Studi mengenai Collision pada MD5 dan Modifkasi Program Lama dalam menjawab solusi tersebut

Andzarrahim
13504013
Departemen Teknik Informatika
Institut Teknologi Bandung
E-mail : if14013@students.if.itb.ac.id

Abstraksi

MD5 dikenal sebagai fungsi *hash* dalam kriptografi yang memiliki nilai *hash* 128 bit. MD5 telah banyak digunakan dalam berbagai jenis aplikasi keamanan dan umumnya digunakan untuk memeriksa integritas dari file. Nilai dari Message Diggest / Hash dari fungsi ini biasanya diekspresikan karakter Hexadecimal .

MD5 memiliki penyangga atau Inintial value yang memiliki besar 8 karakter yang merupakan penampiung nilai dari hasil proses yang terjadi pada MD5. Akan tetapi layaknya sebuah algoritma selalu saja ada masalah yang dapat ditemukan. Masalah ini disebut *Collision*, *Collision* memiliki definisi dua buah file input yang berbeda memiliki output / MD yang sama.

Dalam makalah ini penulis akan mencoba menjelaskan kenapa collision bisa terjadi pada MD 5 dengan menggunakan eksperimen yang akan dilakukan oleh penulis sendiri dengan menggunakan source code yang dibuat oleh penulis di tugas sebelumnya. Lalu penulis akan melakukan modifikasi yang bertujuan meminimalisir terjadinya collision pada MD 5

Kata Kunci: *MD5, Collision, Inital Vector, Message Digest(MD)*

1. PENDAHULUAN

Di zaman sekarang seperti ini, trasnfer data atau dokumen dalam bentuk file sudah merupakan kegiaatan yang hampir dilakukan setiap hari. Selain itu, dengan perkembangan teknologi yang semakin pesat, pengiriman data dapat dilakukan kapanpun, dimanapun dan sampai dalam sekejap waktu.
Oleh karena itu , pentingnya keamanan dalam pengantaran datanya perlu diutamakan apalagi jika data tersebut mengandung informasi yang sangat serius. Perlunya pengecekan akan integritas data yang dikirimkan oleh pihak yang dikirimkan merupakan prioritas pertama dalam keamanan berjaringan .
Karena alasan itulah MD5 diciptakan, MD5 berguna dalam memerikasa integritas data yang dikirim oleh pengirim. Itu bertujuan untuk mengecek apakah file yang dikirim file yang sama atau berasal dari pengirim yang benar.

Akan tetapi dibalik kehebatan yang dimiliki oleh MD 5 ternyat a masih ada saja kekurangan yang muncul. Collision yang dapat menciptakan kerancuan dalam pengiriman data. Collision yang tercipta ketika ada dua dokumen yang berbeda akan tetapi memilki MD yang sama.

Pada dokumen ini akan dijelaskan tentang MD 5, cara kerjanya. Selain itu, akan dijelaskan beberapa sejarah singkat tentang terjadinya collision.

2. DASAR TEORI

2.1.Fungsi Hash MD5

Algoritma MD5 disusun oleh Profesor Ronald L. Rivest dari MIT. Berdasarkan RFC 1321, Profesor Ron Rivest memberikan standar baku untuk fungsi hash ini. Fungsi hash ini mengeluarkan output yang memiliki panjang 128 bit, selain itu fungsi hash ini memiliki penyangga yang nilai awalnya ditentukan yang berguna untuk menyimpan nilai awal dari blok bit yang akan diproses.

Pada dasarnya MD5 digunakan untuk mem-*verifikasi* integritas data, selain itu MD5 sendiri merupakan perbaikan dari algoritma MD4 akan tetapi tidak akan dibahas lebih lanjut.

Adapun langkah langkahnya adalah sebagai berikut :

1. Penambahan panjang bit
   Di sini pesan diperpanjang sampai berukuran kelipatan 448 bit modulo 512, maksudnya jika ada pesan yang memiliki panjang melebihi 448 bit, maka perpanjangan pesan akan dilakukan sebesar 512 + 448 bit sehingga sebesar kelipatan 448 modulo 512 bit.

   Penambahan panjang pesan itu sendiri

dilakukan dengan cara menambahkan bit “1” setelah pesan asli lalu menambahkan bit “0” setelah nya sehingga memiliki panjang 448 bit.

