STUDI TRANSISI KE FUNGSI HASH BARU

Dominikus D Putranto - 13506060

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : dominikus.d.putranto@gmail.com

Abstrak

Fungsi hash 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. String keluaran panjangnya umumnya berukuran jauh lebih kecil daripada ukuran string semula.

Fungsi hash digunakan banyak sekali hampir di seluruh protokol kriptografi. Oleh karena itu keberadaannya mutlak diperlukan, contohnya antara lain MD5, SHA-1, SHA-256, dan SHA-512,. Dan yang paling sering digunakan adalah MD5 dan SHA-1.

Pada tahun 2004 dan 2005, *attack* untuk MD5 dan SHA-1 dipublikasikan. Sehingga tidak tertutup kemungkinan suatu saat memang perlu untuk merancang fungsi hash yang baru. Namun sampai sekarang belum ada pondasi teori untuk mengkonstruksi sebuah fungsi hash.

Makalah ini akan membahas mengenai fungsi-fungsi hash yang baru, dan bagaimana cara dan strategi untuk transisi dari penggunaan fungsi hash yang lama ke penggunaan fungsi hash yang baru. Protokol yang akan dibahas di makalah ini adalah S/MIME.

***Kata kunci**: Hash, Kritptografi, MD5, SHA-1, S/MIME*

1. Pendahuluan

Saat ini hampir semua protokol kriptografi bergantung pada keamanan dari fungsi hash. Walaupun berbagai macam fungsi hash telah tersedia. Seperti MD5 dan SHA-1 yang telah digunakan secara luas.

Namun, keduanya berasal dari fungsi hash MD4, yang dikenal lemah keamanannya, yang kemudian mengarah kepada pemikiran bahwa MD5 dan SHA-1 pun juga lemah. Dan hal ini akhirnya diperkuat dengan ditemukannya *collision* pada MD5 pada tahun 2004, dan ditemukan juga *collision* pada SHA-1 pada tahun 2005. Di mana Wang dapat menemukan *collision* untuk SHA-1 hanya dengan 269 operasi.

Sehingga jelas bahwa MD5 maupun SHA-1 sangat lemah tingkat keamanannya, sehingga perlu dipikirkan.

Oleh karena itu pula, jelas bahwa transisi ke fungsi hash yang baru mutlak diperlukan. Dan walaupun ini bukan kebutuhan yang langsung, namun hal ini tidak bisa ditunda-tunda.

2. Latar Belakang

2.1 Penggunaan Dari Fungsi Hash

Fungsi hash digunakan untuk berbagai tujuan. Di bagian ini akan dibahas mengenai penggunaannya secara umum.

2.1.1 Tanda tangan digital

Tujuan dari fungsi hash kriptografis pertama kali memang sebagai preparasi untuk masukan untuk tanda tangan digital. Algoritma semacam RSA terlalu mahal untuk diaplikasikan secara langsung untuk setiap blok masukan untuk sebagian besar pesan. Oleh karena itu, masukan secara aman dikompresi menggunakan fungsi hash.

2.1.2 *Message Authentication Codes (MAC)*

Fungsi hash juga digunakan untuk autentifikasi pesan secara cepat antara 2 pihak yang men-*share* sebuah pesan rahasia.

Yang paling umum digunakan adalah :

$$H (K \oplus \text{opad}, H (K \oplus \text{ipad}, M))$$

Di mana H adalah sebuah fungsi hash, K adalah rahasia yang di-share dan M adalah pesan yang akan diautentikasi. Di sini kemunculan H maupun K yang berulang dapat membantu mencegah berbagai serangan, di mana musuh tidak bisa tahu dan tidak bisa mengkontrol masukan di luar hash tersebut.

2.1.3 Fungsi Pseudo-random

Fungsi hash dapat digunakan untuk membuat fungsi pseudo-random. Di mana adalah sebuah mekanisme deterministik untuk membangkitkan sebuah stream output dari sebuah stream input sehingga tampak acak.

