Kupas Tuntas Serangan MD5 dengan Collision Attack

Ginanjar Pramadita
NIM 13506014
Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha10, Bandung
2008
e-mail: if16014@students.if.itb.ac.id

ABSTRAK

Salah satu fungsi hash satu arah yang banyak digunakan adalah algoritma MD5 yang ditemukan oleh Ronald Rivest pada tahun 1991 namun telah berhasil di kriptanalisis oleh Arjen Lenstra, Xiaoyun Wang, dan Benne de Weger pada tahun 2005 dan beberapa waktu kemudian, Vlastimil Klima telah berhasil memperbaiki algoritma Arjen Lenstra dkk. sehingga mampu mengkriptanalisis MD5 lebih cepat.

Pada makalah yang akan saya buat ini, saya akan mengupas dengan mendalam mengenai algoritma MD5 dan teknik-teknik kriptanalisis menggunakan algoritma Arjen Lenstra dkk. dan Vlastimil Klima beserta berbagai perbaikan terhadap algoritma-algoritma tersebut agar proses kriptanalisis menjadi lebih mangkus dan sangkil.

Kata Kunci: *MD5, collision attack, kriptografi, kriptanalisis, hash*

1. Pendahuluan

MD5 adalah salah satu fungsi has yang dapat mengkompresi keberagaman panjang *message* menjadi ukuran tertentu yang terdefinisi (misalnua 128 bit). Dalam fungsi hash digunakan XOR, rotasi bit, dan berbagai operasi lain yang ringan sehingga dapat dikakulasi dengan cepat.

Salah satu hal yang sangat penting dalam fungsi hash adalah *collision resistance*. Misalnya p adalah *plaintext* dan h(p) adalah fungsi hash yang menghasilkan *message digest* v, maka *collision resistance* berarti kesulitan (bahkan tidak boleh) untuk menemukan pasaman *message*(p, q) dimana h(p) = h(q).

Pada tahun 2005, Wang et all menemukan *collision attack* yang efisien untuk MD5. Teknik dalam serangan ini menggunakan *differential attack*. Umumnya, *differential attack* menggunakan operasi XOR namun Wang et al menggunakan *modular substraction* [2].

Dalam metoda Wang ini,

2. Deskripsi MD5

MD5 adalah fungsi kompresi yang mengkalkulasikan 128 bit bilangan acak dari suatu *message* dengan berbagai panjang. Pada saat *message* M dimasukan, nilai hash dari M dihitung dengan cara di bawah ini:

1) *Message M* dibagi ke dalam 512-bit *messag*: $M = (M_0, M_1, M_2, \ldots M_n), |M_i| = 512$
2) Misalkan h_i adalah keluaran dari operasi ke-i maka h_I dihitung mungginakan fungsi kompresi MD5 dengan M_{i-1} dan h_{i-1}. Ulangi proses ini dari M_0 hingga M_n.
   Nilai awal h_0 didefinisikan sebagai berikut:
   h_0 = (0x67452301; 0xefcdab89; 0x98badcfe; 0x10325476).
3) Keluaran dari operasi untuk *message* terakhir adalam nilai hash dari *M*

Misalnya *message* M_j dikompresi dengan fungsi kompresi dan operasi ini disebit operasi blok (j + 1).d Setiap blok terdiri dari 64 langkah.

- Langkah ke-1 hingga ke-16 disebut putaran ke-1.

- Langkah ke-17 hingga ke-32 disebut putaran ke-2,
- langkah ke-33 hingga ke-48 disebut putaran ke-3 dan
- Langkah ke-49 hingga langkah ke64 disebut putaran ke-4

Dalam setiap langkah, salah satu rantai variable a, b, c, d dihitung dan urutan variable adalah a1, d1, c1, b1, a1, … . Keluaran dari setiap blok adalah

$$h_i = (a_0 + a_{16}, b_0 + b_{16}, c_0 + c_{16}, d_0 + d_{16}) \quad (1)$$

Variable rantai dihitung dengan cara sebagai berikut. Pertama, M_j dibagi ke dalam 32 bit *message* m_0, … m_{15}. Selanjutnya fungsi ψ didefinisikan sebagai berikut

$$\psi(x,y,z,w,m,s,t) = y + \big((x + \phi(y,z,w) + m + t) mod 2^{32}\big) \lll s \quad (2)$$