2. Penambahan panjang pesan total
   Pada tahap ini, pesan akan ditambahkan dengan panjang pesan itu sendiri . Panjang pesan itu sendiri berukuran 64 bit dan jika panjang pesan memiliki panjang yang berukuran lebih dari 64 bit maka yang diambil adalah bit pertamanya saja (low order). Tujuan dari tahap ini adalah menutupi 64 bit yang kurang pada tahap pertama.

3. Inisialisasi penyangga
   Di tahap ini, akan diinisialisasi 4 penyangga yang nilainy ditentukan dari awal. Setiap penyangganya memiliki panjang 32 bit Adapun nilai yang penyangga biasa miliki adalah

   A = 01 23 45 67
   B = 89 ab cd ef
   C = fe dc ba 98
   D = 76 54 32 10

   Umumnya nilai nilai yang dimiliki penyangga ada seperti ini, akan tetapi, nilai tersebut dapat diubah sesuai dengan keinginan asalkan memiliki panjang bit yang sama yang sesuai dengan rfc 1321

4. Pemroresan pesan dalam blok 512 bit
   Pada tahap ini pesan dibagi menjadi beberapa blok yang memiliki panjang bit yang seragam yaitu 512 bit dan akan diproses dengan penyangga sehingga mengeluarkan MD yang memiliki panjang 128 bit.

   Proses H MD5 memilki 4 buah putaran, dimana masing masing putaran memiliki operasi dasar sebanyak 16 kali dan setiap operasi tersebut menggunakan tabel T yang membantu dalam proses peputaran tersebut.
   Adapun operasi dasar MD5 yang dilakukan memiliki persamaan sebagai berikut :

   A ← b + CLS(a+g(b,c,d)+X[k]+T[i])

   a,b,c,d = 4 penyangga bernilai 32 –bit
   g = salah satu fungsi MD 5 (f,g,h,i)
   CLSx = pergeseran bit ke kiri sebanyak x bit
   X[k] = kelompok 32 bit ke-k dari blok 512 bit ke q
   T[i] = elemen tabel T ke i
   + = operasi penjumlahan modulo

   Adapun perputaran yang terjadi pada penyangga pada satu kali perputaran adalah sebagai berikut :

   Temp ← a
   d ← c
   c ← b
   b ← a
   a ← temp

   Secara umum persamaan diatas dapat dipermudah menjadi

   Output = LSB(A) + LSB(B) + LSB (C) + LSB (D)

   A,B,C,D = penyangga dari MD 5

   Output yang dihasilkan memiliki panjang 128 bit

2.2.Collision

Collision memiliki arti secara harfiah adalah tumbukan. Di sini, collision memiliki pengertian dua file memiliki output yang sama setelah melalui proses MD5.

MD5(A) = MD5(B)

Note : A memiliki pesan yang berbeda dengan B

Adapun beberapa kasus collision yang terkenal berdasarkan penemunya adalah :

1. Hans Dobbertin
   Pada tahun 1996, pada tahun itu dia menemukan collision ketika melakukan proses dengan ketentuan *initial value* untuk kedua input sama, sedangkan pesan yang akan di hash memiliki isi yang berbeda.

   Hash (IV:X) = Hash(IV:X’)

   IV = initial value
   X, X’ = pesan yang akan di hash

   Pada kasus ini, dobbertin mendapatkan kesimpulan jika kita memilih suatu *initial value* , lalu akan digunakan untuk mengolah dua pesan yang berbeda, walaupun perbedaannya hanya beberapa bit ada kemunngkinan MD yang ditemukan akan bernilai sama.

2. Xiaoyun Wang
   Di sini, ia melakukan sedikit perubahan dari nilai *initial value* yang ditentukan dobertin. Dia merubahnya menjadi seperti yang ada pada standar MD5 yaitu :

IV = 0x67452301 0xefcdab89 0x98badcfe 0x10325476

Dalam menemukan collisionnya, ia menggunakan IV dan pesan yang masing masing berbeda (MD5(IV,X) = MD5(IV',X')). Dari sini, dia berhasil menemukan suatu algoritma yang cukup efektif dalam menemukan collision

3. Philip Hawks, Michael Paddon, Gregory G. Rose

   Pada dasarnya penelitian mereka bertujuan untuk menjelaskan algoritma yang dipakai pada proses sebelumnya (Wang).

