Studi dan Analisis Kolisi pada Fungsi Hash SHA-1

Hapsari Tilawah 13509027[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509027@std.stei.itb.ac.id

***Abstract*— Dewasa ini bidang teknologi informasi mengalami perkembangan yang pesat, salah satu dampaknya adalah pertukaran informasi melalui bentuk data digital. Namun banyak sekali timbul permasalahan-permasalahan baru terkait dengan data-data yang beredar, salah satunya mengenai integritas data. Salah satu cara untuk mengetahui tingkat integritas data adalah dengan menggunakan fungsi hash. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai hash / message digest. Fungsi hash SHA-1 adalah salah satu fungsi hash yang cukup banyak digunakan. Pada makalah ini, penulis akan membahas kolisi pada SHA-1, serangan-serangan kolisi pada SHA-1, eksperimen dan analisis, serta terakhir, penulis akan menyimpulkan apakah algoritma SHA-1 masih layak digunakan sebagai sebuah metode untuk menguji integritas data atau tidak.**

***Index Terms*— Fungsi Hash, integritas data, kolisi, SHA-1, serangan kolisi**

I. PENDAHULUAN

Perkembangan dunia digital saat ini membuat lalu lintas pengiriman data elektronik semakin ramai. Hampir setiap orang melakukan transaksi data setiap harinya. Oleh karena itu, pentingnya keamanan dalam pengantaran datanya perlu diutamakan apalagi jika data tersebut mengandung informasi yang sangat serius. Untuk mengatasi masalah keamanan pengiriman pesan digtal tersebut, dikembangkanlah teknologi kriptografi untuk melindungi para pengguna media informasi.

Kriptografi merupakan ilmu yang mempelajari bagaimana membuat suatu pesan yang dikirim pengirim dapat disampaikan kepada penerima dengan aman. Selain itu, pesan yang dikirim harus terjaga kerahasiaannya dari penyerang atau kriptanalis. Konsep penggunaan kriptografi pada pengiriman pesan adalah dengan mengenkripsi pesan yang dikirim menjadi cipherteks, kemudian penerima mendekripsi menjadi pesan asli. Dengan begitu, jika ada pihak yang menyadap pesan saat pengiriman, pesan yang disadap masih berupa cipherteks yang tidak memiliki arti. Telah banyak algoritma kriptografi, baik kriptografi modern maupun klasik yang sudah diterapkan untuk menjaga keamanan suatu pesan.

Terdapat beberapa permasalahan dalam kriptografi berkaitan dengan masalah keamanan yang tidak dapat diselesaikan dengan fungsi enkripsi dan dekripsi biasa. Masalah keamanan yang harus dihadapi itu antara lain masalah verifikasi dan integritas data. Fungsi hash adalah salah satu fungsi dalam kriptografi yang mampu memenuhi kebutuhan akan verifikasi dan integritas data. Dengan menggunakan fungsi hash satu arah kita dapat mengetahui apakah data yang kita terima dari sumber adalah data yang benar dan tidak mengalami perubahan. Misalnya, fungsi hash bersama algoritma enkripsi kunci publik telah digunakan sebagai fitur dalam digital signature, selain itu fungsi hash juga digunakan dalam penyimpanan *password*.

Algoritma untuk melakukan fungsi hash cukup banyak jumlahnya. Akan tetapi, saat ini yang cukup populer dipakai adalah algoritma hash SHA-1. Pada awalnya, algoritma SHA-1 ini dianggap cukup aman karena satu-satunya cara untuk mencari kolisi (*collision*) adalah dengan menggunakan serangan *brute-force* yang memiliki kompleksitas operasi sebesar 2^{80}, yang dengan super komputer yang ada sekalipun hanya mampu dicari dalam hitungan jutaan tahun. Namun sampai saat ini sudah ditemukan beberapa serangan terhadap SHA-1 yang mampu mencari kolisi dengan kompleksitas kurang dari 2^{80}. Karena itu penulis ingin melakukan sebuah eksplorasi mengenai kolisi pada SHA-1, serangan-serangan kolisi pada SHA-1, eksperimen dan analisis, serta terakhir apakah algoritma SHA-1 masih layak digunakan sebagai sebuah metode untuk menguji integritas data atau tidak.