Dimana *m* adalah *message input dalam langkah*. *S* adalah banyaknya left siklik sift yang didefiniskan untuk setiap langkah. *t* adalah kostanta yang didefiniskan pada setiap langkah. ϕ adalah fungsi nonlinear yang didefinisikan pada setiap putaran. Rincian fungsi ϕ adalah sebagai berikut:

- 1R : $\phi(y,z,w) = (y \wedge z) \vee (\sim y \wedge w)$
- 2R : $\phi(y,z,w) = (y \wedge w) \vee (z \wedge \sim w)$
- 3R : $\phi(y,z,w) = y \oplus z \oplus$ w
- 4R : $\phi(y,z,w) = z \oplus (y \vee \sim w)$

(3)

Variable berantai a_i, b_i, c_i, d_i didefinisikan sebagai berikut

- $a_i = \psi(a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, m, s, t)$
- $b_i = \psi(b_{i-1}, c_i, d_i, a_i, m, s, t)$
- $c_i = \psi(c_{i-1}, d_i, a_i, b_{i-1}, m, s, t)$
- $d_i = \psi(d_{i-1}, a_i, b_{i-1}, c_{i-1}, m, s, t)$

(4)

3. Modifikasi *Multi-Message*

3.1 Kondisi Tambahan

Untuk memperbaiki kondisi di putaran kedua, diperlukan untuk membuat "kondisi tambahan". Tujuan untuk menset kondisi tambahan adalah untuk menjamin bahwa kondisi putaran kedua dapat diperbaiki.

Kondisi tambahan pada putaran pertama diset bersama dengan kondisi uang memenuhi untuk putaran pertama dengan modifikasi *message* tunggal. Kondisi tambahan pada putaran kedua diset dengan modifikasi *multi-message*. Algoritma pencarian *collision* untuk blok pertama termasuk kondisi tambahan, adalah sebagai berikut

1) Mengenerate *random message*
2) Modifikasi *message* untuk memenuhi kondisi wajib dan pada putaran pertama dengan modifikasi *message* tunggal.
3) Jika kondisi wajib dan kondisi tambahan pada putaran kedua tidak terpenuhi, maka dapat diperbaiki dengan modifikasi *multi-message*.
4) Jika modifikasi *message* tidak memenuhi semua kondisi wajib, pilih m_{14} dan m_{15} lagi dan kembali ke 2).

Algoritma pencarian *collision* untuk blok kedua hampir sama dengan blok pertama.

3.2 Rincian Modifikasi *Multi-Messsage*

Terdapat 14 kondisi pada putaran pertama yang dapat diperbaiki, yaitu: *a*5;4, *a*5;16, *a*5;18, *a*5;32, *d*5;18, *d*5;30, *d*5;32; *c*5;18, *c*5;32, *b*5;32, *a*6;18, *a*6;32, *d*6;32 and *c*6;32.

3.2.1 Perbaikan untuk $a_{5,i}$ (i =4, 16, 18, 32)

a_5 dihitung dengan ekspresi berikut

$$a_5 = b_4 + (a_4 + \phi(b_4, c_4, d_4) + m_1 + t) \lll 5 \quad (5)$$

Bit ke-i pada a_5 dapat diberpaiki dengan modifikasi *message* yang ditunjukan pada tabel 1

Tabel 1: Modifikasi *message* untuk memperbaiki $a_{5,i}$

	shift	Modify m_i
2	12	$m_1 \leftarrow m_1 \pm 2^{i-6}$
3	17	$m_2 \leftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$
4	22	$m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$
5	7	$m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$
6	12	$m_5 \leftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi(a_2, b_1, c_1) - t$

3.2.2 Perbaikan untuk $d_{5,i}$ (i = 18, 30)

d_5 dihitung dengan ekspresi sebagai berikut

$$d_5 = a_5 + (d_4 + \phi(a_4, b_4, c_4) + m_6 + t) \lll 9 \quad (6)$$

Bit ke-i pada d_5 dapat diperbaiki dengan modifikasi message pada Table 2.

