of message
       break chunk into sixteen 32-bit little-endian
words w(i), 0 = i = 15

       //Inisialisasi nilai hash pada gumpalan ini:
       var int a := h0
       var int b := h1
       var int c := h2
       var int d := h3

       //Kalang utama:
       for i from 0 to 63
           if 0 = i = 15 then
               f := (b and c) or ((not b) and d)
               g := i
           else if 16 = i = 31
               f := (d and b) or ((not d) and c)
               g := (5×i + 1) mod 16
           else if 32 = i = 47
               f := b xor c xor d
               g := (3×i + 5) mod 16
           else if 48 = i = 63
               f := c xor (b or (not d))
               g := (7×i) mod 16

           temp := d
           d := c
           c := b
           b := ((a + f + k(i) + w(g)) leftrotate r(i))
+ b
           a := temp

       //Tambahkan hash dari gumpalan sebagai hasil:
       h0 := h0 + a
       h1 := h1 + b
       h2 := h2 + c
       h3 := h3 + d

var int digest := h0 append h1 append h2 append h3
//(diwujudkan dalam little-endian)
```

Seperti fungsi hash yang lainnya MD5 juga sensitive atau peka terhadap perubahan satu bit. contoh, Berikut ini adalah file “cumi” sebelum dan setelah diubah:

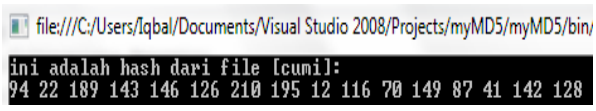


Dan berikut ini hash sebelum dan sesudah:

Hash sebelum:



Hash sesudah:



III. PEMBAHASAN

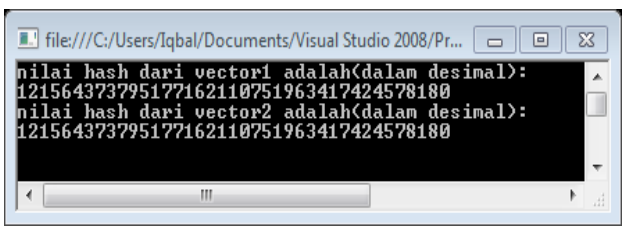
A. *Pembuktian Terjadinya Kolisi Pada MD5*

Sebuah paper yang disusun oleh Xiaoyun Wang, Dengguo Feng, Xuejia, dan Hongbo yang diposting pada 17 agustus 2004, menyatakan ada dua buah vector yang berbeda tetapi menghasilkan nilai MD5 yang serupa, kedua vector itu adalah:

Tabel1: dua buah vector yang berkolisi, sumber: http://www.winhex.com/md5collision.html

Vektor 1					Vektor 2				
d1	31	dd	02	c5	d1	31	dd	02	c5
e6	ee	c4	69	3d	e6	ee	c4	69	3d
9a	06	98	af	f9	9a	06	98	af	f9
5c					5c				
2f	ca	b5	87	12	2f	ca	b5	07	12
46	7e	ab	40	04	46	7e	ab	40	04
58	3e	b8	fb	7f	58	3e	b8	fb	7f
89					89				
55	ad	34	06	09	55	ad	34	06	09
f4	b3	02	83	e4	f4	b3	02	83	e4
88	83	25	71	41	88	83	25	f1	41
5a					5a				
08	51	25	e8	f7	08	51	25	e8	f7
cd	c9	9f	d9	1d	cd	c9	9f	d9	1d
bd	f2	80	37	3c	bd	72	80	37	3c
5b					5b				
d8	82	3e	31	56	d8	82	3e	31	56
34	8f	5b	ae	6d	34	8f	5b	ae	6d
ac	d4	36	c9	19	ac	d4	36	c9	19
c6					c6				
dd	53	e2	b4	87	dd	53	e2	34	87
da	03	fd	02	39	da	03	fd	02	39
63	06	d2	48	cd	63	06	d2	48	cd
a0					a0				
e9	9f	33	42	0f	e9	9f	33	42	0f
57	7e	e8	ce	54	57	7e	e8	ce	54
b6	70	80	a8	0d	b6	70	80	28	0d
1e					1e				
c6	98	21	bc	b6	c6	98	21	bc	b6
a8	83	93	96	f9	a8	83	93	96	f9
65	2b	6f	f7	2a	65	ab	6f	f7	2a
70					70				

Tulisan yang dicatak kuning di atas adalah bit-bit yang mengalami perbedaan, bila kita hitung nilai MD5-nya maka akan menghasilkan MD5 seperti berikut:



Dengan ditemukannya dua buah vector yang memiliki

nilai hash yang sama maka dapat disimpulkan bahwa fungsi hash MD5 telah mengalami kolisi.