II. DASAR TEORI

A. Fungsi Hash

Secara umum fungsi hash adalah sebuah fungsi kriptografi yang menerima masukan *string* dengan panjang dan ukuran sembarang dan mengubahnya menjadi *string* keluaran yang panjangnya selalu tetap (*fixed length*) yang umumnya berukuran jauh lebih kecil daripada ukuran *string* semula. Keluaran dari fungsi hash disebut nilai hash (*hash value*) atau *message digest*. Keluaran fungsi hash pasti tidak sama untuk setiap pesan yang berbeda.

Hash punya beberapa nama lain, yaitu: fungsi kompresi, *fingerprint*, *cryptographic checksum* (MIC), dan *manipulation detection code* (MDC). Pada umumnya hash yang dipakai untuk keperluan konsistensi dan otentikasi adalah fungsi hash satu arah, disebut fungsi hash satu arah karena pesan yang diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula (*irreversible*).

Fungsi hash yang baik memiliki beberapa sifat dan karakteristik. Sifat-sifat tersebut adalah sebagai berikut:

1. Fungsi hash dapat diterapkan pada blok data berukuran berapa saja.
2. Hash menghasilkan nilai hash dengan panjang tetap (*fixed-length output*).
3. H(x) mudah dihitung untuk setiap nilai x yang diberikan.
4. *Preimage Resistant*, yaitu untuk setiap nilai hash h yang diberikan, tidak mungkin menemukan pesan x sedemikian sehingga H(x) = h. Itulah sebabnya fungsi H dikatakan fungsi hash satu arah (*one-way hash function*).
5. *Second Preimage Resistant*, yaitu untuk setiap pesan x yang diberikan, tidak mungkin mencari pesan y yang tidak sama dengan pesan x (y != x) sedemikian sehingga H(y) = H(x).
6. *Collision Resistant*, yaitu tidak mungkin secara komputasi mencari pasangan pesan x dan y sedemikian sehingga keduanya memiliki nilai hash yang sama (H(x) = H(y)).

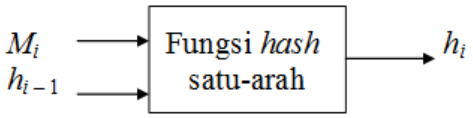
Keenam karakteristik diatas penting sebab sebuah fungsi hash seharusnya berlaku seperti fungsi acak. Karakteristik nomor lima dan enam merupakan aspek yang hangat dibicarakan dalam pengujian keamanan suatu fungsi hash. Serangan terhadap suatu fungsi hash pun dikonsentrasikan terhadap kedua jenis hal tersebut. Serangan-serangan tersebut disebut juga sebagai berikut:

1. *Preimage attacks*: Suatu bentuk penyerangan terhadap karakteristik nomor 5, yaitu serangan dengan berusaha mencari suatu pesan yang memiliki nilai hash yang sama dengan suatu pesan yang telah terdefinisi sebelumnya.
2. *Collision attacks*: Sutu bentuk serangan terhadap karakteristik nomor 6, yaitu serangan dengan berusaha mencari dua pesan yang memiliki nilai hash yang sama.

Fungsi hash memiliki algoritma yang iteratif dan searah, yang dapat memproses pesan yang diberikan untuk menghasilkan representasi yang lebih pendek yang disebut *message digest*. Untuk menghasilkan nilai *message digest* dari pesan besar, fungsi hash menerima masukan berupa isi blok pesan saat ini serta nilai hash dari blok pesan sebelumnya,

$$h_i = H(M_i, h_{i-1})$$

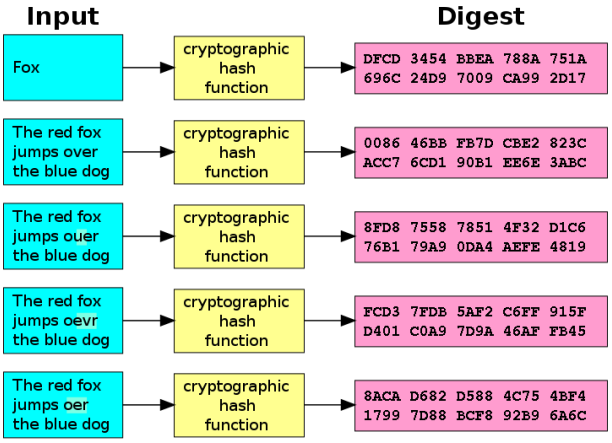
Fungsi hash adalah publik (tidak dirahasiakan) dan keamanannya terletak pada sifat satu arahnya itu. Ada beberapa fungsi hash satu arah yang telah diciptakan, antara lain MD2, MD4, MD5, Secure Hash Function (SHA), Snefru, N-hash, RIPE-MD, dan lain lain.