Tabel 2: Modifikasi *message* untuk memperbaiki $d_{5,i}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
15	17	$m_{14} \leftarrow m_{14} \pm 2^{i-27}$	$c_4[\pm i - 9], d_4, a_4, b_4$	
16	22	$m_{15} \leftarrow ((b_4 - c_4^{new}) \ggg 22) - b_4 - \phi(c_4^{new}, d_4, a_4) - t$	$b_4, c_4[\pm i - 9], d_4, a_4$	
17	5		$a_5, b_4, c_4[\pm i - 9], d_4$	$d_{4,i-9} = 1$
18	9		$d_5[\pm i], a_5, b_4, c_4[\pm i - 9]$	$a_{5,i-9} \neq b_{4,i-9}$

Tabel 3: Modifikasi *message* untuk memperbaiki $d_{5,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
3	17	$m_2 \leftarrow m_2 + 2^6$	$c_1[23], d_1, a_1, b_0$	$c_{1,23} = 0$
4	22	$b_1 \leftarrow b_1 + 2^{22}$ $m_3 \leftarrow ((b_1^{new} - c_1^{new}) \ggg 22) - b_0 - \phi(c_1^{new}, d_1, a_1) - t$	$b_1[23], c_1[23], d_1, a_1$	
5	7	$m_4 \leftarrow ((a_2 - b_1^{new}) \ggg 7) - a_1 - \phi(b_1^{new}, c_1^{new}, d_1) - t$	$a_2, b_1[23], c_1[23], d_1$	
6	12	$m_5 \leftarrow ((d_2 - a_1) \ggg 12) - d_1 - \phi(a_2, b_1^{new}, c_1^{new}) - t$	$d_2, a_2, b_1[23], c_1[23]$	
7	17	$m_6 \leftarrow m_6 - 2^{22}$	$c_2, d_2, a_2, b_1[23]$	$d_{2,23} = 1$
8	22	$m_7 \leftarrow ((b_2 - c_2) \ggg 22) - b_1 - \phi(c_2, d_2, a_2) - t$	b_2, c_2, d_2, a_2	

Pada Tabel 3, tujuan menset $c_{1,23}$ = 0 adalam untuk membatasi arah perubahan karena kondisi wajib $b_{1,23} = c_{1,23}$ telah ada. Terdapat perbaikan nilai untuk $b_{1,23}$ setelah adanya modifikasi m_2.

3.2.3 Perbaikan Kondisi Lainnya

Perbaikan kondisi-kondisi lainnya ditunjukan pada Tabel 4 hingga 10.

Tabel 4: Modifikasi *message* untuk memperbaiki $c_{5,18}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
10	12	$m_9 \leftarrow m_9 + 2^{23}$	$d_3[4], a_3, b_2, c_2$	$d_{3,4} = 0$
11	17	$m_{10} \leftarrow ((c_3 - d_3^{new}) \ggg 17) - c_2 - \phi(d_3^{new}, a_3, b_2) - t$	$c_3, d_3[4], a_3, b_2$	
12	22	$m_{11} \leftarrow m_{11} - 2^3$	$b_3, c_3, d_3[4], a_3$	$c_{3,4} = 1$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 7) - a_3 - \phi(b_3, c_3, d_3^{new}) - t$	$a_4, b_3, c_3, d_3[4]$	
14	12	$m_{13} \leftarrow ((d_4 - a_4) \ggg 12) - d_3^{new} - \phi(a_4, b_3, c_3) - t$	d_4, a_4, b_3, c_3	

