Penerapan Fungsi Hash Untuk Bagian – Bagian Data

Khaidzir Muhammad Shahih 13512068
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13512068@std.stei.itb.ac.id

***Absrak*—Penggunaan fungsi hash sangat berguna dalam bidang kriptografi. Fungsi hash seperti MD5 dapat digunakan untuk otentikasi data. Penerapan fungsi hash selama ini banyak dilakukan untuk satu bagian data yang utuh. Dalam makalah ini, dipaparkan bagaimana data dibagi – bagi terlebih dahulu, kemudian fungsi hash diberlakukan untuk setiap bagian data.**

***Kata kunci* – Fungsi hash, fragmen data, tandatangan digital, kolisi**

I. PENDAHULUAN

Fungsi hash banyak digunakan dalam berbagai bidang. Kriptografi merupakan salah satu bidang yang bergantung terhadap fungsi hash. Fungsi hash dalam kriptografi dilakukan untuk berbagai penggunaan yang menyangkut keamanan. Beberapa penggunaannya antara lain untuk verifikasi data, tandatangan digital, pengecekan file, dan lain – lain.

Pada sebagian besar penggunaan, fungsi hash diterapkan pada sebuah data yang utuh. Namun ternyata dalam beberapa kasus penggunaan fungsi hash untuk data yang utuh tidak optimal, seperti pada transmisi fragmen – fragmen data[4]. Pada transmisi data kebanyakan, file diberi tandatangan digital kemudian dikirimkan melalui jaringan. Dalam perjalanannya, kemudian file tersebut dibagi – bagi menjadi beberapa fragmen agar bandwidth yang digunakan mencukupi dalam proses transmisi. Akibatnya, proses verifikasi harus menunggu semua fragmen selesai diterima, digabungkan, kemudian diverifikasi. Jika terjadi kerusakan data, tidak dapat diketahui data bagian mana yang rusak, sehingga harus pengirim harus menstransmisikan kembali seluruh bagian data. Sehingga ada pendekatan proses otentikasi dilakukan pada fragmen – fragmen data.

Penggunaan fungsi hash untuk bagian – bagian data juga dapat diterapkan untuk memperkecil kemungkinan kolisi, karena sekarang ini telah ditemukan beberapa kasus kolisi yang terjadi pada beberapa algoritma fungsi hash.

II. TEORI DASAR

A. Fungsi Hash

Fungsi hash adalah fungsi yang menerima masukan sebuah string yang panjangnya sembarang dan menghasilkan sebuah string lain yang panjangnya tetap untuk berapapun panjang string masukannya. Hasil keluaran fungsi hash selanjutnya kita sebut *message digest*. Fungsi hash bersifat satu arah karena kita tidak bisa mengembalikan *message digest* ke string awal.

Fungsi hash dalam kriptografi dapat digunakan untuk menjaga keaslian data dengan cara menghitung nilai hash sebuah data. Jika data berubah, maka nilai hashnya juga akan berubah. Perubahan sedikit saja dalam data dapat mengakibatkan nilai hash yang berubah drastis.

Dalam kriptografi, dikenal beberapa fungsi hash antara lain MD2, MD4, MD5, SHA-0, SHA-1, RIPEMD, WHIRLPOOL, dan lain – lain.

Salah satu sifat yang ingin dicapai dalam sebuah fungsi hash H adalah untuk setiap string masukan x, tidak mungkin ada string y, $x \neq y$ sedemikian sehingga $H(x) = H(y)$. Namun, dalam kenyataannya pada beberapa algoritma fungsi hash terdapat kolisi, yaitu keadaan dimana dua buah string sembarang memiliki nilai hash yang sama.

B. Kriptografi Kunci Simetri

Kriptografi kunci simetri adalah metode enkripsi/dekripsi menggunakan sebuah kunci yang sama baik untuk enkripsi maupun dekripsi, akibatnya semua pihak yang berhubungan harus mengetahui kunci yang sama. Beberapa algoritma kunci simetri antara lain *Blowfish*, *DES*, *RC4*, *RC5*, *Rijndael*, dan lain – lain. Sampai saat ini telah dikenal dua standar kriptografi simetri, yaitu *DES (Data Encryption Standard)* dan *AES* (*Advanced Encryption Standard*, menggunakan algoritma *Rijndael*) sebagai pengganti *DES*, karena dianggap sudah tidak aman lagi.