Gambar 1 Fungsi hash satu arah

Berikut ini beberapa aplikasi fungsi hash satu arah:

1. Menjaga integritas data. Fungsi hash sangat peka terhadap perubahan 1 bit pada pesan. Jika pesan berubah 1 bit, nilai hash berubah sangat signifikan. Sifat inilah yang digunakan untuk menjaga integritas data yaitu dengan cara membandingkan nilai hash baru dengan nilai hash lama. Jika sama artinya pesan masih asli, sebaliknya jika tidak sama artinya pesan sudah dimodifikasi.
2. Menghemat waktu pengiriman. Misalnya untuk memverifikasi sebuah salinan arsip dengan arsip asli dimana salinan dokumen berada di tempat yang jauh dari basis data arsip asli. Daripada mengirim salinan arsip tersebut secara keseluruhan ke komputer pusat (yang membutuhkan waktu transmisi lama), lebih efektif mengirimkan *message digest*-nya. Jika *message digest* salinan arsip sama dengan *message digest* arsip asli, berarti salinan arsip tersebut sama dengan arsip master.
3. Menormalkan panjang data yang beraneka ragam. Misalkan *password* panjangnya bebas (minimal 8 karakter). *Password* disimpan di dalam basis data di komputer *host* (*server*) untuk keperluan otentikasi pemakai komputer. Untuk menyeragamkan panjang *field password* di dalam basis data, *password* disimpan dalam bentuk nilai hash (panjang nilai hash tetap).



Gambar 2 Fungsi hash sangat peka terhadap perubahan 1 bit pada pesan

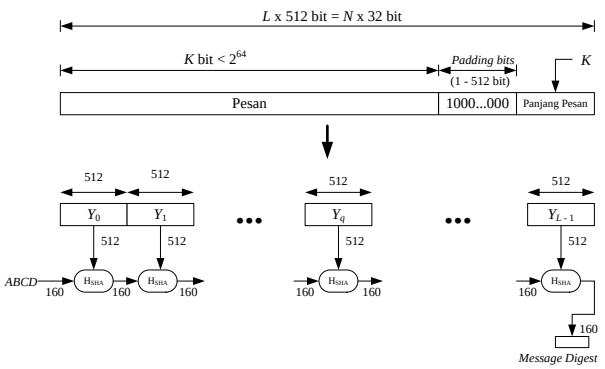
B. Algoritma SHA-1

Dalam kriptografi, SHA-1 adalah fungsi hash kriptografi yang didesain oleh NSA (National Security Agency) dan diumumkan oleh NIST sebagai U.S. Federal Information Processing Standard. SHA artinya adalah Secure Hash Algorithm. SHA hingga saat ini ada 3 macam yaitu SHA-0, SHA-1, dan SHA-2. Struktur SHA-1 dan SHA-0 mirip satu sama lain. Hal ini dikarenakan SHA-1 merupakan perbaikan dari SHA-0.

SHA-1 akan menerima masukan berupa pesan dengan ukuran maksimum 264 bit dan menghasilkan *message digest* yang mempunyai panjang sebesar 160 bit. Langkah-langkah yang dilakukan dalam pembuatan *message digest* dengan menggunakan SHA-1 adalah sebagai berikut.