2.1.4 *Data Fingerprinting*

Fungsi hash sering digunakan untuk memproduksi sidik jari untuk berbagai berkas maupun pesan. Kadang, nilai tersebut disimpan terpisah dengan data, sehingga data yang yang telah diubah dapat dideteksi.

2.2 Serangan Terhadap Fungsi Hash

Secara teori, fungsi hash didesain untuk mempunyai 3 kriteria seperti di bawah ini :

Collision Resistance Tidak mungkin dapat dilakukan komputasi untuk menemukan $x, y, x \neq y$ sehingga $H(x) = H(y)$

Preimage Resistance Diberikan sebuah nilai keluaran *y*, tidak mungkin dapat dilakukan komputasi sehingga ditemukan *x* di mana $H(x) = y$

Second Preimage Resistance Diberikan sebuah niai *x*', tidak mungkin dapat dilakukan komputasi sehingga ditemukan *x* di mana $H(x) = H(y)$

Dilihat dari kriteria-kriteria tersebut di atas, SHA-1 sebenarnya lebih lemah dari serangan dibanding kelihatannya.

Walaupun dibanding 2 properti lainnya collision-resistance bukan sesuatu yang bisa dianggap kegagalan, namun collision-resistance adalah sebuah isu yang serius juga. Di mana ada 2 orang bernama Lucks dan Daum yang telah dapat membangkitkan sebuah berkas Postscript yang merupakan serangan. Mereka mengambil keuntungan dari properti fungsi hash :

$$H(x) = H(y) \rightarrow H(x||\Sigma) = H(y||\Sigma)$$

Di mana Σ adalah sebuah string sembarang, dan *x* dan *y* mempunyai panjang yang sama.

Pertama, mereka membangkitkan dua buah Postcript yang mengandung sebuah kolisi yang secara sintaks tetap. Untuk setiap berkasnya, mereka menambahkan sebuah program Postcript yang mengecek nilai dari variabel tersebut dan menampilkan satu dari dua huruf. Pihak penyeran dapat meyakinkan seseorang untuk menandatangani secara digital surat yang pertama, yang tidak berbahaya. Dan tanda tangan tersebut yang sama akan cocok dengan surat yang kedua, di mana surat yang kedua tersebut adalah sebuat surat yang berbahaya.

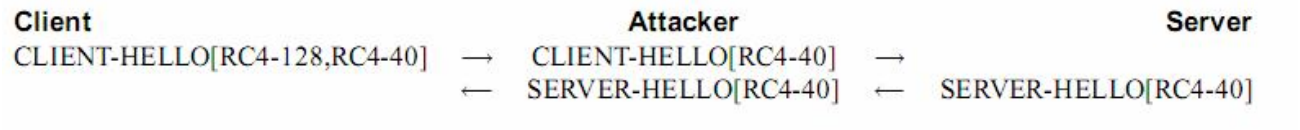
3. Masalah Transisi Hash

Meskipun details dari strategi transisi untuk berbagai protokol mungkin berbeda-beda, namun banyak terdapat elemen yang sama. Di bagian ini, kita akan membahas hal-hal yang meliputi masalah yang berhubungan dengan masalah transisi hash dan tujuan dari desain strategi transisi hash.

Masalah transisi hash adalah sebuah kasus khusus dari masalah transisi protokol secara umum. Di mana versi baru dari sebuah protokol akan menggantikan versi yang lama. Di mana harus dipikirkan cara mengenai transisi yang halus dari yang lama menjadi yang baru dengan level kesalahan yang minimum. Dalam transisi protokol secara umum, ada 3 tipe agen :

Old Agen yang hanya berbicara pada versi yang lebih lama.

Switch-hitting Agen yang akan berbicara pada kedua versi



New Agen yang hanya akan berbicara pada versi baru

Pada saat permulaan dari transisi, semua agen adalah Old. Saat transisi selesai, semua agen adalah New. Tujuan dari strategi transisi adalah untuk melakukan transisi antara kedua state tersebut dengan tingkat kerusakan yang minimum.