C. Kriptografi Kunci Publik

Pertama kali dikenalkan oleh Diffie - Hellman dalam makalah berjudul *"New Directions in Crytography"*. Dalam kriptografi kunci publik digunakan dua buah kunci untuk setiap pelaku. Kunci tersebut adalah kunci privat, kunci yang hanya diketahui oleh pemilik kunci; dan kunci publik, kunci yang semua orang dapat mengetahuinya.

Pengirim pesan mengenkripsi pesannya menggunakan kunci privat miliknya. Pihak yang dikirim pesan mendekripsi menggunakan kunci publik milik pengirim. Dengan algoritma kunci publik ini tidak diperlukan pertukaran kunci rahasia, karena kunci publik untuk dekripsi boleh dipublikasikan secara bebas.

Pembangkitan sepasang kunci didasarkan pada persoalan integer klasik seperti pemfaktoran, logaritma diskrit, dan logaritma diskrit kurva eliptik. Beberapa contoh algoritma kunci publik antara lain *RSA*, *ElGamal*, dan *Elliptic Curve Cryptography* (*ECC)*.

D. Tanda Tangan Digital

Tanda tangan digital merupakan metode untuk otentikasi pada data digital. Tanda tangan digital berbeda bergantung pada isi pesan, berbeda dengan tanda tangan dokumen cetak yang selalu sama.

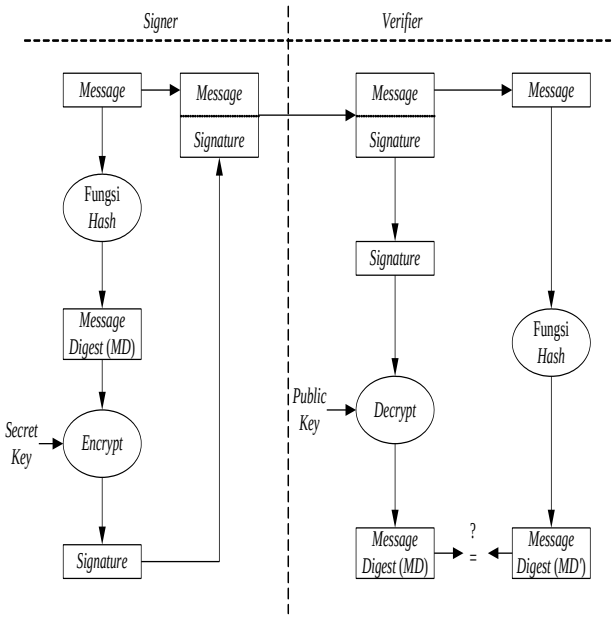
Terdapat dua cara dalam menandatangani pesan digital, yaitu dengan enkripsi pesan dan menggunakan kombinasi antara fungsi hash dan kriptografi kunci publik.

Tanda tangan menggunakan enkripsi dibagi menjadi dua, yaitu menggunakan kriptografi kunci simetri dan menggunakan kriptografi kunci publik. Pada tanda tangan digital menggunakan kunci simetri sudah memberikan solusi untuk otentikasi pesan, karena kunci enkripsi/dekripsi hanya diketahui pihak pengirim dan penerima.

Cara yang kedua untuk tanda tangan digital enkripsi adalah menggunakan kriptografi kunci publik. Dalam metode ini pesan dienkripsi menggunakan kunci private pengirim. Setelah itu, di sisi penerima pesan didekripsi menggunakan kunci publik pengirim. Dengan cara ini, kerahasiaan pesan dan otentikasi tercapai, karena jika pesan berhasil didekrip menjadi pesan bermakna, artinya benar bahwa pesan yang dikirim berasal dari penerima karena hanya kunci publik pengirim yang bersesuaian dengan kunci privat pengirim.