Tabel 5: Modifikasi *message* untuk memperbaiki $c_{5,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
13	7	$m_{12} \leftarrow m_{12} + 2^5$	$a_4[13], b_3, c_3, d_3$	$a_{4,13} = 0$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[13], b_3, c_3$	
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[13], b_3$	
16	22	$b_4 \leftarrow b_4 - 2^{17} + 2^{22}$ $m_{15} \leftarrow ((b_4^{new} - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4[-18, 23], c_4, d_4, a_4[13]$	$b_{4,18} = 1, b_{4,23} = 0$
17	5		$a_5, b_4[-18, 23], c_4, d_4$	$d_{4,23} = 0, (d_{4,18} = 1)$
18	9		$d_5, a_5, b_4[-18, 23], c_4$	$c_{4,18} = 1, c_{4,23} = 1$
19	14		$c_5[-32, 5], d_5, a_5, b_4[-18, 23]$	

Tabel 6: Modifikasi *message* untuk memperbaiki $b_{5,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
13	7	$m_{12} \leftarrow m_{12} + 2^{31}$	$a_4[7], b_3, c_3, d_3$	$a_{4,7} = 0$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[7], b_3, c_3$	
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[7], b_3$	
16	22	$b_4 \leftarrow b_4 - 2^{11}$ $m_{15} \leftarrow ((b_4^{new} - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4[-12], c_4, d_4, a_4[7]$	$b_{4,12} = 1$
17	5		$a_5, b_4[-12], c_4, d_4$	$d_{4,12} = 0$
18	9		$d_5, a_5, b_4[-12], c_4$	$c_{4,12} = 1$
19	14		$c_5, d_5, a_5, b_4[-12]$	$d_{5,12} = a_{5,12}$
20	20		$b_5[-32], c_5, d_5, a_5$	

Tabel 7: Modifikasi *message* untuk memperbaiki $a_{6,18}$

step	shift	Modify m_i	Chaining Variables		Extra Conditions
13	7	$m_{12} \leftarrow m_{12} - 2^{25}$	$a_4[-1], b_3, c_3, d_3$		$a_{4,1} = 1$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[-1], b_3, c_3$		
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[-1], b_3$		
16	22	$m_{15} \leftarrow ((b_4 - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4, c_4, d_4, a_4[-1]$		
17,2	5,12	$m_1 \leftarrow m_1 + 1$	a_5, b_4, c_4, d_4	$d_1[13], a_1, b_0, c_0$	$d_{1,13} = 0$
3	17	$m_2 \leftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$		$c_1, d_1[13], a_1, b_0$	
4	22	$m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$		$b_1, c_1, d_1[13], a_1$	
5	7	$m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$		$a_2, b_1, c_1, d_1[13]$	
6	12	$m_5 \leftarrow m_5 - 2^{12}$		d_2, a_2, b_1, c_1	

Tabel 8: Modifikasi *message* untuk memperbaiki $c_{6,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
4	22	$m_3 \leftarrow m_3 + 2^4$	$b_1[27], c_1, d_1, a_1$	$b_{1,27} = 0$
5	7	$m_4 \leftarrow ((a_2 - b_1^{new}) \ggg 7) - a_1 - \phi(b_1^{new}, c_1, d_1) - t$	$a_2, b_1[27], c_1, d_1$	
6	12	$m_5 \leftarrow m_5 - 2^{26}$	$d_2, a_2, b_1[27], c_1$	$a_{2,27} = 1$
7	17		$c_2, d_2, a_2, b_1[27]$	$(d_{2,27} = a_{2,27})$
8	22	$m_7 \leftarrow m_7 - 2^{26}$	b_2, c_2, d_2, a_2	

Tabel 9: Modifikasi *message* untuk memperbaiki $d_{6,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
9	7	$m_8 \leftarrow m_8 + 2^{15}$	$a_3[23], b_2, c_2, d_2$	$a_{3,23} = 0$
10	12	$m_9 \leftarrow ((d_3 - a_3^{new}) \ggg 12) - d_2 - \phi(a_3^{new}, b_2, c_2) - t$	$d_3, a_3[23], b_2, c_2$	
11	17	$m_{10} \leftarrow m_{10} - 2^{22}$	$c_3, d_3, a_3[23], b_2$	$d_{3,23} = 1$
12	22		$b_3, c_3, d_3, a_3[23]$	$c_{3,23} = 1$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 9) - a_3^{new} - \phi(b_3, c_3, d_3) - t$	a_4, b_3, c_3, d_3	

