Modifikasi SHA-1 Untuk Mengurangi *Hash collision*

Odit Ekwardo – 135 04 079
Jurusan Teknik Informatika ITB, Bandung, email: if14079@students.if.itb.ac.id

Abstraksi – *Komunikasi merupakan faktor penting dalam kehidupan manusia. Kini manusia semakin dipermudah oleh teknologi untuk menyampaikan informasi. Media–media komunikasi yang diciptakan manusia tersebut memang memudahkan penyampaian informasi, tapi di sisi lain penyampaian pesan melalui media tertentu tidak menjamin keamanan terhadap kerahasiaan pesan. Oleh karena itu dikembangkan ilmu kriptografi untuk menjaga keamanan informasi saat penyampaian melalui media informasi. Salah satu teknik kriptografi modern adalah algoritma SHA-1. SHA-1 merupakan salah satu fungsi hash yang dikembangkan untuk menghasilkan sebuah message digest. Namun, SHA-1 masih memiliki kelemahan, karena mudah terjadi hash collision. Modifikasi terhadap teknik ini dapat dilakukan untuk mengurangi hash collision. Salah satu caranya adalah dengan menambhakan jumlah iterasi yang ada pada algoritma SHA-1. Message digest yang dihasilkan menggunakan modifikasi ini lebih rumit sehingga kemungkinan terjadinya hash collision dapat dikurangi. Modifikasi dengan penambahan jumlah iterasi dapat mengurangi terjadinya hash collision karena dengan pemrosesan ganda, message digest yang dihasilkan menjadi lebih rumit dan kemungkinan output yang dihasilkan mejadi lebih banyak. Dengan begitu, hash collision dapat dikurangi.*

Kata Kunci*: SHA-1, Fungsi Hash, Hash collision.*

1 PENDAHULUAN

Dalam hidup manuasia salah satu faktor penting yang mempengaruhi kemajuan manusia adalah komunikasi. Tanpa komunikasi manusia tidak akan bisa berhubungan, berinteraksi dan bertukar pikiran untuk membuat sesuatu untuk kemajuan manusia itu sendiri dan menyampaikannya pada orang lain. Dilatarbelakangi oleh kebutuhan manusia tersebut, teknologi komunikasi dewasa ini maju dengan pesat. Dengan kemajuan teknologi tersebut, manusia dapat melakukan pengiriman pesan dengan mudah dimana saja dan kapan saja dengan menggunakan berbagai media.

Media–media komunikasi yang diciptakan manusia tersebut memang memudahkan penyampaian informasi, tapi di sisi lain penyampaian pesan melalui media tertentu tidak menjamin keamanan terhadap kerahasiaan pesan. Banyak pihak yang tidak bertanggung jawab memanfaatkan ketidakamanan tersebut untuk mencuri atau menyadap pesan milik orang lain untuk disalahgunakan.

Masalah keamanan pengiriman pesan mulai dikembangkan oleh manusia untuk melindungi para pengguna media informasi. Diantaranya dengan memperkenalkan teknologi kriptografi. Telah banyak algoritma kriptografi, baik kriptografi modern maupun klasik yang sudah diterapkan untuk menjaga keamanan suatu pesan. Konsep penggunaan kriptografi pada pengiriman pesan adalah dengan meng-enkripsi pesan yang dikirim menjadi cipherteks, kemudian si penerima men-dekripsi menjadi pesan asli. Dengan begitu, jika ada pihak yang menyadap pesan saat pengiriman, pesan yang disadap masih berupa cipherteks yang tidak memiliki arti. Selain itu kini juga telah populer penggunaan digital signature untuk menjamin keabsahan suatu dokumen telah dikirim oleh orang yang benar.

Salah satu algoritma kriptografi modern untuk membuat digital signature yang cukup populer adalah SHA. SHA ini memiliki berbagai versi, yaitu SHA-0, SHA-1, SHA-256, SHA-512, dan SHA-384. Yang akan dibahas pada makalah ini adalah SHA-1.

SHA-1 merupakan salah satu teknik kriptografi untuk membuat digital signature yang pernah populer dan sudah cukup lama digunakan orang-orang, sehingga para kriptanalis di dunia pun sudah banyak yang menemukan cara untuk memecahkan teknik ini. Ada banyak cara untuk mengurangi serangan terhadap SHA-1, salah satunya adalah dengan mengurangi *hash collision*.

Upaya untuk mengurangi *hash collision* yang terjadi pada algoritma SHA-1 dapat dilalui melalui berbagai cara, salah satunya dengan menambah jumlah iterasi algoritma atau dengan

2 DASAR TEORI

2.1 SHA-1

Fungsi Hash SHA adalah lima fungsi hash kriptografis yang dirancang oleh National Security Agency (NSA) dan yang diterbitkan oleh NIST sebagai U.S. *Federal Information Processing Standard*. SHA adalah kepanjangan dari Secure Hash Algorithm. Algoritma-algoritma hash menghitung suatu representasi digital dengan panjang yang pasti atau telah ditentukan (yang yang dikenal sebagai suatu *message digest*) dari suatu urutan data masukan (pesan) yang panjangnya

bermacam-macam. Mereka disebut "aman" ketika, "ini dapat dikomputasi untuk:

1. Menemukan suatu pesan yang berpasangan dengan suatu *message digest* yang diberi, atau

2. Menemukan dua pesan yang berbeda yang menghasilkan *message digest* yang sama.

Setiap perubahan akan, dengan kemungkinan yang sangat tinggi, menghasilkan suatu *message digest* yang berbeda."

