PENGGUNAAN FUNGSI *HASH* SATU-ARAH UNTUK ENKRIPSI DATA

Budi Maryanto

Sekolah Tinggi Manajemen Informatika dan Komputer LIKMI
Jl. Ir. H. Juanda 96 Bandung 40132

E-mail : budimaryanto@likmi.ac.id

Abstrak

Fungsi hash satu arah (*one-way hash function*) adalah salah satu algoritma kriptografi modern yang banyak dimanfaatkan untuk aplikasi pengamanan data seperti integrasi pesan dan otentikasi. Sesuai dengan namanya, fungsi ini bekerja dalam satu arah : pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula. Dua pesan yang berbeda akan selalu menghasilkan nilai *hash* yang berbeda pula. Beberapa fungsi *hash* satu-arah yang sudah dikembangkan, antara lain : *MD2, MD4, MD5, SHA (Secure Hash Function), RIPMEND, WHIRLPOOL.* Dalam tulisan ini juga akan dipaparkan implementasi fungsi *hash* satu-arah menggunakan MD5 pada sebuah contoh riil enkripsi data.

Kata-kata kunci : kriptografi, fungsi *hash* satu-arah, enkripsi, *message-digest*.

1. PENDAHULUAN

Di dalam kriptografi terdapat sebuah fungsi yang sesuai untuk aplikasi keamanan seperti otentikasi dan integrasi pesan. Fungsi tersebut adalah **fungsi hash**, yaitu fungsi yang menerima *string* dengan panjang sembarang dan mengkonversinya menjadi *string* keluaran yang panjangnya tetap (*fixed*).

Fungsi *hash* dapat menerima masukan *string* apa saja. Jika *string* menyatakan pesan (*message*), maka sembarang pesan *M* berukuran bebas dikompresi oleh fungsi *hash H* melalui persamaan : $h = H(M)$.

Keluaran fungsi *hash* disebut juga nilai *hash* (*hash-value*) atau pesan ringkas (*message digest*). Pada persamaan di atas, *h* adalah *message digest* dari fungsi *H* untuk masukan *M*. Dengan kata lain, fungsi *hash* mengkompresi sembarang pesan yang berukuran berapa saja menjadi *message digest* yang selalu tetap.

2. KARAKTERISTIK FUNGSI *HASH* SATU-ARAH

Fungsi *hash* satu-arah adalah fungsi *hash* yang bekerja dalam satu arah : pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula. Dua pesan yang berbeda akan selalu menghasilkan nilai *hash* yang berbeda pula.

Sifat-sifat fungsi *hash* satu-arah selengkapnya adalah sebagai berikut :

1. Fungsi *H* dapat diterapkan pada blok data berukuran berapa saja.
2. *H* menghasilkan nilai *h* dengan panjang tetap (*fixed-length-output*).
3. *H(x)* mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap *h* yang diberikan, tidak mungkin menemukan *x* sedemikian sehingga *H(x) = h*. Itulah sebabnya fungsi *H* dikatakan fungsi *hash* satu-arah (*one-way hash function*).
5. Untuk setiap x yang diberikan, tidak mungkin mencari $y \neq x$ sedemikian sehingga *H(y) = H(x).*
6. Tidak mungkin (secara komputasi) mencari pasangan *x* dan *y* sedemikian sehingga *H(x) = H(y).*

Keenam sifat di atas penting sebab sebuah fungsi *hash* seharusnya berlaku seperti fungsi acak. Sebuah fungsi *hash* dianggap tidak aman jika :

1. Secara komputasi dimungkinkan menemukan pesan yang bersesuaian dengan pesan ringkasnya.
2. Terjadi kolisi (*collision*), yaitu terdapat beberapa pesan berbeda yang mempunyai pesan ringkas yang sama.

Fungsi *hash* bekerja secara iteratif. Masukan fungsi *hash* adalah blok pesan (*M*) dan keluaran dari *hashing* blok pesan sebelumnya : $h_i = H(M_i , h_{i-1})$. Skema fungsi hash satu arah diperlihatkan pada gambar berikut ini :

Gambar 1
Skema Fungsi *Hash* Satu Arah

3 . PENGEMBANGAN ALGORITMA FUNGSI *HASH* SATU-ARAH

Ada beberapa fungsi *hash* satu-arah yang sudah dikembangkan, antara lain: *MD2, MD4, MD5, SHA, RIPMEND, WHIRLPOOL*, dan lain-lain. Pada Tabel 1 dipaparkan resume parameter beberapa fungsi *hash*.