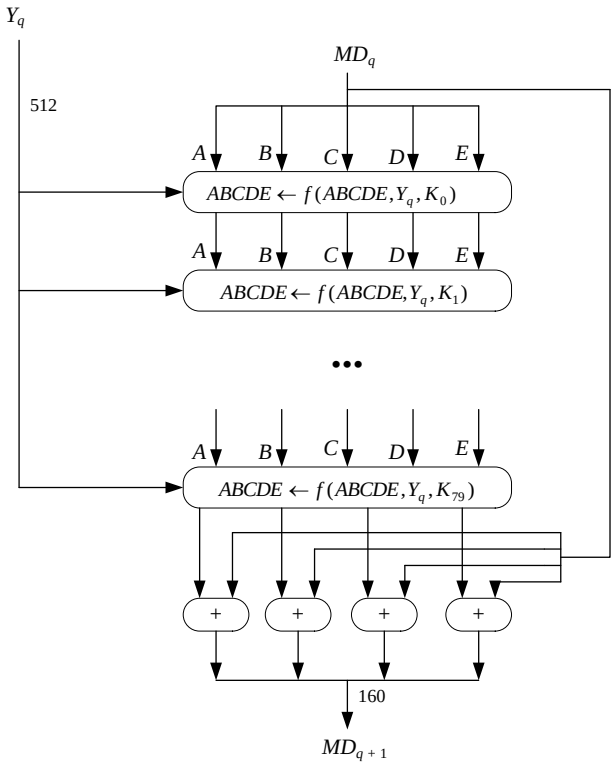
a. Penambahan bit pengganjal (*padding bits*). Pesan yang masuk akan ditambah sejumlah bit sehingga panjang pesan dapat menjadi kongruen dengan 448 modulo 512. Meskipun pesan yang dimasukkan sudah memiliki panjang 448 bit, maka akan tetap ditambah bit-bit sepanjang 512.
b. Penambahan nilai panjang pesan semula. Pesan yang telah diberikan bit-bit tambahan ditambah lagi dengan 64 bit yang menyatakan panjang pesan awal sehingga sekarang panjang pesan yang dipunyai merupakan kelipatan 512.
c. Inisialisasi penyangga (*buffer*) MD. SHA-1 membutuhkan 5 buah penyangga yang masing-masing panjangnya 32 bit. Sehingga total panjang penyangga adalah 160 bit. Kelima penyangga ini diberi nama A, B, C, D, dan E. Masing-masing penyangga tersebut diinisialisasi dengan nilai-nilai sebagai berikut.
   A = 67452301
   B = EFCDAB89
   C = 98BADCFE
   D = 10325476
   E = C3D2E1F0
d. Pengolahan pesan dalam blok berukuran 512 bit.

Berikut ini adalah skema pembuatan *message digest* dengan meggunakan fungsi SHA-1.



Gambar 3 Skema pembuatan *message digest* dengan menggunakan SHA-1

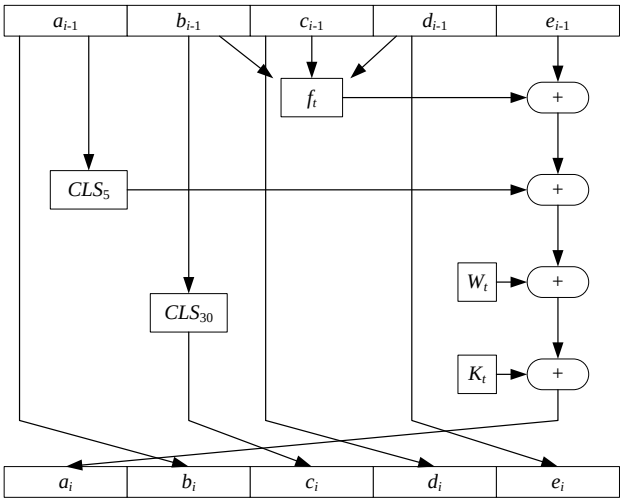
Dalam SHA-1 pesan dibagi menjadi blok-blok kecil berukuran 512 bit untuk kemudian diproses dalam fungsi SHA-1 ini. Berikut ini adalah skema dari pengolahan blok 512 bit di dalam proses SHA-1 yang dinamakan proses H_{SHA}.



Gambar 4 Skema pengolahan blok 512 bit dalam fungsi SHA-1, proses H_{SHA}

Proses H_{SHA} terdiri dari 80 kali putaran dan masing-masing putaran menggunakan bilangan penambah K_t. Nilai Kt ditentukan sebagai berikut.
Putaran $0 \le t \le 19$ K_t = 5A827999
Putaran $20 \le t \le 39$ K_t = 6ED9EBA1
Putaran $40 \le t \le 59$ K_t = 8F1BBCDC
Putaran $60 \le t \le 79$ K_t = CA62C1D6

Berikut ini adalah skema detail operasi dasar yang terjadi pada setiap putaran proses H_{SHA}.