3.1 Backward Compatibility

Kebutuhan paling dasar yang utama dari transisi adalah *backward compatibility*. Agen Old dan Switch-hitting harus dapat berkomunikasi, menggunakan versi yang lama. Tanpa backward-compatibility, pengguna harus melakukan banyak hal untuk melakukan upgrade. Pendekatan yang umum adalah dengan menimplementasikan Switch-hitting sampai sebagian besar implementasi adalah Switch-hitting. Sekali hal tersebut terlaksana, implementasi dari New dapat dilakukan dengan aman.

Untuk protokol yang interaktif seperti SSL/TLS ataupun IKE, backward compatibility tercapai dengan mempunyai peer Switch-hitting yang mengenali bahwa dia sedang berkomunikasi dengan peer yang Old. Sedangkan untuk protokol yang tidak interaktif seperti S/MIME, implementasi dari Switch-hitting harus mengirimkankan pesan yang dapat dibaca oleh Old kecuali jika mereka tahu bahwa sebuah peer adalah Switch-hitting atau New.

3.2 Newest Common Version

Ketika dua buah Switch-hittin client berkomunikasi, mereka dapat menggunakan protokol yang lama maupun protokol yang baru. Karena tujuan utamanya adalah untuk mendeploy versi yang baru, maka mereka akan menggunakan versi yang mungkin. Dengan protokol yang interaktif, hal ini mudah, peer mendeteksi mereka berkomunikasi dengan versi yang baru dan menggunakannya. Namun, dengan protokol yang tidak interaktif, hal tersebut menjadi lebih sulit. Pendekatan yang umum adalah untuk implementasi Switch-hitting yang memulai berbicara kepada versi yang lama namun dapat mendukung penggunaan versi yang baru. Ini mengijinkan penerima untuk men-cache bahwa peer tersebut adalah Switch-hitting dan menggunakan versi yang baru. Penerima pesan versi yang baru dapat mengasumsikan bahwa pengirim dapat membaca versi yang baru.

3.3 Downgrade Protection

Kebutuhan tambahan untuk protokol yang aman adalah dapat bertahan untuk penurunan versi / algoritma. Berdasarkan situasi di mana 2 peer masing-masing mensupport dua algoritma kriptografi, satu yang kuat dan satu yang lemah. Jika seseorang yang menyerang memaksa peer untuk memakai algoritma yang lebih lemah, mungkin dia akan bisa menyerang sistem komunikasi tersebut. Di mana adalah mungkin untuk memaksa kedua implementasi tersebut menggunakn algoritma yang lemah, padahal tersedia algoritma yang lebih kuat.

Serangan dapat dilihat di gambar 1. Penyerang mengintersepsi pesan CLIENT-HELLO milik klien dan mencegah sistem menggunakan algoritma yang kuat. Server akan berpiki bahwa client hanya menyediakan algoritma yang lemah dan oleh karena itu sistem menjadi terbuka terhadap serangan.

SSLv3/TLS dan IKE sudah memasukkan pertahanan terhadap downgrade, di mana strategi yang dilakukan adalah bertukar MAC untuk keseluruhan handshake yang dilakukan. Jika MAC terlah terverifikasi, maka peer dapat jaminan jika handshake bersih.

3.4 Credentials versus Implementations

Di sistem yang pemakaiannya berdasarkan public-key secara umum, public key milik peer diautentikasi menggunakan *certificate*. *Certificate* adalah sebuah mandat yang tidak terikat di sebuah revisi spesifik dari sebuah protokol keamanan. Peer harus bisa berkomunikasi dengan agen yang mempunyai kombinasi antara mandat yang baru dan yang lama yang bervariasi.

4. S/MIME

Protokol pertama yang akan dibahas adalah S/MIME (Secure / Multipurpose Internet Mail Extensions), yang adalah merupakan standar untuk enkripsi *public key* dan *signing* untuk

sebuah email yang dienkapsulasi dalam MIME. S/MIME adalah sebuah protokol yang menggunakan prinsip *store-and-forward*.