Metode tanda tangan digital yang lainnya adalah kombinasi antara kriptografi kunci publik dengan fungsi hash. Pertama – tama, pesan dihitung nilai hashnya menghasilkan *message digest*. *Message Digest* ini kemudian dienkripsi menggunakan kunci privat pengirim. Hasil enkripsi ini disebut *signature*. Kemudian, *signature* ini "ditempelkan" ke pesan asli untuk selanjutnya dikirimkan ke pihak penerima. Di pihak penerima, pesan yang dikirimkan dipisahkan dengan *signature* penerima. *Signature* tersebut didekripsi menggunakan kunci publik pengirim menghasilkan sebuah *message digest*. Pesan asli yang sudah dipisahkan dengan *signature* dihitung nilai hashnya menggunakan fungsi hash yang sama dengan pengirim. Hasil nilai hash dari pesan asli dan *message digest* hasil dekripsi kemudian dibandingkan. Jika sama, artinya pesan tersebut adalah asli dan orang yang mengirim adalah orang yang sebenarnya (terotentikasi).

Berikut adalah skema dari proses tanda tangan digital menggunakan kombinasi kunci publik dan fungsi hash.



Proses penandatangan digital[1]

III. TANDA TANGAN DIGITAL PADA FRAGMENT DATA SAAT TRANSMISI

Dalam transmisi data, data yang akan dikirimkan biasanya akan dibagi – bagi menjadi beberapa datagram agar bandwidth mencukupi atau sesuai protokol komunikasi. Selanjutnya tidak menutup kemungkinan dari datagram – datagram yang dikirimkan di tengah transmisi dibagi lagi menjadi beberapa fragmen data.

Penerima di tengah ataupun di akhir transmisi harus dapat memverifikasi keaslian dan integritas dari semua fragmen yang diterima. Mekanisme fragmentasi tradisional, seperti menghitung nilai hash dari semua isi pesan dan menandatanganinya tidak optimal ketika pesan diharuskan difragmentasi.[4]

Pada proses pengiriman data, pemberian tandatangan digital diberikan untuk sebuah data secara utuh. Data tersebut kemudian ditransmisikan melalui jaringan komunikasi. Pada proses di tengah transmisi, sangat mungkin dilakukan pembagian data menjadi beberapa fragmen agar bandwidth selama transmisi mencukupi pengiriman data. Di sisi penerima, fragmen – fragmen tersebut kemudian disatukan kembali menjadi data yang utuh. Untuk proses otentikasi, tentu saja karena yang diotentikasi adalah sebuah data yang utuh, penerima harus menunggu semua fragmen terkirim. Setelah fragmen – fragmen data disatukan menjadi satu data yang utuh, maka proses verifikasi dilakukan. Namun, jika terjadi kerusakan pada file selama proses transmisi, tidak dapat diketahui dimana kerusakan file terjadi. Akibatnya, pengirim harus mengirim kembali data yang sebelumnya ditransmisikan. Cara ini tentu akan sangat mengganggu jika ukuran data besar.

Maka untuk menangani kasus ini dengan mengotentikasi semua fragment data. Setiap fragmen data diberi tandatangan digital sendiri – sendiri. Jadi, saat data difragmentasi, sebelum ditransmisikan, dihitung nilai hash untuk setiap fragmen, kemudian nilai hasil hash

yang dihasilkan dienkrip dan ditempelkan pada setiap fragment.

Dengan cara ini, otentikasi untuk setiap fragmen dapat dilakukan tanpa harus menunggu fragmen – fragmen data yang lain. Disini dicobakan cara diatas.

Dalam percobaan ini yang dilakukan hanya membagi data menjadi beberapa fragment dengan ukuran fragment maksimal didefinisikan untuk tiap percobaan. Setiap fragment dihitung nilai hashnya menggunakan algoritma MD5 kemudian dienkripsi menggunakan algoritma kunci publik RSA.

Percobaan pertama dilakukan untuk sebuah file berukuran 3.21 MB dengan ukuran fragment 4 kB dan 8 kB diperoleh hasil sebagai berikut.

Ukuran Fragmen	Jumlah Fragmen	Lama eksekusi (ms)
4 kB	824	2826
8 kB	412	2542

Selanjutnya dilakukan untuk data berukuran sekitar 10 kali data pertama (32.2 MB). Diperoleh hasil sebagai berikut.

Ukuran Fragmen	Jumlah Fragmen	Lama eksekusi (ms)
4 kB	8268	7683
8 kB	4134	5308

Percobaan terakhir diujikan pada data berukuran sekitar 10 kali data yang kedua (301 MB). Diperoleh hasil sebagai berikut.