Gambar 5 Skema operasi dasar pada setiap putaran proses H_{SHA}

Fungsi f_t pada operasi di atas merupakan fungsi logika yang mengikuti aturan sebagai berikut untu setiap putaran prosesnya.

Putaran	$f_t(b, c, d)$
0 .. 19	$(b \wedge c) \vee (\sim b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

Lalu nilai W_1 sampai W_{16} berasal dari 16 word pada blok yang sedang diproses, sedangkan nilai Wt berikutnya didapat dari persamaan berikut.

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$$

III. SERANGAN KOLISI PADA SHA-1

Untuk fungsi hash dengan L adalah jumlah bit dalam *message digest*, penemuan pesan yang sesuai dengan *message digest* yang diberikan selalu dapat dilakukan dengan menggunakan pencarian *brute force* dengan kompleksitas operasi sebesar 2^L. Serangan ini disebut *preimage attack* dan mungkin atau tidak mungkin dilakukannya bergantung pada besarnya L dan lingkungan komputasi tertentu. Kriteria kedua adalah menemukan dua pesan yang berbeda yang menghasilkan *message digest* yang sama. Serangan ini yang dikenal sebagai kolisi, membutuhkan rata-rata hanya $2^{L/2}$ operasi. Kekuatan fungsi hash biasanya dibandingkan dengan cipher simetris setengah panjang *message digest*. Jadi SHA-1 pada awalnya dianggap memiliki kekuatan 80-bit.

Pada awal 2005, Rijmen dan Oswald mempublikasikan serangan pada versi reduksi dari SHA-1—53 dari 80 putaran—yang menemukan kolisi dengan komputasi kurang dari 2^{80} operasi.

Pada bulan Februari 2005, serangan oleh Xiaoyun Wang, Yiqun Lisa Yin, dan Hongbo Yu dipublikasikan. Serangan dapat menemukan kolisi dalam versi lengkap dari SHA-1, membutuhkan kurang dari 2^{69} operasi. Mereka menulis: "Secara khusus, analisis kami dibangun berdasarkan *original differential attack* pada SHA-0, *near collision attack* pada SHA-0, teknik *multiblock collision*, serta teknik modifikasi pesan yang digunakan dalam serangan pencarian kolisi pada MD5. Menemukan kolisi SHA-1 tidak akan mungkin dilakukan tanpa teknik analisis yang kuat tersebut." Mereka telah melaporkan kolisi untuk 58-putaran SHA-1, ditemukan dengan 2^{33} operasi hash. *Paper* dengan deskripsi serangan tersebut sudah dipublikasikan pada Agustus 2005 di konferensi CRYPTO. Dalam sebuah wawancara, Yin menyatakan "Kurang lebih, kita menggunakan dua kelemahan berikut: Yang pertama adalah bahwa langkah *file preprocessing* tidak cukup rumit, yang kedua adalah bahwa operasi matematika dalam 20 putaran pertama mengalami masalah keamanan yang tak terduga"

Pada tanggal 17 Agustus 2005, perbaikan pada SHA-1 serangan diumumkan atas nama Xiaoyun Wang, Andrew Yao dan Frances Yao pada sesi rump CRYPTO 2005, menurunkan kompleksitas yang dibutuhkan untuk menemukan kolisi pada SHA-1 menjadi 2^{63}. Pada tanggal 18 Desember 2007, rincian hasil ini dijelaskan dan diverifikasi oleh Martin Cochran.

Christophe De Cannière dan Christian Rechberger meningkatkan serangan terhadap SHA-1 lebih jauh lagi dalam *paper* "*Finding SHA-1 Characteristics: General Results and Applications*," menerima Best Paper Award di ASIACRYPT 2006. Sebuah kolisi dua blok untuk 64-putaran SHA-1 dilaporkan, ditemukan menggunakan *unoptimized method* dengan 2^{35} operasi fungsi kompresi. Oleh karena serangan ini membutuhkan setara dengan sekitar 2^{35} operasi, hal ini dianggap serangan teoritis yang signifikan. Serangan mereka kemudian diperpanjang sampai 73-putaran pada tahun 2010 oleh Grechnikov. Dalam rangka untuk mencari kolisi yang sebenarnya pada versi penuh 80-putaran dari fungsi hash, dibutuhkan waktu komputasi yang lama. Untuk itu, pencarian kolisi untuk SHA-1 menggunakan platform komputasi terdistribusi BOINC mulai 8 Agustus 2007, yang diselenggarakan oleh Graz University of Technology. Usaha ini dihentikan pada 12 Mei 2009 karena kurangnya kemajuan.

