SERANGAN PADA FUNGSI *HASH* MD5

Arya Tri Prabawa – NIM : 13506063

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16063@students.if.itb.ac.id

Abstrak

Penggunaan fungsi *hash* telah menjadi bagian dari penggunaan teknologi modern saat ini. Salah satu kegunaannya adalah dalam dunia kriptografi. Di sini, fungsi *hash* salah satunya digunakan untuk membuat tanda tangan digital.

Seiring dengan perkembangan teknologi, dokumen-dokumen penting tidak lagi ditulis dan didistribusikan di atas kertas, tetapi dibuat dalam bentuk digital. Oleh karena itu, seperti halnya sebuah tanda tangan, diperlukan adanya sebuah mekanisme yang dapat menjamin keaslian dari sebuah dokumen digital.

Fungsi *hash* pun digunakan untuk membangkitkan tanda tangan digital ini. Fungsi *hash* yang banyak digunakan saat ini adalah fungsi *hash* MD5. Namun, kenyataanya, MD5 tidak lagi dapat dikatakan sebagai sebuah fungsi *hash* yang benar-benar aman. Untuk itu kita perlu mengetahui jenis-jenis serangan yang sudah ditemukan dapat terjadi pada MD5.

***Kata kunci**: hash, MD5*, *collision*, *attack*

1. Pendahuluan

Penggunaan fungsi *hash* telah menjadi bagian dari penggunaan teknologi modern saat ini. Salah satu kegunaannya adalah dalam dunia kriptografi. Di sini, fungsi *hash* salah satunya digunakan untuk membuat tanda tangan digital.

Seiring dengan perkembangan teknologi, dokumen-dokumen penting tidak lagi ditulis dan didistribusikan di atas kertas, tetapi dibuat dalam bentuk digital. Oleh karena itu, seperti halnya sebuah tanda tangan, diperlukan adanya sebuah mekanisme yang dapat menjamin keaslian dari sebuah dokumen digital.

Fungsi *hash* pun digunakan untuk membangkitkan tanda tangan digital ini. Fungsi *hash* yang banyak digunakan saat ini adalah fungsi *hash* MD5. Namun, kenyataanya, MD5 tidak lagi dapat dikatakan sebagai sebuah fungsi *hash* yang benar-benar aman. Untuk itu kita perlu mengetahui jenis-jenis serangan yang sudah ditemukan dapat terjadi pada MD5.

Kemungkinan serangan pada MD5 dikemukakan oleh tim kriptografer dari Cina pada konferensi CRYPTO tahun 2004 di mana mereka menyatakan telah menemukan kemungkinan serangan terhadap MD5, yaitu yang berupa tubrukan (collision), di mana dua buah teks yang berbeda dapat memiliki nilai Md5 yang identik.

Namun sebelum membahas lebih lanjut tentang serangan tersebut, terlebih dahulu penulis akan memberikan penjelasan singkat mengenai apa itu fungsi hash MD5 sendiri.

2. Fungsi *Hash* MD5

MD5 ialah fungsi *hash* kriptografik yang digunakan secara luas dengan *hash value* 128-bit. Keluaran dari MD-5 berupa 4 buah blok yang masing-masing berukuran 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai hash[3,10]. Simpul utama MD5 mempunyai blok pesan dengan panjang 512 bit yang masuk ke dalam 4 buah ronde. Hasil keluaran dari MD-5 adalah berupa 128 bit dari byte terendah A dan tertinggi byte D.

3. Cara Kerja Fungsi *Hash* MD5

Terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan yang akan dienkripsi; anggaplah sebanyak b bit. Di sini b adalah bit non negatif integer, b bisa saja nol dan tidak harus selalu kelipatan delapan. Lalu dilakukan 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.

Selanjutnya dilakukan 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 > 264 maka yang diambil adalah panjangnya dalam modulo 264. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo 264. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

Kemudian dilakukan inisialisasi penyangga MD. *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:

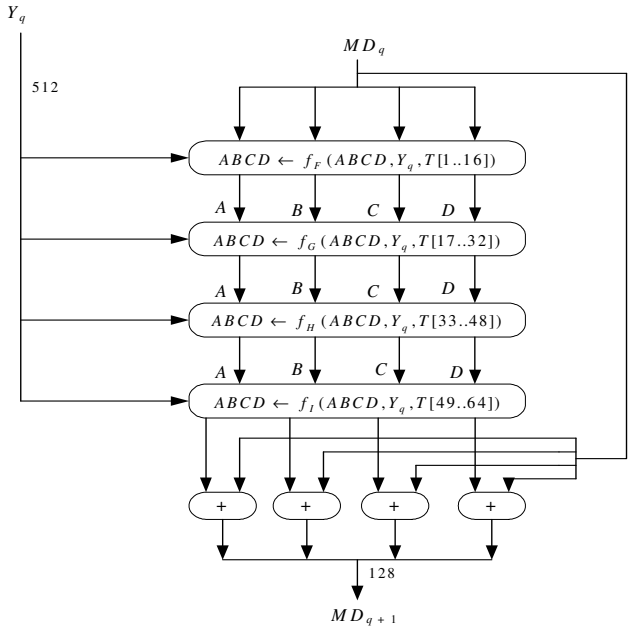
A = 01234567

B = 89ABCDEF

C = FEDCBA98

D = 76543210

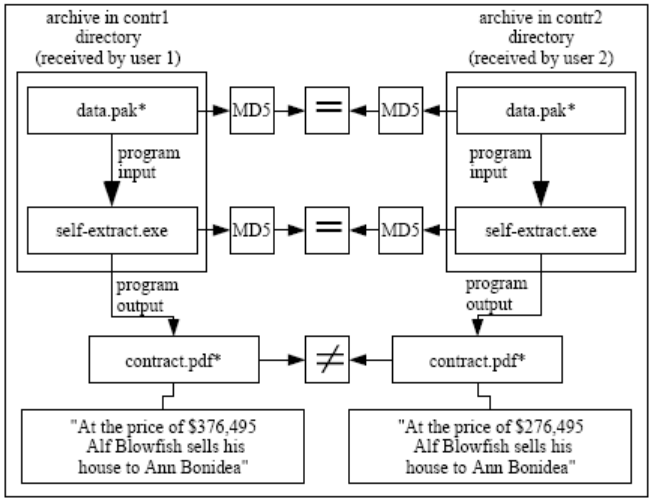
Langkah selanjutnya adalah melakukan pengolahan pesan dalam blok berukuran 512 bit. Pesan dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit (*Y*0 sampai *YL* – 1). Setiap blok 512-bit diproses bersama dengan penyangga *MD* menjadi keluaran 128-bit, dan ini disebut proses *H*MD5. Gambaran proses *H*MD5 diperlihatkan pada Gambar berikut ini.



4. Serangan pada Fungsi *Hash* MD5

Ketika diadakan konferensi CRYPTO pada tahun 2004, tim kriptografer dari Cina menunjukkan bahwa mereka telah menemukan sebuah algoritma untuk menemukan sepasang pesan yang bertabrakan, yaitu pesan yang berbeda dengan hasil *hash* MD5 yang sama. Algoritma tersebut sangat cepat dan dapat menemukan adanya kolisi dalam waktu sekitar satu jam. Fakta ini menunjukkan bahwa MD5 tidak lagi aman untuk mengecek integritas file karena dua file yang berbeda dapat dibuat dengan nilai *hash* yang sama.

Untuk mendemonstrasikannya, digunakan self-extract archive. Pengguna menerima dua file: self-extract.exe dan data.pak. Nilai hash MD5 dari kedua file tersebut dapat diperiksa. Pengguna menjalankan self-extract.exe dan program menggunakan data.pak untuk untuk mengekstraksi dokumennya, yaitu contract.pdf/ Pengguna lain menerima self-extract yang sama, tetapi dengan data.pak yang berbeda. Kedua data.pak dibuat denagn MD5 yang sama sehingga pengguna akan mengira bahwa kontrak yang diekstraksi adalah sama.



Terdapat dua direktori: contr1 dan contr2. Keduanya menunjukkan file yang diterima oleh pengguna yang satu dan lainnya.

Selanjutnya akan dijelaskan langkah-langkah bagaimana seorang penyerang dapat membuat sepasang arsip self-extract yang masing-masing mengekstraksi file yang berbeda sesuai pilihan penyerang.

Data.pak yang telah dibuat memiliki nilai MD5 yang sama. Digunakan sebuah program bernama create-package. Penggunaannya adlaah sebagai berikut:

```
create-package outfile infile1
infile2
```

Outfile menyatakan penamaan file hasil ekstraksi, infile1 dan infile2 menyatakan kedua file yang diletakkan di direktori contr1 dan contr2 di dalam arsip kemudian dinamakan data.pak. Menjalankan program self-extract akan menghasilkan file dengan nama outfile yang sebelumnya menyimpan data dari infile1 atau infile2. Contoh:

```
create-package contract.pdf
contract1.pdf contract2.pdf
```

akan mengambil contract1.pdf dan contract2.pdf lalu meletakkannya ke dalam data1.pak dan data2.pak. Ketika digunakan dengan self-extract, kedua data.pak tersebut akan menghasilkan sebuah file contract.pdf yang menyimpan data dari contract1.pdf dan contract2.pdf.