B. Mengapa Kolisi Bisa Terjadi?

Secara kemungkinan, memang selalu memungkinkan untuk terjadi kolisi pada sebuah fungsi hash, Karena string ayng dapat dibuat jumlahnya bisa sangat banyak dan tidak terbatas, sedangkan hash value hanya memiliki kemungkinan yang terbatas. Contoh, panjang string hash MD5 adalah 128 bit artinya, semua kemungkinan hash value yang dapat dihasilkan sejumlah $2^{(128)}$, dan tidak mustahil bagi kita untuk membuat $2^{(128)}$ string acak yang berbeda, jika kita memiliki lebih dari $2^{(128)}$ string yang berbeda maka salah satu atau lebih string pasti akan mengalami kolisi. Namun pertanyaannya adalah apakah ada sebuah cara yang cukup efektif untuk mencari kolisi pada MD5 dari serangan brute force attack?

Jawabanya adalah dengan metode difrensial, dengan membaca paper yang didudun oleh Xiaoyun Wang dan Hongbo Yu kita bisa meihat beberapa persamaan diferensial yang digunakan untuk mencari kolisi pada MD5 dengan menggunakan persamaan sebagai berikut:

$$M' = M + \Delta C_1, \Delta C_1 = (0,0,0,0,2^{31},...,2^{15},...,2^{31},0)$$

$$N_i' = N_i + \Delta C_2, \Delta C_2 = (0,0,0,0,2^{31},...,-2^{15},...,2^{31},0)$$

Yang memenuhi persamaan di bawah ini:

$$MD5(M, N_i) = MD5(M', N_i').$$

Mereka meng-klaim bahwa mereka dapat men-generate nilai M, dan M', hanya dengan waktu satu jam, dan setelah ditemukan M dan M', mereka hanya memerlukan waktu l5 menit untuk menemukan N dan N'

C. Dapatkah Kita Merekayasa Sebuah Kolisi pada MD5?

Dengan algoritma yang telah disebutkan diatas, kita bisa men-generate M dan M', lalu N dan N' sehingga MD5(M,N) = MD5(M',N') selain itu, ada kebocoran pada fungsi MD5 ini yaitu bahwa:

$$if\ md5(x) = md5(y) \rightarrow md5(x+p) = md5(y+p)$$

Dari persamaan diatas jelas sekali bahwa kita dapat dengan mudahnya membuat dua buah vector atau file yang sama nilai MD5-nya. Dengan adanya kelemahan fungsi MD5 di atas maka sebuah kolisi pada MD5 dapat di rekayasa, sebagai contoh kita akan gunakan dua buah vector, vektor1 dan vektor2 pada Tabel1, lalu kita tambahkan vector payload sperti dibawah ini:

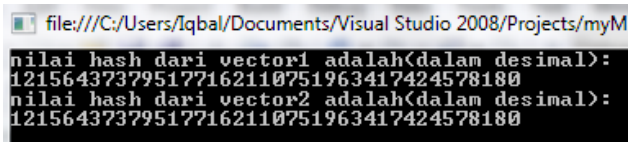
```
byte[] payload = { 0x2a, 0x3a, 0x2b, 0x4c, 0x07, 0x3a, 0x2b, 0x2c };
```

Sehingga vektor1 dan vector 2 akan menjadi seperti pada table berikut:

Tabel2: dua buah vector yang berkolisi yang telah ditambahkan vector payload

Vektor 1	Vektor 2
d1 31 dd 02 c5	d1 31 dd 02 c5
e6 ee c4 69 3d	e6 ee c4 69 3d
9a 06 98 af f9	9a 06 98 af f9
5c	5c
2f ca b5 87 12	2f ca b5 07 12
46 7e ab 40 04	46 7e ab 40 04
58 3e b8 fb 7f	58 3e b8 fb 7f
89	89
55 ad 34 06 09	55 ad 34 06 09
f4 b3 02 83 e4	f4 b3 02 83 e4
88 83 25 71 41	88 83 25 f1 41
5a	5a
08 51 25 e8 f7	08 51 25 e8 f7
cd c9 9f d9 1d	cd c9 9f d9 1d
bd f2 80 37 3c	bd 72 80 37 3c
5b	5b
d8 82 3e 31 56	d8 82 3e 31 56
34 8f 5b ae 6d	34 8f 5b ae 6d
ac d4 36 c9 19	ac d4 36 c9 19
c6	c6
dd 53 e2 b4 87	dd 53 e2 34 87
da 03 fd 02 39	da 03 fd 02 39
63 06 d2 48 cd	63 06 d2 48 cd
a0	a0
e9 9f 33 42 0f	e9 9f 33 42 0f
57 7e e8 ce 54	57 7e e8 ce 54
b6 70 80 a8 0d	b6 70 80 28 0d
1e	1e
c6 98 21 bc b6	c6 98 21 bc b6
a8 83 93 96 f9	a8 83 93 96 f9
65 2b 6f f7 2a	65 ab 6f f7 2a
70	70
/*Payload*/	/*Payload*/
2a 3a 2b 4c 07 3a	2a 3a 2b 4c 07 3a
2b 2c	2b 2c

Dengan input vector di atas, maka nilai hash yang dihasilkan adalah seperti berikut:



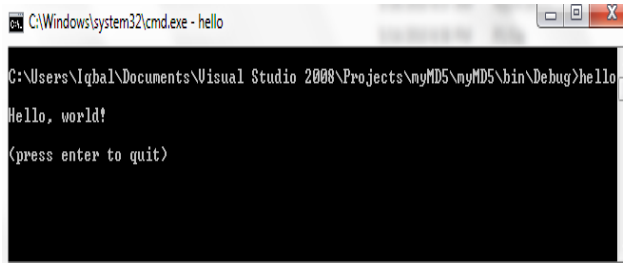
Dengan dihasilkannya dua buah nilai hash yang sama dengan hanya menambahkan sebuah payload maka kita dapat memahami bahwa kolisi pada MD5 adalah sesuatu yang bisa direkayasa.

D. Sejauhmana Tingkat Keamanan Algoritma MD5 saat ini?

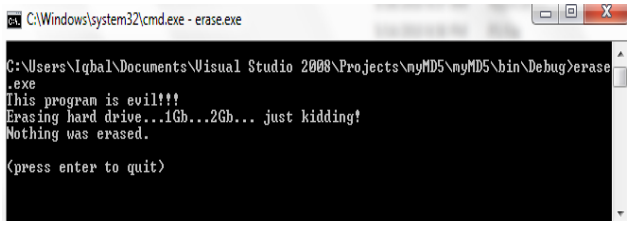
Dari bagian makalah sebelumnya, kita dapat mengetahui bahwa kolisi pada MD5 ini bisa direkayasa, namun pertanyaannya, apakah terdapat sebuah scenario untuk menggunakan rekayasa kolisi tersebut untuk melakukan sebuah intrusi pada data?