Tabel 10: Modifikasi *message* untuk memperbaiki $c_{6,32}$

step	shift	Modify m_i	Chaining Variables	Extra Conditions
11	17	$m_{10} \leftarrow m_{10} + 2^{12}$	$c_3[30], d_3, a_3, b_2$	$c_{3,30} = 0$
12	22	$m_{11} \leftarrow m_{11} - 2^7$	$b_3, c_3[30], d_3, a_3$	$d_{3,30} = a_{3,30}$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 7) - a_3 - \phi(b_3, c_3^{new}, d_3) - t$	$a_4, b_3, c_3[30], d_3$	
14	12	$m_{13} \leftarrow ((d_4 - a_4) \ggg 12) - d_3 - \phi(a_4, b_3, c_3^{new}) - t$	$d_4, a_4, b_3, c_3[30]$	
15	17	$m_{14} \leftarrow m_{14} + 2^{22} - 2^{29}$	$c_4[8], d_4, a_4, b_3$	$c_{4,8} = 0$
16	22	$m_{15} \leftarrow m_{15} - 2^7 - 2^{17}$	$b_4, c_4[8], d_4, a_4$	$(a_{4,8} = 0), (d_{4,8} = 1)$
17	5		$a_5, b_4, c_4[8], d_4$	$(d_{4,8} = 1)$
18	9		$d_5, a_5, b_4, c_4[8]$	$a_{5,8} = b_{4,8}$
19	14	($m_{11} \leftarrow m_{11} - 2^7$: modified in step 12)	c_5, d_5, a_5, b_4	

4. Aplikasi Pencarian Collision MD5

Misalnya terdapat dua vektor vec1 dan vec2 yang yang memiliki *message digest* MD5 yang sama.

(sumber: http://www.doxpara.com/md5_someday.pdf)

```
static byte[] vec1 =
{
   0xd1, 0x31, 0xdd, 0x02, 0xc5, 0xe6
   , 0xee , 0xc4 , 0x69 , 0x3d, 0x9a , 0x06
   , 0x98 , 0xaf , 0xf9 , 0x5c , 0x2f , 0xca
   , 0xb5 , /**/0x87 , 0x12 , 0x46 , 0x7e
   , 0xab , 0x40 , 0x04 , 0x58 , 0x3e , 0xb8
   , 0xfb , 0x7f , 0x89 , 0x55 , 0xad
   , 0x34 , 0x06 , 0x09 , 0xf4 , 0xb3 , 0x02
   , 0x83 , 0xe4 , 0x88 , 0x83 , 0x25
   , 0x71 , 0x41 , 0x5a, 0x08 , 0x51 , 0x25
   , 0xe8 , 0xf7 , 0xcd , 0xc9 , 0x9f ,
   0xd9 , 0x1d , 0xbd , 0xf2 , 0x80 , 0x37
   , 0x3c , 0x5b , 0xd8 , 0x82 , 0x3e
   , 0x31 , 0x56 , 0x34 , 0x8f , 0x5b , 0xae
   , 0x6d , 0xac , 0xd4 , 0x36 , 0xc9
   , 0x19 , 0xc6 , 0xdd , 0x53 , 0xe2 , 0xb4
   , 0x87 , 0xda , 0x03 , 0xfd , 0x02
   , 0x39 , 0x63 , 0x06 , 0xd2 , 0x48 , 0xcd
   , 0xa0 , 0xe9 , 0x9f , 0x33 , 0x42
   , 0x0f , 0x57 , 0x7e , 0xe8 , 0xce , 0x54
   , 0xb6 , 0x70 , 0x80 , 0xa8 , 0x0d
   , 0x1e , 0xc6 , 0x98 , 0x21 , 0xbc , 0xb6
   , 0xa8 , 0x83 , 0x93 , 0x96 , 0xf9
   , 0x65 , 0x2b , 0x6f , 0xf7 , 0x2a , 0x70
};


static byte[] vec2 =
{
   0xd1 , 0x31, 0xdd , 0x02 , 0xc5 , 0xe6
   , 0xee , 0xc4 , 0x69 , 0x3d , 0x9a , 0x06
   , 0x98 , 0xaf , 0xf9 , 0x5c, 0x2f , 0xca
   , 0xb5 , /**/0x07 , 0x12 , 0x46 , 0x7e
   , 0xab , 0x40 , 0x04 , 0x58 , 0x3e , 0xb8
   , 0xfb , 0x7f , 0x89 , 0x55 , 0xad
   , 0x34 , 0x06 , 0x09 , 0xf4 , 0xb3 , 0x02
   , 0x83 , 0xe4 , 0x88 , 0x83 , 0x25
   ,/**/ 0xf1 , 0x41 , 0x5a , 0x08 , 0x51 ,
0x25
   , 0xe8 , 0xf7 , 0xcd , 0xc9 , 0x9f
   , 0xd9 , 0x1d , 0xbd , /**/0x72 , 0x80
   , 0x37 , 0x3c , 0x5b, 0xd8 , 0x82
   , 0x3e , 0x31 , 0x56 , 0x34 , 0x8f , 0x5b
   , 0xae , 0x6d , 0xac , 0xd4 , 0x36
   , 0xc9 , 0x19 , 0xc6 , 0xdd , 0x53 , 0xe2
   , /**/0x34 , 0x87 , 0xda , 0x03 , 0xfd
   , 0x02 , 0x39 , 0x63 , 0x06 , 0xd2 , 0x48
   , 0xcd , 0xa0, 0xe9 , 0x9f , 0x33
   , 0x42 , 0x0f , 0x57 , 0x7e , 0xe8 , 0xce
   , 0x54 , 0xb6 , 0x70 , 0x80 , /**/ 0x28
   , 0x0d , 0x1e, 0xc6 , 0x98 , 0x21 , 0xbc
   , 0xb6 , 0xa8 , 0x83 , 0x93 , 0x96
   , 0xf9 , 0x65 , /* flag byte*/0xab
   , 0x6f , 0xf7 , 0x2a , 0x70
};
```