Pada Sesi Rump dari CRYPTO 2006, Christian Rechberger dan Christophe De Cannière mengklaim telah menemukan serangan kolisi pada SHA-1 yang akan memungkinkan penyerang untuk memilih paling sedikit bagian dari pesan.

Pada tahun 2008, metodologi serangan oleh Stéphane Manuel melaporkan kolisi hash dengan kompleksitas teoritis diperkirakan 2^{51}-2^{57} operasi. Namun ia kemudian menarik kembali klaim tersebut setelah menemukan bahwa jalur kolisi lokal tersebut tidak benar-benar independen, dan akhirnya mengutip bahwa vektor kolisi yang paling efisien adalah yang sudah diketahui sebelumnya.

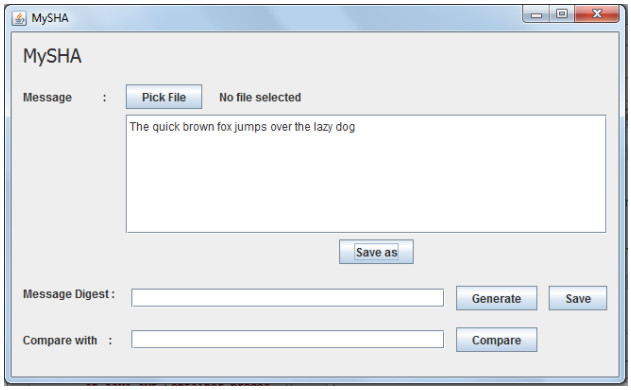
Cameron McDonald, Philip Hawkes dan Josef Pieprzyk melaporkan serangan kolisi hash dengan mengklaim kompleksitas 2^{52} pada sesi Rump dari Eurocrypt 2009. Namun kemudian *paper* mereka "*Differential Path for SHA-1 with complexity* $O(2^{52})$" telah ditarik karena penemuan mereka bahwa perkiraan mereka salah.

Pada 2012, serangan terhadap SHA-1 yang dianggap sebagai satu yang paling efisien dilakukan oleh Marc Stevens dengan perkiraan biaya sebesar $2.77M untuk memecahkan sebuah nilai hash tunggal dengan menyewa kekuatan komputasi CPU dari *cloud server*. Stevens mengembangkan serangan ini sebagai sebuah proyek yang disebut HashClash, menerapkan *differential path*

attack. Pada tanggal 8 November 2010, ia mengklaim bahwa ia menemukan *fully working near-collision attack* melawan versi penuh SHA-1 dengan kompleksitas yang setara sekitar $2^{57,5}$ kompresi SHA-1. Ia memperkirakan serangan ini dapat dikembangkan menjadi kolisi penuh dengan kompleksitas sekitar 2^{61}.

IV. EKSPERIMEN

Pada bab ini, penulis melakukan eksperimen kecil untuk mencoba menemukan kolisi pada SHA-1 dengan menggunakan aplikasi MySHA yang merupakan implementasi fungsi hash SHA-1 (tucil 4).



Gambar 6 Implementasi fungsi hash SHA-1

Eksperimen pertama adalah dengan menyisipkan nilai null di tempat-tempat tertentu dalam pesan, hasilnya sebagai berikut.

Pesan	***Message Digest***
The quick brown fox jumps over the lazy dog	2fd4e1c67a2d28fced849 ee1bb76e7391b93eb12
The quick brown fox jumps over the lazy dog**<NUL>**	af26cb3bc7c9aa489f781 40d295999a29ba5ddf8
<NUL>The quick brown fox jumps over the lazy dog	e36fc5ed0a22a9e33ae03f f44f6d6871edf6011f
<NUL>The quick brown fox jumps over the lazy dog**<NUL>**	049f672fd4b9f416c1245 e4eee05a310e490062b
<NUL>The**<NUL>** quick**<NUL>** brown**<NUL>** fox**<NUL>** jumps**<NUL>** over**<NUL>** the**<NUL>** lazy**<NUL>** dog**<NUL>**	ae284690b0c9ef1b08057 13c1dc730de52b64bca

Eksperimen kedua adalah dengan mencari sinonim setiap kata dari pesan, contohnya sebagai berikut.