4. Vlastimil Klima

   Klima pada prosesnya menggunakan dua versi yang berbeda dalam penggunaan *initial value* nya. Dalam proses penelitiannya, Klima menggunakan pendekatan yang berbeda dengan Hawks, di sini klima berhasil menemukan cara untuk menghasilkan MD pada blok pertama yang sama dengan IV dan pesan yang berbeda.

5. Stefan Lucks, Magnus Daum

   Ini kasus terbaru, dan merupakan kasus yang berdasarkah hasil temuan bukan merupakan hasil peneitian . Pada kasus ini ada beberapa langkah yang dapat digunakan untuk menemukan MD yang berbeda dari pesan yang memiliki sedikit perbedaan namun tidak akan dibahas lebih lanjut dalam makalah ini.

3. HASIL DAN PEMBAHASAN

Dengan menggunakan algoritma MD5 yang dibuat penulis sebelumnya. Penulis akan mencoba melakukan eksperimentasi untuk menemukan kasus collision pada dua input yang berbeda.
Selain itu dengan menggunakan teori yang didapat penulis dari kasus di atas, penulis akan berusaha melakukan eksperimentasi yang beracuan dari penelitian di atas.

3.1.Studi Kasus

Untuk menemukan collision, penulis menggunakan input teks yang berbeda yang dapat menghasilkan MD yang sama yang didapat dari internet.

Potongan input text pertama

```
%!PS-Adobe-1.0
%%BoundingBox: 0 0 612 792
(èB¦jÞM_àÕ‰_øåoeÁ§/Ê·_
F~ªÀ_T>±û _Em3__üSà[ï
ƒ_#yRíZ3Î6™
oe nEÚR„yë/½_•õWåv:ì¿ª_
UEµsYH2ô}
,¹£vÔ6_
ýï;ƒ¡ò}ëÊ6_S†XkÈô”ôNa_|„€`Ïï”µ_
)(èB¦jÞM_àÕ‰_øåoeÁ§/Ê·—
F~ªÀ_T>±û _Em3__üSà[ï
ƒ_£xRíZ3Î6™
oe nEZR„yë/½_•õWåv:ì¿ª_
UEµsYE2o}
,¹£vÔ6_
ýï;ƒ¡ò}ëÊ6_S†ØkÈô”ôNa_|„€`Ïo”µ_
)eq{/Times-Roman
findfont 20 scalefont setfont
```

Potongan input text kedua

```
%!PS-Adobe-1.0
%%BoundingBox: 0 0 612 792
(èB¦jÞM_àÕ‰_øåoeÁ§/Ê·—
F~ªÀ_T>±û _Em3__üSà[ï
ƒ_£xRíZ3Î6™
oe nEZR„yë/½_•õWåv:ì¿ª_
UEµsYE2o}
,¹£vÔ6_
ýï;ƒ¡ò}ëÊ6_S†ØkÈô”ôNa_|„€`Ïo”µ_
)(èB¦jÞM_àÕ‰_øåoeÁ§/Ê·—
F~ªÀ_T>±û _Em3__üSà[ï
ƒ_£xRíZ3Î6™
oe nEZR„yë/½_•õWåv:ì¿ª_
UEµsYE2o}
,¹£vÔ6_
ýï;ƒ¡ò}ëÊ6_S†ØkÈô”ôNa_|„€`Ïo”µ_
)eq{/Times-Roman
findfont 20 scalefont setfont
```

di atas dapat dilihat ada sedikit perbedaan yang dapat dilihat dari kedua potongan di atas. Akan tetapi dari kedua input text tersebut didapatkan MD yang sama yaitu

5421a523481fdc6a2a1c832e72c7b8a5

Selain dengan kasu s di atas penulis melakukan beberapa percobaan lain dengan menggunakan acuan input text di atas dan tidak berhasil menemukan *collision* yang dapat digunakan sebagai bahan analisis.