Dengan adanya pasanyan ini, maka sembarang data dengan sembarang ukuran akan memenuhi aturan berikut: MD5(vec1 + data) = MD5(vec2 + data). Data dibangun dengan:

- Panjang message 1
- Panjang message 2
- Isi message 1
- Isi message 2

Program untuk mengenerate adalah sebagai berikut

```
static void Main(string[] args)
{
   if (args.Length != 2)
    Usage();


   byte[] goodFile=ReadBinaryFile(args[0]);
   byte[] evilFile=ReadBinaryFile(args[1]);

   WriteBinary("good.bin", vec1, goodFile,
         evilFile);
   WriteBinary("evil.bin", vec2, goodFile,
         evilFile);

   ShowMD5("good.bin");
   ShowMD5("evil.bin");
}

private static void Usage ()
{
   Console.WriteLine("Usage: md5gen
         good_file evil_file");
   Environment.Exit (-1);
}


private static void WriteBinary (string
         sOutFileName, byte[] prefix,
         byte[] goodFile, byte[] evilFile)
{
   using (FileStream fs =
         File.OpenWrite (sOutFileName))
   {
      using (BinaryWriter writer = new
         BinaryWriter(fs))
      {
         writer.Write(prefix);
         writer.Write ( goodFile.Length );
         writer.Write ( evilFile.Length );
         fs.Write (goodFile, 0,
                  goodFile.Length);
         fs.Write (evilFile, 0,
                  evilFile.Length);
         fs.Close ();
      }
   }
}
```