Untuk mengetahui jawabannya mari kita sebuah contoh scenario di berikut. Ada dua buah aplikasi console, yang satu bernama erase.exe, dan yang satu bernama hello.exe. hello.exe merepresentasikan “good” file, sedangkan erase.exe merepresentasikan “evil” file, mari kita lihat isi hello.exe dan erase.exe:

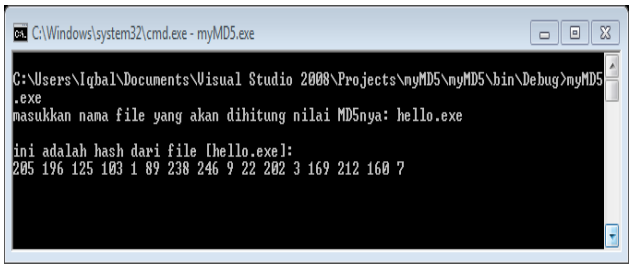
hello.exe:

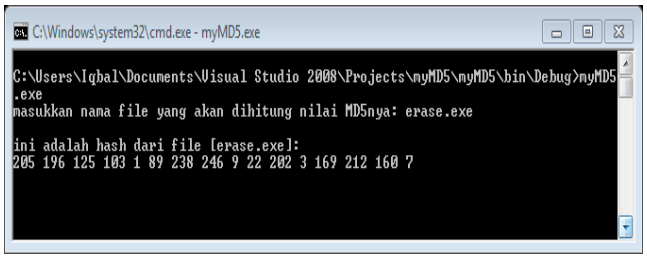


erase.exe:



kita lihat perbedaan yang cukup signifikan pada jalannya kedua aplikasi console tersebut, namun mari kita hitung nilai MD5nya:

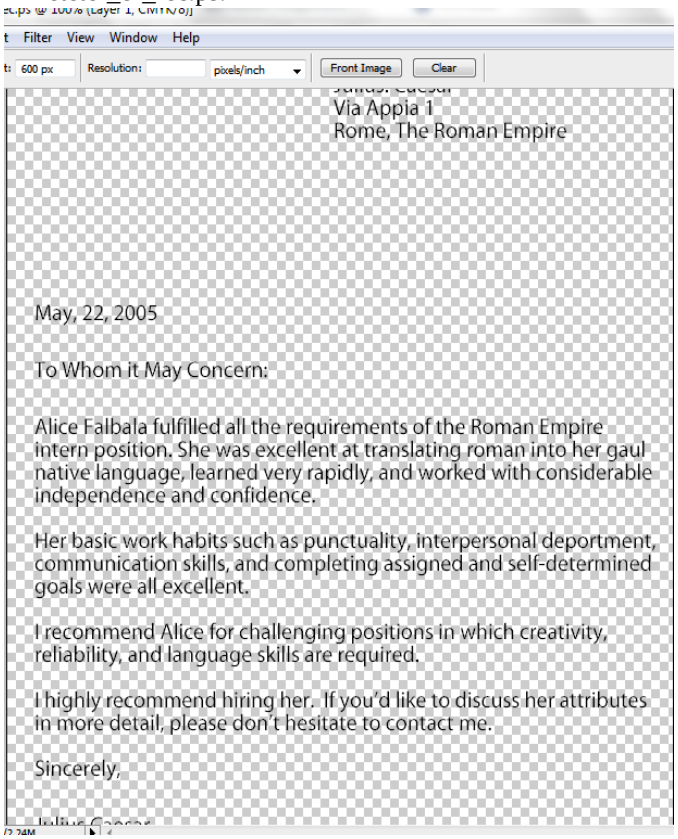




nah, ternyata kedua buah aplikasi di atas mempunyai nilai hash yang sama!!