	Receiver				
Sender	Old	Switch/Old	Switch/Both	Switch/New	New
Old	Old	Old	Old	-	
S/O	Old	Old	Old	Old	-
S/B	Old	Either	Either	Send New	New
S/N	-	New	New	New	New
New	-	New	New	New	New

Tabel 1 Tabel Interoperability untuk S/MIME

Di dalam modus yang umum, S/MIME menggunakan kriptografi kunci publik (menggunakan RSA dan DH) dan penandatanganan digital (RSA dan DSA) untuk autentikasi pesan. Kunci publik milik pengguna akan diautentikasi menggunakan PKIX *ceritificates*.

Ada 5 tipe dari S/MIME klien yang utama :

1. *Old Clients*
2. *Switch-hitting clients* dengan *old certificates* saja
3. *Switch-hitting clients* dengan kedua tipe *certificates* saja
4. *Switch-hitting clients* dengan *new certificates*
5. *New clients* dengan *new certificates*

Tabel di atas mengasumsikan informasi tentang kemampuan dari penerim adalah benar dan sempurna, yang sebenarnya di kasus nyata tidak selalu begitu. Sekarang kita mempertimbangkan bagaimana cara untuk mencapai ke-*interoperability*-an dalam prakteknya, di mana kita mencoba untuk mengira-ngira kemampuan penerima dan membuat pesan yang dapat didekodekan dengan relatif mudah.

Implementor dari S/MIME harus dapat menangani tandatangan yang jamak dengan baik dan benar. Program harus dapat melaporkan jika telah dapat memproses dengan sebuah tandatangan dengan sukses ketika saat itu juga sedang memproses tandatangan yang belum diverifikasi.

4.1 Pesan Pertama

Untuk kasus pertama, dipertimbangkan di mana user mengirim sebuah pesan kepada seseorang yang belum pernah melakukan komunikasi sebelumnya, di sini terdapat 2 kemungkinan kasus :

1. Pengirim tidak mempunyai *certificate* si penerima
2. Pengirim telah memilikinya

4.1.1 Mengirim Tanpa *Certificate* Penerima

Jika pengirim tidak dapat mengakses certificate dari penerima, dia akan menemui dua buah batasan, pertama, dia tidak bisa melakukan enkripsi karena tidak mempunyai kunci publik untuk mengenkripsi. Kedua, dia tidak bisa mengetahui mengenai kemampuan dari penerima. Untuk kasus-kasus tertentu, dia tidak bisa mengasumsikan bahwa perangkat lunak penerima bisa memproses fungsi hash yang baru. Oleh karena itu dia harus menentukan pilihan untuk *certificate* dan fungsi hash yang digunakan.

4.1.2 Mengirim dengan Mengetahui *Certificate* Penerima

Jika pengirim telah mengetahui dan memiliki *certificate* yang dimiliki penerima, maka pengiriman bisa menjadi lebih sederhana. Jika penerima hanya mempunyai sebuah *certificate* pengirim seharusnya menggunakan certificate dengan algoritma yang bersesuaian. Sedangkan jika penerima memiliki beberapa certificate, pengirim harus menggunakan algoritma yang paling kuat. Jadi untuk kasus ini tidak direkomendasikan untuk mengirimkan banyak certificate sekaligus.

4.2 Pesan Selanjutnya

Jika implementasi yang menggunakan S/MIME telah menerima pesan tanda dari peer, maka perkiraan terhadap kemampuan dari penerima. Sebagai contoh misal A telah menerima pesan dari B. Kemungkinan besar B dapat mengetahui algoritma yang telah digunakan untuk memverifikasi tandatangan miliknya. Jika algoritma yang digunakan adalah algoritma yang baru (algoritma yang kuat) maka A hanya perlu untuk menyesuaikan diri untuk menggunakan algoritma tersebut. Namun, jika B menggunakan algoritma yang lama (yang lebih lemah), maka A setidaknya tahu bahwa dia dapat berkomunikasi dengan B menggunakan algoritma tersebut.

