Kriptoanalisis Pada Fungsi Hash Kriptografi MD5

Sibghatullah Mujaddid (13507124)

Jurusan Teknik Informatika ITB, Bandung 40132, email: sibgha07@students.itb.ac.id

Abstract – *Dalam ilmu komputer, fungsi hash sering digunakan untuk penyimpanan dalam database. Selain untuk database, fungsi hash juga dapat digunakan dalam kriptografi. Ada berbagai macam fungsi hash dalam kriptografi yang dapat digunakan. Makalah ini membahas kriptanalisis salah satu fungsi hash kriptografi yaitu MD5.*

Kata Kunci: *hash, kriptografi, MD5.*

1. PENDAHULUAN

Kriptografi adalah ilmu yang mempelajari bagaimana membuat suatu pesan yang dikirim oleh pengirim, dapat tersampaikan dengan aman pada penerima dengan cara menyamarkannya dalam bentuk sandi yang tidak mempunyai makna. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data [A. Menezes, P. van Oorschot and S. Vanstone - Handbook of Applied Cryptography]. Tidak semua aspek keamanan informasi ditangani oleh kriptografi. Namun, keutamaan dari kriptografi adalah menjaga kerahasiaan pesan dari penyadap dan kriptanalis.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu :

- Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
- Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
- Autentikasi, adalah berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
- Non-repudiasi., atau nirpenyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/membuat.

Ada tiga jenis utama algoritma kriptografi, yaitu:

- Algoritma Simetri

  a. Blok Chiper : DES, IDEA, AES

  b. Stream Chiper : OTP, A5 dan RC4
- Algoritma Asimetri : RSA, DH, ECC, DSA
- Fungsi Hash : MD, SHA

2. FUNGSI HASH KRIPTOGRAFI

Hash adalah suatu teknik "klasik" dalam Ilmu Komputer yang banyak digunakan dalam praktek secara mendalam. *Hash* merupakan suatu metode yang secara langsung mengakses record-record dalam suatu tabel dengan melakukan transformasi aritmatik pada key yang menjadi alamat dalam tabel tersebut. Key merupakan suatu input dari pemakai di mana pada umumnya berupa nilai atau string karakter.

Fungsi Hash Kriptografi adalah fungsi *hash* yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Fungsi *hash* adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai *hash*.

Pelacakan dengan menggunakan *hash* terdiri dari dua langkah utama, yaitu:

1. Menghitung Fungsi Hash. Fungsi *hash* adalah suatu fungsi yang mengubah key menjadi alamat dalam tabel. Fungsi *hash* memetakan sebuah key ke suatu alamat dalam tabel. Idealnya, key-key yang berbeda seharusnya dipetakan ke alamat-

alamat yang berbeda juga. Pada kenyataannya, tidak ada fungsi *hash* yang sempurna. Kemungkinan besar yang terjadi adalah dua atau lebih key yang berbeda dipetakan ke alamat yang sama dalam tabel. Peristiwa ini disebut dengan collision (tabrakan). Karena itulah diperlukan langkah berikutnya, yaitu collision resolution (pemecahan tabrakan).

2. Collision Resolution. Collision resolution merupakan proses untuk menangani kejadian dua atau lebih key di-hash ke alamat yang sama. Cara yang dilakukan jika terjadi collision adalah mencari lokasi yang kosong dalam tabel *hash* secara terurut. Cara lainnya adalah dengan menggunakan fungsi *hash* yang lain untuk mencari lokasi kosong tersebut.

2.1 Sifat-Sifat Fungsi Hash Kriptografi

- Tahan preimej (*Preimage resistant*): bila diketahui nilai hash *h* maka sulit (secara komputasi tidak layak) untuk mendapatkan *m* dimana *h* = hash(*m*).
- Tahan preimej kedua (*Second preimage resistant*): bila diketahui input m_1 maka sulit mencari input m_2 (tidak sama dengan m_1) yang menyebabkan hash(m_1) = hash(m_2).
- Tahan tumbukan (*Collision-resistant*): sulit mencari dua input berbeda m_1 dan m_2 yang menyebabkan hash(m_1) = hash(m_2)

2.2 Algoritma-Algoritma Fungsi Hash Kriptografi

Beberapa contoh algoritma fungsi hash Kriptografi:

- MD4
- MD5
- SHA-0
- SHA-1
- SHA-256
- SHA-512

3. MD5

Dalam kriptografi, MD5 (*Message-Digest algortihm 5*) ialah fungsi *hash* kriptografik yang digunakan secara luas dengan nilai *hash* 128-bit. Pada standard Internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah file.

MD5 di desain oleh Ronald Rivest, salah satu pembuat algoritma RSA, pada tahun 1991 untuk menggantikan *hash function* sebelumnya, MD4, yang berhasil diserang oleh kriptanalis. MD5 merupakan kelanjutan daru MD4 yang dirancang dengan tujuan keamanan. Secara perhitungan matetamatis tidak dimungkinkan untuk mendapatkan dua pesan yang memiliki hash yang sama. Tidak ada serangan yang lebih efisien untuk membongkar/mengetahui hash suatu pesan selain brute-force.