Pada bagian berikutnya akan dibahas secara khusus algoritma *MD5* yang akan penulis gunakan dalam contoh riil implementasi fungsi *hash* satu-arah.

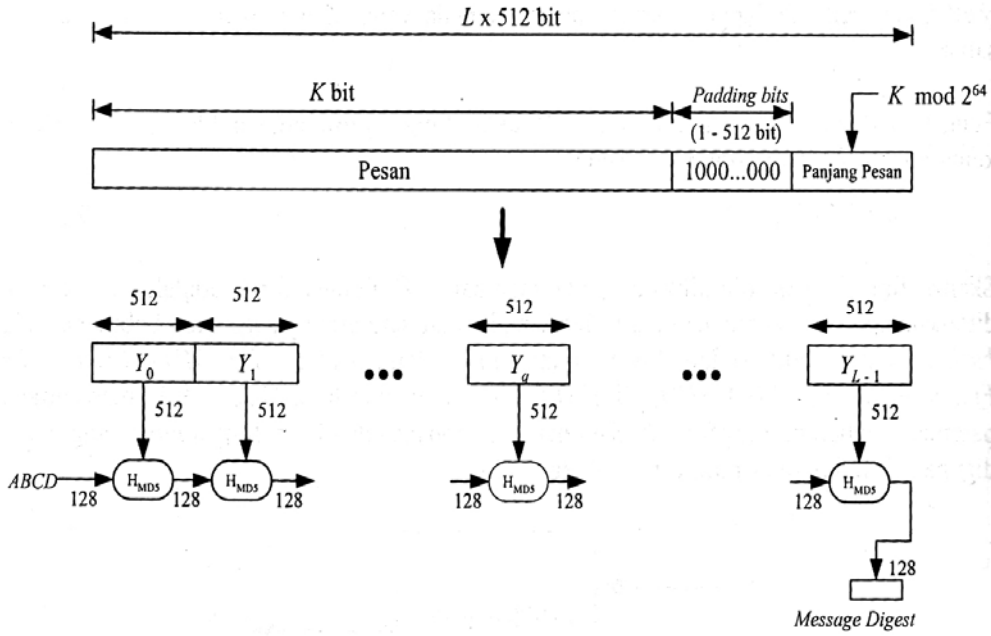
Tabel 1
Beberapa Fungsi *Hash*

Algoritma	Ukuran *Message Digest* (bit)	Ukuran Blok Pesan	Kolisi
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
RIPEMD	128	512	Ya
RIPEMD-128/256	128/256	512	Tidak
RIPEMD-160/320	160/320	512	Tidak
SHA-0	160	512	Ya
SHA-1	160	512	Ada cacat
SHA-256/224	256/224	512	Tidak
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

4. ALGORITMA *MD5*

MD5 adalah fungsi *hash* satu arah yang dibuat oleh Ronald Rivest pada tahun 1991. *MD5* merupakan perbaikan dari *MD4*, setelah *MD4* berhasil diserang *cryptanalyst*. Algoritma *MD5* menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.

Gambaran pembuatan *message digest* dengan algoritma *MD5* diperlihatkan dengan ilustrasi sebagai berikut



:

Gambar 2
Pembuatan *Message Digest* dengan Algoritma *MD5*

Langkah-langkah pembuatan *message digest* secara garis besar adalah sebagai berikut :

1. **Penambahan bit-bit pengganjal (*padding bits*).**

   Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambah bit-bit pengganjal adalah 64 kurang dari kelipatan 512. Angka 512 ini muncul karena *MD5* memproses pesan dalam blok-blok berukuran 512. Pesan dengan panjang 448 bit pun tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri atas sebuah bit 1 diikuti dengan sisanya bit 0.