3.2.Analisis

Berdasarkan percobaan yang dilakukan oleh penulis ada beberapa kelemahan yang ditemukan oleh penulis untuk enkripsi MD5. Kelemahan pertama adalah dalam jumlah penyangga yang digunakan. Dalam standar MD5, hanya ada 4 penyangga yang digunakan

sehingga memiliki panjang output 128 bit. Output yang ditampilkan jelas tidak memiliki perbedaan yang signfikan jika ada karakter yang memang memiliki bit yang berdekatan.

Kondisi ini bisa dianalogikan dengan penggunaan beberapa angka di belakang koma.

Misal
A = 8 / 3 = 2 → 2,312

Kelemahan kedua adalah perputaran bit (LSB) yang dilakukan hanya sekali. Untuk beberapa kasus di atas (yang ada didasar teori) perputaran bit merupakan perhatian utama dalam alasan terjadinya collision. Dikarenakan perputaran yang seragam yang dimiliki oleh MD5 tidak begitu efektif dalam meningkatkan kompleksitas output yang akan dikeluarkan.

Kelemahan ketiga untuk MD5 tidak adanya pre-proses untuk karakter sebelum dirubah menjadi data dalam bit. Jika ada pre-proses yang bisa memproses karakter dengan masukan parameter tertentu dan dengan fungsi tertentu tentu akan menambahkan kerumitan dalam penyjian data.

Aku adalah

A → ??? (?,?) → [8 bit data]
k→ ??? (?,?) → [8 bit data selanjutnya]
u → ??? (?,?) → [8 bit data selanjutnya]
.
.
.

??? (?,?) → fungsi pre-proses dengan masukan tertentu dan parameter tertentu

3.3.Modifikasi Program

Berdasarkan analisis di atas dan beberapa literatur yang telah dibaca penulis ada beberapa modifikasi yang penulis tamabahkan ke dalam algoritma yang telah dibuat. Beberapa modifikasi tersebut adalah :

1. Penambahan jumlah penyangga

   Tujuan dari penambahan ini adalah untuk meningkatkan kompleksitas yang dapat dihasilkan oleh fungsi MD5. Penulis menambahkan 4 variabel penampung lagi sehingga output dari MD5 menjadi 256 bit

   A = 01234567
   B = 89abcdef
   C = fedcba98
   D = 76543210
   E = abcd1111
   F = 0000abcd
   G = efab1111
   H = 1111efab

   Dengan adanya penambahan penyangga menjadi dua kali lipat panjangnya meningkatkan kompleksitas output yang ditampilkan. Selain itu juga dapat membantu menjamin integritas data.

2. Perputaran bit yang ditentukan oleh jenis penyangga

   Pada modifikasi ini, penulis merubah aturan dalam banyaknya perputaran bit untuk setiap proses dalam penyangganya . Standar MD5 pertama hanya menggunakan sekali perputaran, maka untuk modifikasi ini penulis menggunakan parameter jenis penyangga.

   Contoh : (hanya menggunakan 8 bit pertama namun kenyataannya MD5 memproses 32 bit secara keseluruhan)

   A → 1 kali → 11101111 → 11011111
   B → 2 kali → 10101010 → 01010101 → 10101010 (kebetulan sama dengan yang pertama)

   ... dan selanjutnya hingga penyangga H memiliki putaran sebanyak 8 kali. Dengan ini, kemungkinan dari munculnya nilai MD yang identik dari karakter yang seragam sekalipun dapat dituntaskan.

3. Penambahan bit (xor) yang ditentukan oleh posisi karakter

   Salah satu jenis pre – proses yang penulis implementasikan. Pada proses Md5 terdahulu karakter input langsung diproyeksikan menjadi bilangan biner sesuai dengan ketentuan ASCII sebesar 8 bit dan langsung dimasukkan ke dalam pesan yang akan diproses.

   Akan tetapi dengan menggunakan pre proses ini bilangan biner yang dihasilkan akan diproses lebih lanjut sehingga akan mengurangi kemungkinan MD yang identik dari karakter yang mirip.