3.1 SEJARAH DAN KRIPTOANALISIS

MD5 adalah salah satu dari serangkaian algortima *message digest* yang didesain oleh Profesor Ronald Rivest dari MIT (Rivest, 1994). Saat kerja analitik menunjukkan bahwa pendahulu MD5 — MD4 — mulai tidak aman, MD5 kemudian didesain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin).

Pada tahun 1993, den Boer dan Bosselaers memberikan awal hasil dari penemuan *pseudo-collision* dari fungsi kompresi MD5. Dua vektor inisialisasi berbeda *I* dan *J* dengan beda 4-bit diantara keduanya.

$$MD5compress(I,X) = MD5compress(J,X)$$

Pada tahun 1996 Dobbertin mengumumkan sebuah kerusakan pada fungsi kompresi MD5. Dikarenakan hal ini bukanlah serangan terhadap fungsi *hash* MD5 sepenuhnya, hal ini menyebabkan para pengguna kriptografi menganjurkan pengganti seperti WHIRLPOOL, SHA-1 atau RIPEMD-160.

MD5CRK berhenti pada tanggal 17 Agustus 2004, saat kerusakan *hash* pada MD5 diumumkan oleh Xiaoyun Wang, Dengguo Feng, Xuejia Lai dan Hongbo Yu. Serangan analitik mereka dikabarkan hanya memerlukan satu jam dengan menggunakan IBM P690 cluster.

Pada tanggal 1 Maret 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger mendemontrasikan konstruksi dari dua buah sertifikat X.509 dengan *public key* yang berbeda dan *hash* MD5 yang sama, hasil dari demontrasi menunjukkan adanya kerusakan. Konstruksi tersebut melibatkan *private key* untuk kedua *public key* tersebut. Dan beberapa hari setelahnya, Vlastimil Klima menjabarkan dan mengembangkan algortima, mampu membuat kerusakan Md5 dalam beberapa jam dengan menggunakan sebuah komputer notebook. Hal ini menyebabkan MD5 tidak bebas dari kerusakan.

Dikarenakan MD5 hanya menggunakan satu langkah pada data, jika dua buah awalan dengan

hash yang sama dapat dibangun, sebuah akhiran yang umum dapat ditambahkan pada keduanya untuk membuat kerusakan lebih masuk akal. Dan dikarenakan teknik penemuan kerusakan mengijinkan pendahuluan kondisi *hash* menjadi arbitari tertentu, sebuah kerusakan dapat ditemukan dengan awalan apapun. Proses tersebut memerlukan pembangkitan dua buah file perusak sebagai file templat, dengan menggunakan blok 128-byte dari tatanan data pada 64-byte batasan, file-file tersebut dapat mengubah dengan bebas dengan menggunakan algoritma penemuan kerusakan.

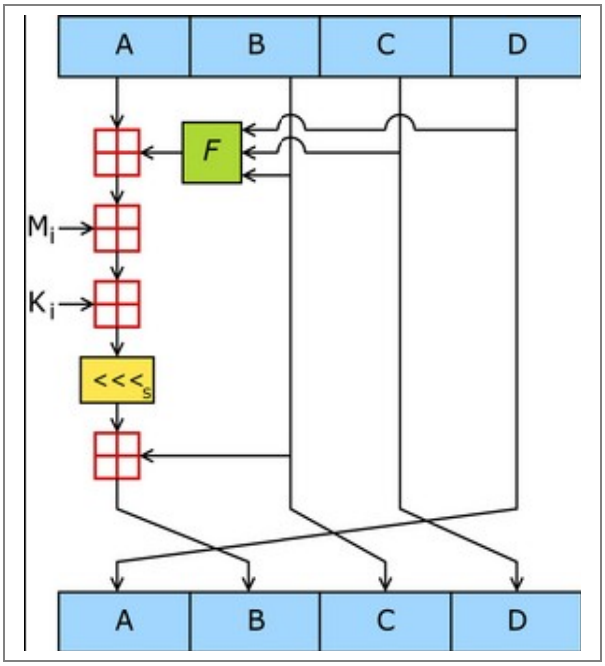
Proses pembangkitan kerusakan MD5 kini dapat diketahui dengan mudah yaitu dengan membangkitkan dua byte *string* dengan *hash* yang sama. Sebagai hasilnya bahwa *hash* MD5 dari informasi tertentu tidak dapat lagi mengenalinya secara berbeda. Jika ditunjukkan informasi dari sebuah *public key*, *hash* MD5 tidak mengenalinya secara berbeda jika terdapat *public key* selanjutnya yang mempunyai *hash* MD5 yang sama.