Ukuran Fragmen	Jumlah Fragmen	Lama eksekusi (ms)
4 kB	77191	48468
8 kB	38596	29601

Hasil ketiga percobaan dapat dilihat dalam tabel berikut.

Ukuran file	Ukuran Fragmen	Jumlah Fragmen	Lama eksekusi (ms)
3.21 MB	4 kB	824	2826
	8 kB	421	2542
32.2 MB	4 kB	8268	7683
	8 kB	4134	5308
301 MB	4 kB	77191	48468
	8 kB	38696	29601

Tabel hasil ujicoba penandatanganan digital untuk setiap fragmen data

Dengan cara tersebut dapat dilakukan otentikasi pada setiap fragmen data pada saat transmisi. Namun, masalah yang ditimbulkan dari penerapan cara ini adalah membesarnya *traffic* pada jaringan dikarenakan pada fragment ditambahkan tandatangan digital yang cukup besar untuk fragment yang kecil.

Masalah utama yang lain adalah untuk menandatangani setiap fragment dibutuhkan kalkulasi yang mahal, sehingga dapat mempengaruhi performa dari transmisi data. Seperti dilihat pada percobaan diatas, waktu total yang dibutuhkan untuk menandatangani setiap fragmen cukup lama dibanding dengan hanya membagi data menjadi fragmen – fragmen saja, tanpa memberi otentikasi pada fragmen. Dari hasil ujicoba, didapatkan hasil sebagai berikut.

Ukuran file	Ukuran Fragmen	Jumlah Fragmen	Lama eksekusi (ms)
3.21 MB	4 kB	824	8
	8 kB	421	7
32.2 MB	4 kB	8268	118
	8 kB	4134	72
301 MB	4 kB	77191	930
	8 kB	38696	853

Tabel hasil ujicoba pembagian file menjadi beberapa fragmen.

Terlihat waktu eksekusi yang dibutuhkan hanya untuk membagi file menjadi beberapa fragmen sangat singkat dibanding dengan penambahan kerja untuk mengotentikasi data setiap fragmen. Selain itu, perlu waktu tambahan di sisi penerima untuk memverifikasi tandatangan digital yang ada dalam setiap fragmen. Pada saat transmisi dibutuhkan m kali proses penghitungan nilai hash dan m kali proses pemberian tanda tangan, dengan m adalah jumlah fragmen. Setelah itu, pada sisi penerima juga dibutuhkan n kali proses verifikasi, dengan n adalah banyaknya fragmen data yang diterima. Cara yang dilakukan ini kuran efisien untuk diterapkan.

IV. Fungsi Hash Pada Bagian Data untuk Mengurangi *Collision*

Pada beberapa fungsi hash, seperti yang telah disebutkan diatas, telah ditemukan adanya *collision*, seperti pada fungsi hash MD5. Pada Maret 2005, Xiaoyun Wang dan Hongbo Yu dari Shandong University di China Mempublikasikan bahwa mereka telah menemukan algoritma untuk mencari 2 buah rangkaian data berbeda yang memiliki nilai fungsi hash MD5 yang sama. Salah satu pasangan rangkaian data tersebut adalah seperti berikut :

```
d131dd02c5e6eec4693d9a0698aff95c
2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a
085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6
dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e
c69821bcb6a8839396f9652b6ff72a70
```

dan

```
d131dd02c5e6eec4693d9a0698aff95c
2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a
085125e8f7cdc99fd91dbd7280373c5b
```

```
d8823e3156348f5bae6dacd436c919c6
dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e
c69821bcb6a8839396f965ab6ff72a70
```

Warna merah menunjukkan perbedaan antara kedua rangkaian byte diatas. Kedua rangkaian byte diatas memiliki nilai MD5 yang sama, yaitu `79054025255fb1a26e4bc422aef54eb4`.