Namun, ada juga kemungkinan bahwa B mempunyai implementasi dengan tipe Switch-

hitting. S/MIME mempunyai cara sehingga B dapat memberi tahu A mengenai hal itu dengan menggunakan SMIME Capabilities Signature Attribute yang juga menyertakan daftar dari algoritma-algoritma yang dipakai si B. B dapat mengirim pesan menggunakan SHA-1 namun juga menyertakan pesan bahwa dia juga mendukung penggunaan SHA-512. Di saat Switch-hitting adalah tipe yang diimplementasikan, maka direkomendasikan untuk menyertakan indikasi bahwa dukungan terhadap algoritma yang lebih kuat juga ada.

4.3 Serangan

Di bagian ini, akan dibahas mengenai masalah dalam melindungi implementasi dari Switch-hitting selama masa transisi. Di sini akan dibahas 3 skenario dasar :

1. Penyerang tidak mempunyai certificate yang valid dan kunci privat untuk kedua peer.
2. Penyerang telah memperoleh certificate yang valid (namun masih salah) dan telah mengetahui kunci privat.
3. Penyerang merupakan salah satu *communicating parties*

4.3.1 Serangan Tanpa *Certificate* yang Valid

Jika penyerang tidak mempunyai kunci privat utuk certificate yang valid, maka kemampuannya untuk melakukan penyerangan sangat sedikit, kecuali dia dapat melakukan komputasi untuk mendapatkan pre-images. Karena jika penyerang telah dapat melakukan komputasi untuk mendapatkan pre-images, maka dia dapat melakukan perubahan pesan tanpa terdeteksi. Di kasus ini pertahanan yang bisa dilakukan hanyalah berhenti menggunakan algoritma yang terkait.

4.3.2 Serangan Dengan *Certificate* yang Valid

Jika penyerang telah mempunyai kunci privat, maka dia dapat menirukan peer tersebut. Dan itu akan membuat penyerang akan dapat melakukan pengiriman pesan yang akan tampak dikirim dari peer tersebut. Dan penyerang akan bisa meyakinkan peer yang lain untuk memakai certificatenya yang palsu.

4.3.3 Penyerang adalah Salah Satu dari *Communicating Parties*

Jika mencari kolisi pada hash maka menjadi salah satu dari *communicating parties* adalah sebuah keuntungan yang besar bagi pihak penyerang.

Serangan ini sangat susah untuk ditahan. Serangan ini sangat kuat dan sangat mudah dilakukan bila fungsi hash yang dipakai adalah MD5. Dan karena pertahanan untuk serangan ini susah dilakukan, maka lebih baik dan sederhana bila algoritma yang digunakan adalah algoritma yang kuat.

5. Kesimpulan

Transisi dari penggunaan fungsi hash yang lama dan fungsi hash yang baru tidak mungkin dapat terelakkan lagi, suatu saat hal itu akan terjadi juga jika fungsi hash baru yang lebih baik dan lebih *reliable* dari fungsi hash yang sekarang dipakai telah ditemukan.

Dalam masa transisi tersebut harus dipikirkan strategi dan protokol sehingga sistem yang telah ada yang mengalami perubahan fungsi hash tersebut dapat melakukan perubahan penggunaan fungsi hash dengan benar dan tidak mengalami kesalahan dalam prosesnya. Salah satu contoh protokol yang dapat digunakan adalah S/MIME.

6. Daftar Pustaka

[1] Cryptographic hash function, http://en.wikipedia.org/wiki/Cryptographic_hash_function

[2] Munir, Rinaldi, (2006). Diktat Kuliah IF5054 Kriptografi, Departemen Teknik Informatika Institut Teknologi Bandung,

[3] SmoothPerf: A Perfect Hash Function Based on Smoothing Techniques, http://www.gongcaichun.info/PPT/22.pdf

[4] Deploying A New Hash Function, http://www.rtfm.com/ndss.pdf