3.2 PENGUJIAN INTEGRITAS

MD5 digunakan secara luas dalam dunia perangkat lunak untuk menyediakan semacam jaminan bahwa file yang diambil (*download*) belum terdapat perubahan. Seorang user dapat membandingkan MD5 sum yang dipublikasikan dengan *checksum* dari file yang diambil. Dengan asumsi bahwa *checksum* yang dipublikasikan dapat dipercaya akan keasliannya, seorang user dapat secara yakin bahwa file tersebut adalah file yang sama dengan file yang dirilis oleh para developer, jaminan perlindungan dari *Trojan Horse* dan virus komputer yang ditambahkan pada perangkat lunak. Namun, banyak kasus yang terjadi bahwa *checksum* yang dipublikasikan tidak dapat dipercaya (sebagai contoh, *checksum* didapat dari channel atau lokasi yang sama dengan tempat mengambil file), dalam hal ini MD5 hanya mampu melakukan *error-checking*. MD5 akan mengenali file yang didownload tidak sempurna, cacat atau tidak lengkap.

3.3 CARA KERJA MD5

MD5 mengolah blok 512 bit, dibagi kedalam 16 subblok berukuran 32 bit. Keluaran algoritma diset menjadi 4 blok yang masing-masing berukuran 32 bit yang setelah digabungkan akan membentuk nilai hash 128 bit.



Gambar 1. Algoritma MD5

Pesan diberi tambahan sedemikian sehingga panjang menjadi k-bit, dimana k = 512n – 64 bit. n merupakan blok masukan. Tambahan ini diperlukan hingga pesan menjadi k bit. Kemudian 64 bit yang masing kosong, dibagian akhir, diisi panjang pesan. Inisiasi 4 variabel dengan panjang 32 bit yaitu a,b,c,d. Variabel a,b,c,d dikopikan ke variabel a,b,c,d yang kemudian diolah melalui 4 tahapan yang sangat serupa. Setiap tahapan menggunakan 16 kali operasi berbeda, menjalankan fungsi nonlinear pada tiga variabel a,b,c, atau d. Hasilnya ditambahkan ke variabel keempat, subblok pesan dan suatu konstanta. Kemudian dirotasi kekiri beberapa bit yang kemudian ditambahkan ke salah satu dari a,b,c, atau d. Kemudian nilai a,b,c, dan d menggantikan nilai a,b,c, dan d. Kemudian dikeluarkan output yang merupakan gabungan dari a,b,c, dan d. Fungsi kompresi yang digunakan oleh algoritma MD5 adalah sebagai berikut : a ←b + ((a + g (b,c,d) + X[k] + T[i] <<< s), dimana g adalah salah fungsi primitif F,G,H,I seperti dibawah ini :

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

dan operasi XOR, AND, OR, dan NOT adalah sebagai berikut :

$$\oplus, \wedge, \vee, \neg$$

3.4 PSEUDOCODE

Pseudocode pada algoritma MD5 adalah sebagai berikut.

```
//Catatan: Seluruh variable tidak pada 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)
```

Catatan: Meskipun rumusan dari yang tertera pada RFC 1321, berikut ini sering digunakan untuk meningkatkan efisiensi:

(0 ≤ i ≤ 15): f := d **xor** (b **and** (c **xor** d))
(16 ≤ i ≤ 31): f := c **xor** (d **and** (b **xor** c))

3.4 HASH-HASH MD5

Hash-hash MD5 sepanjang 128-bit (16-byte), yang dikenal juga sebagai *ringkasan pesan*, secara tipikal ditampilkan dalam bilangan heksadesimal 32-digit. Berikut ini merupakan contoh pesan ASCII sepanjang 43-byte sebagai masukan dan *hash* MD5 terkait:

```
MD5("The quick brown fox jumps over the lazy dog") = 9e107d9d372bb6826bd81d3542a419d6
```

Bahkan perubahan yang kecil pada pesan akan (dengan probabilitas lebih) menghasilkan *hash* yang benar-benar berbeda, misalnya pada kata "dog", huruf `d` diganti menjadi `c`:

```
MD5("The quick brown fox jumps over the lazy cog") = 1055d3e698d289f2af8663725127bd4b
```

Hash dari panjang-nol ialah:

```
MD5("") = d41d8cd98f00b204e9800998ecf8427e
```

4. KESIMPULAN

Fungsi *hash* adalah fungsi yang bisa digunakan untuk berbagai keperluan. Fungsi *hash* dalam kriptografi memiliki beberapa sifat tambahan yang dapat digunakan dalam pengamanan data. Jika dalam database fungsi *hash* digunakan untuk memudahkan penyimpanan key dalam tabel *hash*, pada kriptografi, fungsi *hash* digunakan untuk memastikan kebenaran pesan yang dikirim dengan cara membandingkan nilai-nilai *hash* yang diperoleh.

MD5 saat ini tidak lagi direkomendasikan untuk digunakan karena banyak memiliki kelemahan, antara lain:

1. Diggest MD5 yang kurang panjang sehingga dapat dengan mudah diserang dengan *brute-force*.

2. Kelemahan MD5 ada pada design sehingga lebih mudah dilakukan kriptanalisis.

DAFTAR REFERENSI

http://en.wikipedia.org/wiki/MD5, waktu akses: 2 Januari 2009.

http://id.wikipedia.org/wiki/Kriptografi, waktu akses: 2 Januari 2009.

http://lmukomputer.org/2007/03/27/md5-dan-sha-1-kriptografi-dengan-fungsi-hash/, waktu akses: 2 Januari 2009