Disini akan dipaparkan sebuah cara agar kedua string diatas memiliki nilai hash yang berbeda. Cara yang dilakukan adalah dengan membagi data menjadi beberapa bagian sebelum dihitung nilai hashnya. Misalkan data dibagi menjadi 2 :

```
d131dd02c5e6eec4693d9a0698aff95c
2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a
085125e8f7cdc99fd91dbdf280373c5b

dan

d8823e3156348f5bae6dacd436c919c6
dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1e
c69821bcb6a8839396f9652b6ff72a70
```

dan

```
d131dd02c5e6eec4693d9a0698aff95c
2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a
085125e8f7cdc99fd91dbd7280373c5b

dan

d8823e3156348f5bae6dacd436c919c6
dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1e
c69821bcb6a8839396f965ab6ff72a70
```

Kemudian setiap bagian data diatas dihitung nilai hashnya menggunakan MD5.

Untuk data yang pertama memiliki nilai hash `{45751c51da75f0633121c9dd5d022dbd, bdf900aff8fc2ce56ccdc1b2e249dae0}`.

Untuk data yang kedua memiliki nilai hash `{882abde1cf075b73e6fdc82ae9cbc229, a56a75c33931d4acabced05b70691540}`.

Dengan demikian, kedua data yang mengakibatkan collision tersebut sekarang memiliki nilai hash yang berbeda.

Dengan menggunakan cara diatas, collision dapat dikurangi, tetapi menghasilkan nilai hash yang lebih besar. Nilai fungsi hash yang dihasilkan berubah menjadi sebuah vector *message digest* berukuran n, dengan n adalah banyaknya bagian data.

Secara kasar, jika terdapat data sebanyak $2^{128} + 1$ buah yang semuanya berbeda, dapat dipastikan terjadi collision jika menggunakan fungsi hash MD5. Dengan menggunakan cara diatas, kepastian terjadinya collision menjadi lebih kecil, dengan $2^{256} + 1$ buah data yang semuanya berbeda. Namun, ukuran keluaran dari fungsi hash n kali lebih banyak daripada tidak dilakukan pembagian data.

V. KESIMPULAN

Fungsi hash tidak hanya dapat dilakukan untuk sebuah data secara utuh. Fungsi hash dapat juga dilakukan pada bagian – bagian data. Penerapan fungsi hash untuk bagian – bagian data dapat berguna untuk mengotentikasi tanda tangan digital fragmen - fragmen data saat transmisi dan untuk mengurangi kemungkinan kolisi pada data.

Namun, untuk penerapan pada otentikasi fragmen data dengan cara yang dipaparkan dalam makalah ini tidak baik dalam sisi performansi. Ini akan menghambat perjalanan transmisi data, sehingga transmisi data yang selalu mengupayakan kecepatan dapat terhambat karena mahalnya operasi dari otentikasi di tengah transmisi data. Oleh karena itu, diperlukan metode lain untuk masalah ini.

Penerapan fungsi hash pada bagian - bagian data untuk mengurangi kolisi memiliki masalah pada ukuran *message digest* yang berubah menjadi n kali, dengan n adalah jumlah bagian data.

VI. UCAPAN TERIMA KASIH

Pertama – tama saya ucapkan terima kasih kepada Tuhan Yang Maha Esa karena atas rahmat-Nya makalah ini bisa terselesaikan. Saya berterima kasih juga kepada kedua orang tua saya yang telah membesarkan saya sampai saat ini. Tidak lupa terima kasih saya ucapkan kepada Pak Rinaldi Munir selaku dosen Mata Kuliah IF4020 Kriptografi yang telah memberi pengetahuan kepada saya tentang semua teori yang ada dalam makalah ini.

REFERENSI

[1] Slide kuliah Rinaldi Munir 2015. Tandatangan Digital. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Tandatangan%20Digital%20(2015).ppt

[2] Slide kuliah Rinaldi Munir 2015. Algoritma Kriptografi Modern. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Algoritma%20Kriptografi%20Modern%20(2015).ppt

[3] Craig Partridge. Authentication for Fragments. conferences.sigcomm.org/hotnets/2005/papers/partridge.pdf

[4] N. Asokan, Kari Kostiainen, Philip Ginzboorg, dan J¨org Ott, Cheng Luo. Towards Securing Disruption-Tolerant Networking. research.nokia.com/sites/default/files/tr/NRC-TR-2007-007.pdf

[5] Slide kuliah Rinaldi Munir 2015. Fungsi Hash. http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Fungsi%20Hash%20(2015).ppt

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, 11 Mei 2015

Khaidzir Muhammad Shahih 13512068