The (quick | fast) (brown | chocolate) fox (jumps over | skips) the lazy dog

Kemudian mencacahnya dari setiap kemungkinan pesan yang terbentuk, hasilnya sebagai berikut.

Pesan	***Message Digest***
The quick brown fox jumps over the lazy dog	2fd4e1c67a2d28fced849 ee1bb76e7391b93eb12
The quick brown fox skips the lazy dog	dffb47915d3043c55c9cd 40d4d2e1f564906b0a5
The quick chocolate fox jumps over the lazy dog	6a453b427bd4d16d9d65 833904b65730e1e9e89c
The quick chocolate fox skpis the lazy dog	58e5e7b68e5cc902a4ee0 616ad865b7376a8fe46
The fast brown fox jumps over the lazy dog	2e25fbb1758529de93b99 0a8cf36a493612b1f23
The fast brown fox skips the lazy dog	3176570324b9e959b7f4 928423c4394485e64a51
The fast chocolate fox jumps over the lazy dog	da7b396c1917ca0f35090 eb1144205a2fd922c12
The fast chocolate fox skips the lazy dog	342a0e3be6fe530f886a3 813b6aea0f52f515ae2

V. ANALISIS

A. Analisis Hasil Eksperimen

Dari hasil eksperimen pertama, terlihat bahwa penyisipan nilai null di tempat-tempat tertentu dalam pesan yang sama ternyata menghasilkan *message digest* yang berbeda secara signifikan. Padahal biasanya nilai null merupakan karakter yang tidak terlihat pada *file / document viewer*. Hal ini memungkinkan adanya pesan-pesan yang terlihat sama namun masing-masing memiliki *message digest* yang berbeda.

Sedangkan eksperimen kedua bertujuan untuk menunjukkan bahwa kita dapat mengurangi jumlah komputasi *brute force* dalam pencarian kolisi dengan cara membatasi kemungkinan pesan yang terbentuk. Dalam kasus ini, pesan yang terbentuk dibatasi dengan mengganti setiap kata dalam pesan dengan sinonimnya. Ada 3 kata yang memiliki sinonim dengan masing-masing jumlah sinonimnya ada 2 sehingga jumlah komputasi *brute force* adalah 2x2x2 = 8 operasi.

B. Dampak Adanya Kolisi pada SHA-1

Kolisi adalah hal yang buruk dalam fungsi hash. Kolisi memungkinkan terjadinya serangan *varying trailing nulls*. Contohnya dengan skenario berikut: ada satu pesan yang mengatakan "Beli 1000 saham" dengan berbagai jumlah nilai null di akhir. Kemudian kita buat pesan lain yang mengatakan "Jual 1000 saham" dengan berbagai jumlah nilai null di akhir. Hash sejumlah besar pesan, cari kolisinya. Dapatkan korban untuk menandatangani satu pesan dan sekarang kita memiliki pesan tertandatangani lainnya. Ada banyak serangan seperti ini terhadap berbagai protokol yang menggunakan fungsi hash.

Karena blok dan struktur algoritma berulang serta tidak adanya langkah-langkah akhir tambahan, semua fungsi SHA rentan terhadap serangan kolisi ekstensi-panjang

dan pesan-parsial. Serangan ini memungkinkan seorang penyerang untuk memanipulasi pesan, hanya ditandatangani oleh hash berkunci -SHA(message||key) atau SHA(key||message)- dengan memperpanjang pesan dan menghitung ulang hash tanpa mengetahui kunci.

Selain dampak negatif, terselip dampak positif dari kolisi pada SHA-1 yaitu memacu dilakukannya penelitian lebih jauh untuk mengembangkan algoritma hash yang baru. Dengan kata lain, kita belajar bagaimana merancang algoritma baru dengan mematahkan algoritma lain. Fungsi hash adalah kriptografi primitif yang paling tidak-dipahami dengan baik dan teknik hashing jauh kurang berkembang dibandingkan dengan teknik enkripsi.