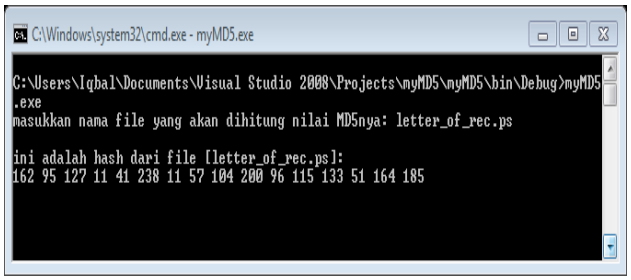
Untuk lebih jelasnya lagi mari kita dua buah file yang memiliki extensi .ps berikut ini, yang pertama adalah letter_of_rec.ps dan file yang lain bernama order.ps, lalu kita buka kedua file tersebut dengan photoshop:

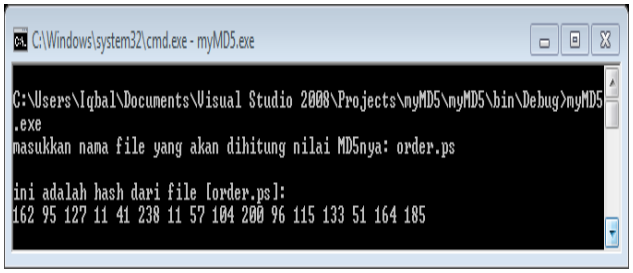
Leteter_of_rec.ps:



Order.ps:

Sekali lagi kita melihat perbedaan yang cukup jelas pada kedua file di atas, lalu kita hitung nilai hashnya:





Kita lihat lagi bahwa kedua file tersebut memiliki nilai MD5 yang sama juga!! Berarti saat ini tingkat keamanan MD5 untuk menguji integritas sebuah file harus kita pertanyakan kembali.

Bayangkan jika kita seorang cracker, kita akan mencoba untuk menyimpan file di sebuah web untuk diunduh oleh orang lain, pada awalnya kita menyimpan sebuah "good" file. Lalu dengan jangka waktu tertentu kita akan mengganti good file dengan "bad" file. Ketika user lain mengunduh file tersebut dan menyamakan nilai MD5nya dengan user lain yang sudah mengunduh "good" file, maka user tersebut tidak akan menyangka bahwa, apa yang dia unduh adalah bad file, dengan begitu sang cracker atau kriptanalis dapat berbuat kerusakan yang cukup fatal di dunia maya.

IV. Conclusion

Ada beberapa hal yang dapat disimpulkan dari pembahasa yang telah penuis sampaikan sebelumnya yaitu:

1. **Mengapa kolisis bisa terjadi?**
   Jawab: karena panjang fungsi hash yang tetap tidak bisa memenuhi seluruh kemungkinan string yang dapat dibuat, karena jumlah string yang dapat dibuat berjumlah tak terbatas
2. **Dapatkah kita merekayasa sebuah kolisi pada MD5?**
   Jawab: Ya, kita dapat merekayasa sebuah kolisi pada MD5 dengan menggunakan metode diferensial, atau dengan mengetahui dua buah vector yang berkolisi
3. **Sejauh mana tingkat keamanan algoritma MD5 saat ini?**
   Jawab: tingkat keamanan MD5 saat ini harus dipertanyakan karena bisa saja terjadi sebuah skenario oleh kriptanalis untuk menipu orang, agar orang tersebut percaya bahwa data yang dia dapat adalah data valid.
4. **Terakhir, Apakah MD5 masih layak untuk digunakan sebagai metode pengujian keaslian data atau tidak?**
   Jawab: tidak, karena berbagai alasan, salah satunya adalah kolisi yang dapat direkayasa. Tetapi kita dapat menggunakan MD5 untuk kasus tertentu

References

Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Y, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD" august 17 2004.

http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Algoritma%20MD5.ppt

http://www.informatika.org/~rinaldi/Kriptografi/2006-2007/Fungsi%20Hash.ppt

http://www.winhex.com/md5collision.html

http://id.wikipedia.org/wiki/MD5

http://www.codeproject.com/KB/security/HackingMd5.aspx

http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

http://www.mscs.dal.ca/~selinger/md5collision/

http://blog.didierstevens.com/2009/01/17/playing-with-authenticode-and-md5-collisions/

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 29 April 2010

ttd

Muhammad Iqbal Faruqi/13507101