2. **Penambahan nilai panjang pesan semula.**

   Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambahi lagi dengan 64 bit yang menyatakan panjang semula. Jika panjang pesan > 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

3. **Inisialisasi penyangga (*buffer*) MD.**

   MD5 membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 x 32 = 128 bit. Keempat penyangga ini diberi nama A, B, C, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut :

   A = 01234567

   B = 89ABCDEF

   C = FEDCBA98

   D = 76543210

   Beberapa versi MD5 menggunakan nilai inisialisasi berbeda, yaitu :

   A = 67452301

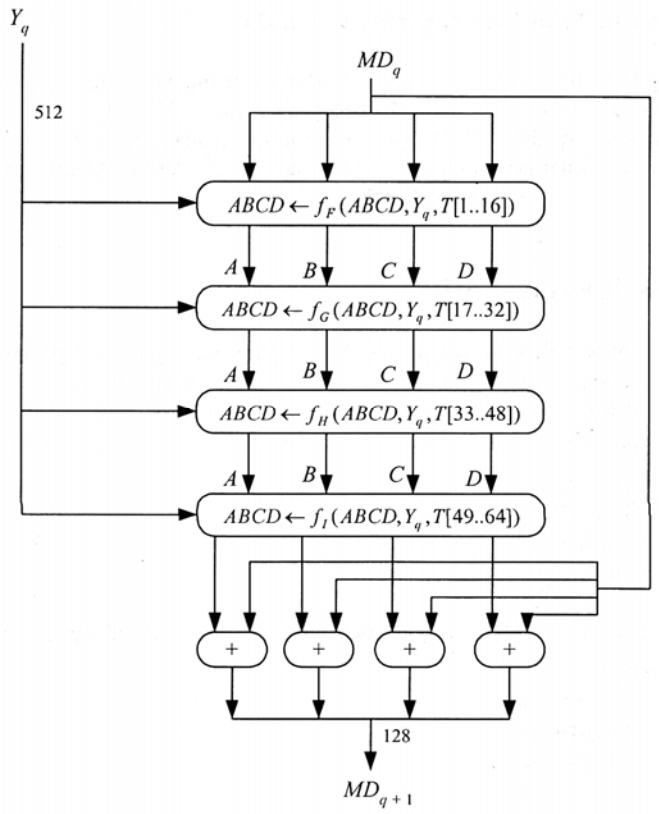
   B = EFCDAB89

   C = 98BADCFE

   D = 10325467

4. **Pengolahan pesan dalam blok berukuran 512 bit.**

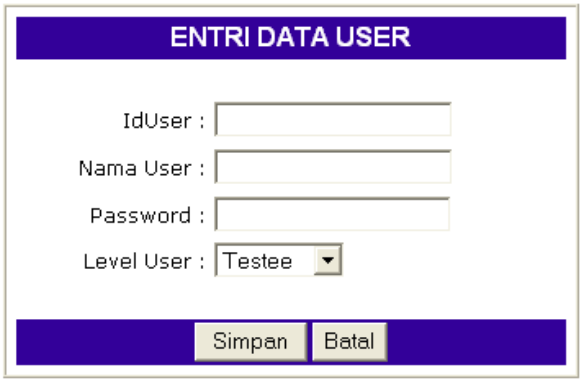
   Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y0 sampai YL-1). Setiap blok 512-bit diproses bersama dengan penyangga MD menjadi keluaran 128-bit, dan ini disebut proses H_{MD5}. Gambaran proses H_{MD5} diperlihatkan dalam diagram Gambar 3. Dalam diagram tersebut diperlihatkan bahwa proses H_{MD5} terdiri atas 4 putaran, masing-masing putaran melakukan operasi dasar *MD5* dengan frekuensi yang sama, dan setiap operasi dasar memakai sebuah elemen *T*. Juga ditunjukkan bahwa Y_q menyatakan blok 512-bit ke-*q* dari pesan-pesan yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. MD_q adalah nilai *message digest* 128-bit dari proses HMD5 ke-q. Pada awal proses, MD_q berisi inisialisasi penyangga *MD*. Fungsi-fungsi f_F, f_G, f_H, f_I masing-masing berisi 16 kali operasi dasar terhadap masukan, setiap operasi dasar menggunakan elemen tabel *T*.



Gambar 3
Pengolahan Blok 512 Bit (Proses H_{MD5})

5. CONTOH IMPLEMENTASI

Pada contoh implementasi ini, sebuah aplikasi berbasis web akan menyimpan data **Password** setiap *user* dalam keadaan ter-enkripsi. Aplikasi dibuat dengan PHP, sedangkan datanya disimpan dalam database MySQL.