C. Solusi untuk Mengurangi Dampak Negatif Adanya Kolisi pada SHA-1

Ukuran message digest adalah tetap dan biasanya pesan itu sendiri harus memiliki properti tertentu lainnya yang akan dianggap sah. Hal ini sangat menghambat penyerang mencari kolisi. Serangan *varying trailing nulls* tidak akan bekerja bagi pesan tersebut. Jadi jika *message digest* sangat fleksibel dan tidak dilakukan pemeriksaan integritas dengan fungsi lain di luar fungsi SHA-1, maka serangan ini lebih mudah dilakukan. Oleh karena itu, kita harus menggunakan lapisan lain yang melakukan pemeriksaan integritas untuk mengurangi kemungkinan serangan.

Untuk mencegah serangan kolisi ekstensi-panjang dan pesan-parsial, paling sederhana adalah dengan melakukan hash dua kali sebagai berikut.

$$SHA_d(message) = SHA(SHA(0^b||message))$$

D. Tingkat Keamanan AlgoritmaSHA-1

Fungsi hash memiliki sifat keamanan tertentu. Tentu saja, sangat mungkin untuk menggunakan fungsi hash yang tidak memiliki sifat keamanan. Anggaplah setiap pengembang perangkat lunak yang kompeten dapat menguji serangan tertentu. Tapi berapa banyak serangan yang mungkin di luar sana? Dapatkah kita berpikir dan mengujinya semua? Dan bagaimana dengan semua pengembang perangkat lunak yang kurang kompeten?

Kolisi membutuhkan komputasi rata-rata sebesar $2^{L/2}$ operasi, oleh karenanya kekuatan fungsi hash biasanya dibandingkan dengan cipher simetris setengah panjang *message digest*. Jadi, SHA-1 pada awalnya dianggap memiliki kekuatan 80-bit (160 bit / 2). Namun seiring ditemukannya kolisi dengan komputasi kurang dari 2^{80} operasi, disimpulkan bahwa SHA-1 telah terpatahkan, walaupun serangan kolisi yang ada hanya sebagai serangan teoritis.

Pada 15 March 2006, NIST mengumumkan bahwa fungsi hash keluarga SHA-2 (yaitu: SHA-224, SHA-256, SHA-384 dan SHA-512) dapat digunakan oleh lembaga pemerintahan untuk semua aplikasi yang menggunakan algoritma *secure hash*. Lembaga-lembaga pemerintahan harus berhenti menggunakan SHA-1 untuk membuat tanda tangan digital, membuat *time stamps* dan untuk aplikasi lain yang membutuhkan *collision resistance*. Lembaga-lembaga pemerintahan dapat menggunakan SHA-1 untuk aplikasi berikut: memverifikasi tanda tangan digital dan *time stamps* yang lama, membuat dan memverifikasi *hash-based message authentication codes* (HMACs), *key derivation functions* (KDFs), dan pembangkit bit/nilai acak.

VI. KESIMPULAN

Sebuah fungsi hash yang baik harus bersikap seperti permutasi acak. Kita tidak mampu untuk melakukan analisis keamanan di setiap tempat yang menggunakan fungsi hash untuk mengetahui apakah serangan kolisi berbahaya. Fungsi hash sangat berguna, banyak aplikasi dari fungsi hash yang sangat bermanfaat. Lebih mudah dan lebih aman menggunakan fungsi hash yang baik daripada mencoba untuk menggunakan fungsi hash yang sebagian terpatahkan. Sudah ditemukan serangan kolisi secara teoritis pada SHA-1. Oleh karena itu, sudah saatnya kita meninggalkan SHA-1 dan menggantinya dengan fungsi hash yang lain yang lebih aman.

REFERENSI

http://eprint.iacr.org/2008/469.pdf, diakses tanggal 24 April 2013.

http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2007-2008/Makalah2/MakalahIF5054-2007-B-052.pdf, diakses tanggal 24 April 2013.

http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2007-2008/Makalah2/MakalahIF5054-2007-B-021.pdf, diakses tanggal 24 April 2013.

http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html, diakses tanggal 19 Mei 2013.

http://csrc.nist.gov/groups/ST/hash/policy_2006.html, diakses tanggal 19 Mei 2013.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Mei 2013

Hapsari Tilawah 13509027