Lima algoritma SHA yang telah diciptakan dinamai dengan SHA-1, SHA-224, SHA-256, SHA-384, dan SHA-512. Empat varian terakhir biasanya dikenal sebagai SHA-2. SHA-1 menghasilkan suatu *message digest* dengan panjang 160 bit; nomor di dalam nama-nama empat algoritma yang lainnya menandakan panjangnya bit dari digest yang mereka hasilkan.

SHA-1 digunakan secara luas di dalam beberapa aplikasi-aplikasi keamanan dan protokol-protokol, termasuk TLS dan SSL, PGP, SSH, S/MIME, dan IPsec. SHA-1 dianggap sebagai pengganti lanjutan MD5, algoritma yang telah populer sebelumnya.

Keamanan SHA-1 telah dibicarakan oleh para ahli kriptografi. Meski tidak ada serangan-serangan telah diberitakan varian-varian SHA-2, mereka secara algoritma serupa dengan SHA-1, dengan demikian masih dilakukan usaha-usaha untuk meningkatkan algoritma fungsi hash alternatif.

Pseudocode untuk SHA-1 adalah sebagai berikut:

```
Note: All variables are unsigned 32 bits and wrap modulo 232 when calculating

Initialize variables:
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0

Pre-processing:
append the bit '1' to the message
append k bits '0', where k is the minimum number ≥ 0 such that the resulting message
    length (in bits) is congruent to 448 (mod 512)
append length of message (before pre-processing), in bits, as 64-bit big-endian integer
```

```
Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 <= i <= 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp

    Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian):
digest = hash = h0 append h1 append h2 append h3 append h4
```

2.2 *Hash collision*

Di dalam ilmu pengetahuan komputer, *hash collision* adalah suatu situasi yang terjadi ketika dua masukan yang berbeda pada suatu fungsi hash menghasilkan keluaran yang identik.

Semua fungsi hash berpotensial untuk menghasilkan *hash collision*, meskipun demikian fungsi hash yang dirancang dengan baik dapat mengurangi intensitas terjadinya *hash collision* (bandingkan dengan suatu fungsi dengan kurang baik merancang) atau lebih sulit untuk ditemukan. Didalam aplikasi-aplikasi yang menggunakan kemungkinan data input yang relatif adalah yang dikenal sebelum waktu yang ditetapkan dimungkinkan untuk membangun suatu fungsi hash yang sempurna dengan memetakan semua masukan pada keluaran-keluaran yang berbeda. Bagaimanapun, banyak sekali fungsi hash, termasuk fungsi hash yang sering digunakan dalam kriptografi, menghasilkan suatu keluaran ukuran yang ditetapkan dari satu pesan yang panjang. Dalam desain yang demikian, akan selalu ada *hash collision*, karena setiap fungsi hash yang diberi harus berpasangan dengan satu bilangan tak hingga dari masukan-masukan yang mungkin.

Contoh terjadinya *hash collision* adalah sebagai berikut:

$f(x) = f(y)\ ;\ x \neq y$
(untuk nilai masukan yang berbeda x dan y, fungsi hash *f* menghasilkankan nilai keluaran yang sama)

3 HASIL DAN PEMBAHASAN

3.1 Modifikasi SHA-1

Hash collision pada SHA-1, maka akan lebih mudah dilakukan kriptanalisis terhadap algoritma kriptografi ini

Seperti yang telah dijelaskan dalam bab pendahuluan, terdapat berbagai cara untuk mengurangi terjadinya *hash collision* pada algoritma SHA-1, diantaranya adalah dengan menambah jumlah iterasi dalam algoritma SHA-1.

Pada *pseudocode* SHA-1 kita dapat melihat bahwa terjadi iterasi pada bagian yang dinamakan "Main Loop". Pada bagian tersebut dilakukan perhitungan untuk mencari salah satu nilai dari penyusun *message digest* yang akan dikeluarkan oleh algoritma SHA-1. Dengan menambah jumlah iterasi maka perhitungan akan dilakukan dua kali sehingga *message digest* yang dihasilkan akan lebih rumit dibandingkan dengan sebelumnya.

Untuk lebih mempermudah pembahasan modifikasi, berikut ini ditampilkan potongan pseudocode SHA-1 pada bagian yang dimodifikasi, yaitu pada bagian "Main Loop". Potongan kodenya adalah sebagai berikut:

```
Main loop:

penambahan kode untuk penambahan iterasi
for j from 0 to 1

    for i from 0 to 79

        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
```

Penambahan iterasi pada main loop membuat proses pengisian variable a, b, c, d, dan e pada algoritma SHA-1 menjadi lebih rumit dan kemungkinan keluarannya pun menjadi bertambah.