   Misal

Aku adalah …

A → 10101101

Karena A merupakan karakter pertama dalam masukan input dan berada dalam posisi ganjil maka dia hanya ditambahkan (xor)

A → 10101101 xor 11111111 → 01010010

Begitu juga seterusnya

k → 11010111

karena u berada dalam posisi ganjil maka u ditambahkan

u → 10010011 xor 11111111 → 01101100

misalkan pun masukan inputnya seragam semua

AAAAAA

maka setelah melalui ketiga proses tersebut maka akan memiliki nilai MD yang berbeda

3.4.Analisis

Dengan menggunakan ketiga perubahan di atas, maka sudah dipastikan apabila input text sama dengan kasus pertama maka akan mengeluarkan MD yang berbeda.

4. KESIMPULAN

Algoritma MD5 merupakan salah satu algoritma yang dapat meng-hash data menjadi output yang berukuran 128 bit. Cara kerja algoritma ini adalah mengubah pesan menjadi bilangan biner lalu ditambahkan dengan panjang dari pesan itu sendiri. Setelah itu pesan ditambahkan oleh bilangan biner menjadi kelipatan 448 modulo 512. Lalu pesan itu dipecah menjadi kelipatan 512 sebelum diproses menjadi output.

MD5 digunakan untuk menguji integritas data di banyak aplikasi/tugas. Dalam kebutuhan berjaringan MD5 digunakan untuk meningkatkan keamanan terutama dalam hal transfer data. Akan tetapi dibalik keunggulan yang diberikan oleh ternyata MD5 masih ada kekurangan. Dalam pemroresan pesan menjadi outputnya ternyata masih terjadi *collision*.

Dalam makalah ini penulis telah melakukan sedikit perubahan dalam algoritma standar MD5. Algoritma baru yang diciptakan penulis berhasil mengurangi peluang terjadinya collision (terutama untuk kasus yang sama). Perubahan yang dilakukan seperti penambahan jumlah penyangga, perputaran bit yang ditentukan oleh jenis penyangga, dan penambahan bit yang ditentukan oleh posisi karakter berhasil meningkatkan keamanan data teruatma dalam kasus pencegahan collision.

5. DAFTAR REFERENSI

[1]. Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006

[2]. Forouzan, Behrouz, *Cryptography and Network Security*, McGraw-Hill, 2008.

[3]. http://en.wikipedia.org/MD5

[4]. http:\\en.wikipedia.org/Hash_Collision

Lampiran

Potongan source code, sisanya hampir secara garis besar kode memiliki fungsi yang sama

```
//inisialisasi penyangga baru

    uint A = 0x01234567;
    uint B = 0x89abcdef;
    uint C = 0xfedcba98;
    uint D = 0x76543210;
    uint E = 0xabcd1111;
    uint F = 0x0000abcd;
    uint G = 0xefab1111;
    uint H = 0x1111efab;

//mengubah text ke array of byte dan diproses sebelum masuk ke MD5

    byte[]ByteContainer = new byte[text.Length];
    for(int i=0;i<text.Length;i++)
    {
            if (i % 2 = 0 ) //ganjil
            {
                    ByteContainer[i] = System.Math.xor((byte) text[i],"11111111")
            }else ) // genap
            {
                    ByteContainer[i] = (byte) text[i];
            }

    }

// Perputaran bit sesuai dengan Jenis penyangga

public void execFF(ref uint a, uint b, uint c, uint d, uint x, uint s, uint ac)
    {
      a += functionF(b, c, d) + x + ac;
      a = leftRotate(a, s) + b;
    }

    public void execGG(ref uint a, uint b, uint c, uint d, uint x, uint s, uint ac)
    {
      a += functionG(b, c, d) + x + ac;
      a = leftRotate(a, s) + b;
                    a = leftRotate(a,s) + b;
    }

    public void execHH(ref uint a, uint b, uint c, uint d, uint x, uint s, uint ac)
```

```
{
   a += functionH(b, c, d) + x + ac;
   a = leftRotate(a, s) + b;
   a = leftRotate (a,s) + b;
   a = leftRotate (a,s) + b;
}

public void execII(ref uint a, uint b, uint c, uint d, uint x, uint s, uint ac)
{
   a += functionI(b, c, d) + x + ac;
   a = leftRotate(a, s) + b;
   a = leftRotate (a,s) + b;
   a = leftRotate (a,s) + b;
   a = leftRotate (a,s) + b;
}
   ...
```