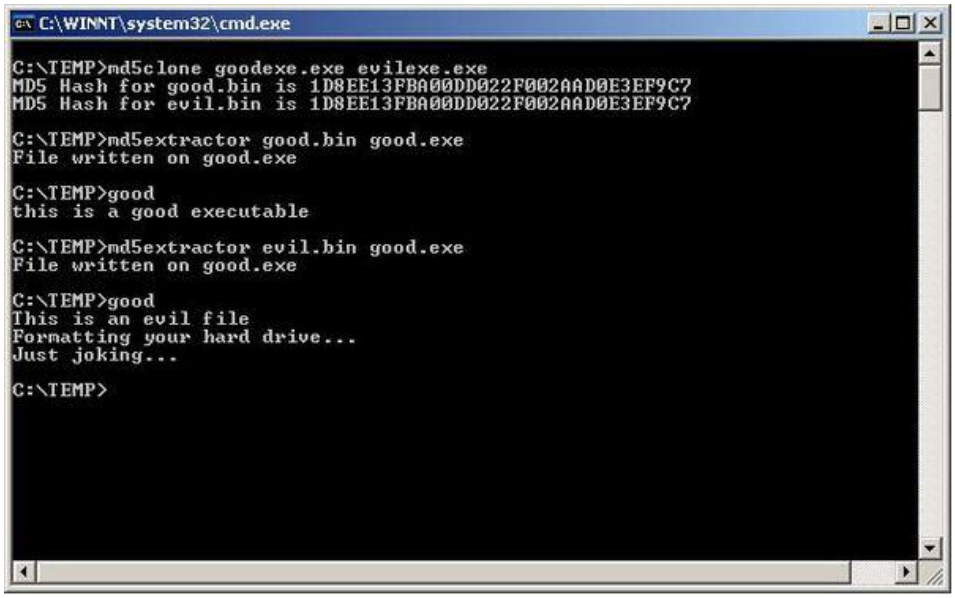
Program mengextraksi

```
[STAThread]
  static void Main(string[] args)
  {
    if (args.Length == 0)
     Usage();
    ExtractFile(args[0], args[1]);
  }


  private static void ExtractFile (
          string sfilename,
          string soutputfile)
  {
    using (BinaryReader reader =
             new BinaryReader(
          File.OpenRead (sfilename)))
    {
      byte[] vec = reader.ReadBytes (128);
      int goodSize = reader.ReadInt32 ();
      int evilSize  = reader.ReadInt32 ();
      /// open evil file

      if (vec[123] == 0xab)
      {
        reader.ReadBytes (goodSize);
        byte[] evil =
          reader.ReadBytes (evilSize);
        using (BinaryWriter writer =
              new BinaryWriter(
          File.OpenWrite (soutputfile)))
        {
           writer.Write (evil);
           writer.Close ();
        }
      }
      else
      {
        byte[] good =
          reader.ReadBytes (goodSize);
        using (BinaryWriter writer =
            new BinaryWriter(
          File.OpenWrite (soutputfile)))
        {
           writer.Write (good);
           writer.Close ();
        }
      }
      Console.WriteLine ("File written on
          {0}", soutputfile);
    }
  }


  private static void Usage ()
  {
     Console.WriteLine(
        "Usage: md5extract file.bin
          output_file.exe");
     Environment.Exit (-1);
  }
```



Gambar 1: Mengenerate Collision

Modus penggunaan (Lihat Gambar 1)

Misal goodFile adalah prorgam yang tidak berbahaya dan telah ditandatangani digital. Selanjutnya, program dengan tanda tangan yang sama dapat dikirim misalnya evilFile.bin yang merupakan *malware* yang akan menimpa program goodFile.bin. Hal ini dapat dilakukan karena kedua file tersebut memiliki hash yang sama.

Pada Gambar 1 goodexe.exe dan evilexe.exe dibuat pasangan collisionnya menggunakan fungsi dan

menghasilkan good.bin dan evil.bin yang memiliki MD5 hash yang sama. Selanjutnya kedua file ini dapat saling menggantikan.

DAFTAR PUSTAKA

[1] Rinaldi Munir. Diktat Kuliah Kriptografi Program Studi Teknik Informatika Institut Teknologi Bandung. 2006

[2] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, Advances in EUROCRYPT2005, LNCS 3494, pp. 1–18, 2005.

[3] R.L. Rivest. The MD5 Message-Digest algorithm. Internet RFC, April 1992. RFC 1321.

[4] V. Klima: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications. 2005

[5] H. Dobbertin: The status of MD5 after a recent attack, CryptoBytes 2 (2), 1996

[6] Jie Liang and Xuejia Lai. Improved Collision Attack on Hash Function MD5. 2005