Gambar 4
Tampilan Form Input *User* Baru

Berikut ini adalah *script* PHP yang memperlihatkan penggunaan fungsi *MD5* untuk meng-enkripsi data **Password**.

```
require('opendb.php');
$iduser = $_POST['iduser'];
$namauser = $_POST['namauser'];
$password = MD5($_POST['password']);
$leveluser = $_POST['leveluser'];
$sql = "insert into user values
        ('$iduser','$namauser','$password','$leveluser')";
$query = mysql_query($sql,$koneksi;
```

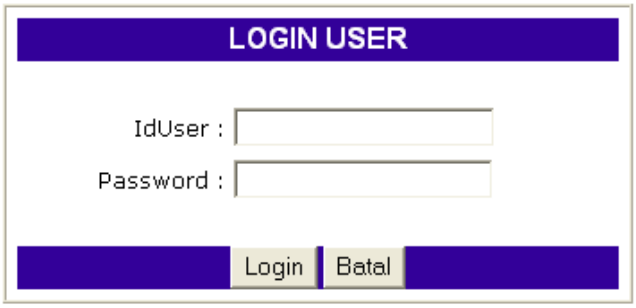
Gambar 5
Script PHP untuk Menyimpan *User* Baru

Data **Password** yang telah di-enkripsi lalu disimpan dalam MySQL. Berikut ini adalah tampilan datanya dalam MySQL :

Etest / user: 16 Records (5 retrieved)

	IdUser	NamaUser	Password	LevelUser
	1122	Danny Setiawan	202cb962ac59075b964b07152d234b70	1
	2222	Iman Budiman	550a141f12de6341fba65b0ad0433500	2
	3333	Heri Setiawan	a516a87cfcaef229b342c437fe2b95f7	3
	1111	Hendi Juanda	01161aaa0b6d1345dd8fe4e481144d84	1
▶	2233	Dadang Hermawan	9fe8593a8a330607d76796b35c64c600	2

Gambar 6
Tampilan Data Ter-enkripsi dalam MySQL



Gambar 7
Tampilan Form *Log-in User*

Ketika *user* melakukan *log-in*, maka data **Password** yang di-inputkan akan di-enkripsi terlebih dahulu, baru kemudian dibandingkan dengan data **Password** di dalam database (yang tersimpan dalam keadaan ter-enkripsi). Jika sama, maka *log-in user* berhasil dan ia dapat mengakses aplikasi tersebut. Sedangkan jika berbeda, maka akan ditampilkan pesan kesalahan bahwa ia tidak dapat mengakses aplikasi.

```
session_start();
require('opendb.php');
$iduser = $_POST['iduser'];
$password = MD5($_POST['password']);
$sql = "select iduser,password
        from user where iduser='$iduser'";
$query = mysql_query($sql,$koneksi);
$data = mysql_fetch_array($query);

if (($data[iduser] == $iduser) and
   ($data[password] == $password))
// iduser & password valid, user dapat mengakses aplikasi
... dst
```

Gambar 8
Script PHP untuk Otentikasi *User*

Permasalahan yang akan muncul jika sebuah data disimpan dalam keadaan ter-enkripsi menggunakan fungsi *hash* satu-arah adalah bahwa data aslinya tidak dapat ditampilkan kembali (di-dekripsi), jadi tentu juga tidak dapat di-*edit*. Solusi yang dapat ditempuh untuk meng-*edit* data tersebut adalah dengan menggantinya dengan data yang baru (di-*replace*).

6. KESIMPULAN

Fungsi *hash* satu-arah adalah fungsi *hash* yang bekerja dalam satu arah : pesan yang sudah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula. Dua pesan yang berbeda akan selalu menghasilkan nilai *hash* yang berbeda pula.

Fungsi ini dapat menerima masukan *string* apa saja. Fungsi *hash* mengkompresi sembarang pesan yang berukuran berapa saja menjadi *message digest* yang selalu tetap. Beberapa algoritma telah dikembangkan menggunakan konsep fungsi *hash* satu-arah, antara lain : *MD2, MD4, MD5, SHA, RIPMEND, WHIRLPOOL.*

Sebagai ilustrasi telah diperlihatkan contoh penggunaan *MD5* untuk kebutuhan suatu proses otentikasi. Algoritma *MD5* menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.

7. DAFTAR PUSTAKA

[1]. Fred Piper & Sean Murphy, *Criptography A Very Short Introduction*, Oxford Press, 2002.

[2]. Lukmanul Hakim, *Membongkar Trik Rahasia Para Master PHP*, Lokomedia, 2008.

[3]. Rinaldi Munir, *Kriptografi*, Penerbit Informatika, 2006.

[4]. Yesus Castagnetto, *Proffesional PHP Programmimg*, Wroxx Press, 1999.

.