Hasil algoritma SHA-1 secara keseluruhan setelah dimodifikasi adalah sebagai berikut:

```
Note: All variables are unsigned 32 bits and wrap modulo 232 when calculating

Initialize variables:
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0
```

```
Pre-processing:
append the bit '1' to the message
append k bits '0', where k is the minimum number ≥ 0 such that the resulting message
    length (in bits) is congruent to 448 (mod 512)
append length of message (before pre-processing), in bits, as 64-bit big-endian integer


Process the message in successive 512-bit chunks:
break message into 512-bit chunks
for each chunk
    break chunk into sixteen 32-bit big-endian words w[i], 0 <= i <= 15

    Extend the sixteen 32-bit words into eighty 32-bit words:
    for i from 16 to 79
        w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor w[i-16]) leftrotate 1

    Initialize hash value for this chunk:
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    Main loop:
for j from 0 to 1
    for i from 0 to 79
        if 0 ≤ i ≤ 19 then
            f = (b and c) or ((not b) and d)
            k = 0x5A827999
        else if 20 ≤ i ≤ 39
            f = b xor c xor d
            k = 0x6ED9EBA1
        else if 40 ≤ i ≤ 59
            f = (b and c) or (b and d) or (c and d)
            k = 0x8F1BBCDC
        else if 60 ≤ i ≤ 79
            f = b xor c xor d
            k = 0xCA62C1D6

        temp = (a leftrotate 5) + f + e + k + w[i]
        e = d
        d = c
        c = b leftrotate 30
        b = a
        a = temp
```

```
Add this chunk's hash to result so far:
    h0 = h0 + a
    h1 = h1 + b
    h2 = h2 + c
    h3 = h3 + d
    h4 = h4 + e

Produce the final hash value (big-endian):
digest = hash = h0 append h1 append h2 append h3 append h4
```

3.2 Perbandingan *message digest* yang dihasilkan algoritma SHA-1 sebelum modifikasi dan setelah dimodifikasi

Untuk mengetahui lebih jelasnya perbandingan antara algoritma SHA-1 sebelum dan sesudah dimodifikasi, dilakukan pembandingan secara langsung *message digest* yang dihasilkan oleh algoritma SHA-1 sebelum dan seudah dilakukan modifikasi.

Jika diberikan sebuah pesan teks “The quick brown fox jumps over the lazy dog” maka *message digest* yang dihasilkan oleh masing-masing algoritma adalah sebagai berikut:

SHA-1(“The quick brown fox jumps over the lazy dog”) = 2fd4e1c6 7a2d28fc ed849ee1 bb76e739 1b93eb12

SHA-1-modif(“The quick brown fox jumps over the lazy dog”) = ih45iu6h d25e1b3a fad3e85a jn75w46v 211en5gz

Dapat dilihat dari hasil *message digest* yang dihasilkan terjadi perbedaan yang sangat signifikan.

3.3 Analisis pengaruh modifikasi terhadap pengurangan terjadinya *hash collision*

Modifikasi algoritma SHA-1 dengan cara menambahkan jumlah looping pada bagian main loop memberikan pengaruh yang signifikan terhadap *message digest* yang dihasilkan.Ternyata modifikasi ini berhasil mengurangi kemungkinan *hash collision* karena terjadi pemrosesan ganda saat membuat *message digest*. Penambahan iterasi menyebabkan input diproses dua kali yang menyebabkan kemungkinan output *message digest* yangh dihasilkan menjadi lebih banyak. Dengan demikian terjadinya *hash collision* dapat dikurangi

4 KESIMPULAN

Dari berbagai macam pembahasan dan penelitian yang telah dilakukan untuk membuat makalah ini, dapat diambil kesimpulan, yaitu:

1. Algoritma SHA-1 merupakan salah satu teknik

kriptografi modern, yang pernah populer digunakan oleh orang di dunia untuk menjaga keamanan pesan yang dikirimkannya, yang menggunakan aturan fungsi hash untuk menghasikan suatu *message digest* dengan panjang yang tetap yaitu 160 bit

2. Modifikasi yang dilakukan terhadap algoritma SHA-1 pada penelitian kali ini adalah dengan menambahkan jumlah looping mada main loop yang telah ada pada algoritma SHA-1.
3. Modifikasi algoritma SHA-1 dengan cara menambah jumlah looping pada main loop hanya merupakan salah satu dari banyak cara yang dapat digunakan untuk memodifikasi algoritma agar kejadian *hash collision* dapat dikurangi..

DAFTAR REFERENSI

[1] Munir, Rinaldi, Diktat Kuliah IF5054 Kriptografi, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, 2006.

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

[3] http://en.wikipedia.org/wiki/Sha-1

[4] http://en.wikipedia.org/wiki/Hash_collision