Size in bytes	Data stored
128	colliding block
1	filename length - *fnamelen*
fnamelen	filename to be extracted
4	32-bit integer size of first stored file - *filesize1*
4	32-bit integer size of second stored file - *filesize2*
filesize1	data of file1
filesize2	data of file2

Kerja program dapat dilihat seperti digambarkan pada tabel di atas. Blok yang bertabrakan berbeda untuk setiap file data.pak dan persis satu dari sepasang string biner yang diberikan oleh sang kriptografer dari Cina. Sedangkan sisa data dari dari kedua data.pak adalah sama.

Ketika menghitung nilai MD5 dari kedua file data.pak, blok bertubrukan yang pertama (1024 bit pertama) menyebabkan konteks hash-nya menjadi identik. Karena sisa data yang ada adalah sama, maka nilai hash yang dihasilkan pun sama.

Program self-extract kemudian menentukan file mana yang akan diekstraksi berdasarkan satu bit padablok yang bertubrukan (yang berbeda pada kedua file data.pak), bit tersebut adalah:

```
#define MD5_COLLISION_OFFSET 19
#define MD5_COLLISION_BITMASK
0x80
```

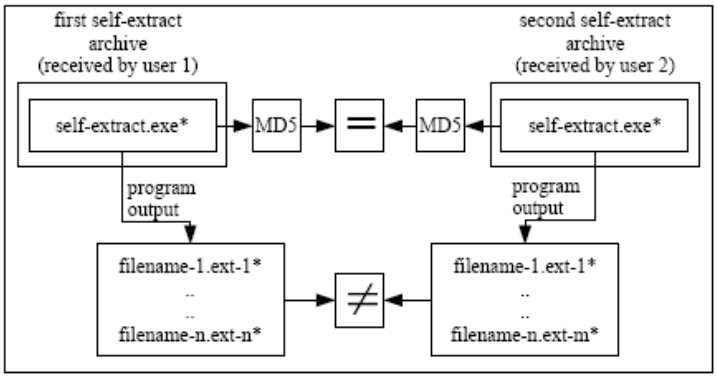
Kedua nilai ini menyatakan posisi dan mask dari bit yang berbeda tersebut. MD5_COLLISION_OFFSET adalah index dari byte yang berbeda dalam file data.pak. Seperti bisa dilihat, byte tersebut berada dalam 128 byte pertama dari blok yang bertubrukan. Sedangkan file executable self-extract.exe adalah sama untuk kedua package sehingga selalu memiliki nilai hash MD5 yang sama.

5. Mengembangkan Serangan

Sebelumnya telah diperlihatkan sebuah serangan menggunakan dua buah file. Kita harus menggunakan string yang telah diketahui dapat menyebabkan terjadinya tubrukan pada hash MD5. Hal ini menghalangi kita untuk meletakkan blok yang bertubrukan pada awal file. Ketika algoritma untuk menemukan tubrukan MD5 untuk intialization vector apapun telah diketahui, blok-blok yang bertubrukan bisa diletakkan pada posisi 512 bit manapun di dalam

file. Pengetahuan tersebut akan memperbolehkan sepasang file executable tunggal untuk dibuat, keduanya memiliki nilai MD5 yang identik, meskipun keduanya akan melakukan aksi yang berbeda tergantung pilihan penyerang. Kini, dua pengguna menerima hanya dua file self-extract tunggal dengan nilai MD5 yang sama. Keduanya tentunya akan mengekstraksi file yang berbeda. File executable self-extract bisanya memang berupa file tunggal sehingga keadaan ini menjadi lebih tidak mudah dicurigai.

Namun untuk mengimplementasikan hal ini, blok yang bertubrukan harus berada pada batas 512 bit di dalam file (hal ini sebagai konsekuensi dari desain algoritma MD5).



6. Kesimpulan

Sebagai fungsi *hash* yang cukup banyak digunakan saat ini, kita patut bersikap waspada terhadap kemungkinan-kemungkinan serangan yang dapat terjadi pada fungsi *hash* MD5.

Dengan sedikit pengembangan, serangan kolisi yang awalnya hanya dapat berlaku jika blok yang bertubrukan berada di awal file, kini dapat terjadi jika blok bertubrukan berada pada 512 bit file. Serangan pun dapat dibuat melalui file self-extract tunggal sehingga mengurangi kemungkinan adanya kecurigaan.

DAFTAR PUSTAKA

[1] Mikle, Ondrej. 2004. *Practical Attacks on Digital Signatures Using MD5 Message Digest*. Charles University, Prague.

[2] Munir, Rinaldi. 2004. *Bahan Kuliah IF5054 Kriptografi*. Departemen Teknik Informatika, Institut Teknologi Bandung.