

Analisis Penerapan Algoritma MD5 Untuk Pengamanan *Password*

Inayatullah

STMIK MDP Palembang
inayatullah@stmik-mdp.net

Abstrak: Data *password* yang dimiliki oleh pengguna harus dapat dijaga keamanannya. Salah satu cara untuk menjaga keamanan *password* tersebut dengan menggunakan fungsi *hash*. Fungsi *hash* yang banyak digunakan adalah algoritma MD5. Data *password* yang disimpan tidak sama dengan data *password* yang diisikan. Data *password* sudah dalam bentuk pesan ringkas (*message digest*) hasil pengolahan fungsi *hash* sehingga data *password* hanya diketahui oleh pembuat itu sendiri. Waktu yang dibutuhkan untuk pencarian kunci dalam MD5 cukup lama.

Kata Kunci: Kriptografi, *Password*, Fungsi *Hash*, Algoritma MD5.

1 PENDAHULUAN

Masalah keamanan dan kerahasiaan data merupakan salah satu aspek penting dari suatu sistem informasi. Jika berbicara mengenai masalah keamanan yang berkaitan dengan penggunaan komputer, maka sulit memisahkannya dengan kriptografi. Kriptografi bertujuan untuk memberikan layanan keamanan, termasuk keamanan untuk menjaga *password*.

Data *password* yang dimiliki harus dapat dijaga atau dilindungi kerahasiaannya. Jangan sampai data *password* yang ada, jatuh ke tangan orang-orang yang tidak berhak atau berkepentingan. *Password* biasa digunakan untuk layanan otentikasi, baik otentikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication* atau *entity authentication*) maupun mengidentifikasi kebenaran sumber pesan (*data origin authentication*).

Untuk menjaga keamanan *password* dapat dilakukan dengan menyandikan *password* (*plaintext*) tersebut menjadi *ciphertexts* dengan cara di enkripsi atau *enciphering* (standar nama menurut ISO 7498-2). Sedangkan proses mengembalikan *ciphertexts* menjadi *plaintexts* (*password*) semula dinamakan dekripsi. Selain dengan menggunakan penyandian, untuk menjaga keamanan *password* dapat juga dengan menggunakan fungsi *hash*.

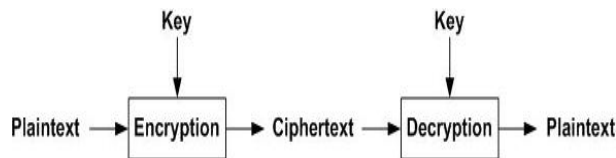
2 TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani "*cryptos*" artinya "*secret*" (rahasia), sedangkan "*graphein*" artinya "*writing*" (tulisan). Jadi kriptografi berarti "*secret writing*" (tulisan rahasia). Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (Bruce Schneier, 1996).

Dalam kriptografi sering ditemukan istilah atau terminologi, seperti pesan (*message*) adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah plainteks (*plaintext*) atau teks jelas (*cleartext*). Pesan dapat berupa data atau informasi yang dikirim (melalui kurir, saluran telekomunikasi, dsb) atau yang disimpan di dalam media perekaman (kertas, *storage*, dsb). Pesan yang tersimpan tidak hanya berupa teks, tetapi dapat berbentuk citra (*image*), suara (*audio*), dan video, atau berkas biner lainnya.

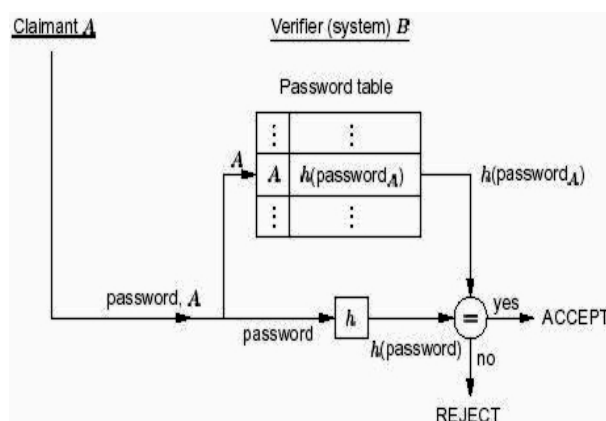
Supaya pesan tidak dapat dimengerti maknanya oleh pihak lain, maka pesan perlu disandikan ke bentuk lain yang tidak dapat dipahami. Bentuk yang tersandi disebut *ciphertext* atau kriptogram yang harus bisa ditransformasikan kembali menjadi plainteks semula agar pesan yang diterima bisa dibaca. Gambar 1 memperlihatkan enkripsi dan dekripsi.



Gambar 1: Proses Enkripsi dan Dekripsi

2.2 Password

Password atau kata sandi dapat digunakan untuk layanan otentikasi, yaitu layanan yang berhubungan identifikasi, baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication* atau *entity authentication*) maupun mengidentifikasi kebenaran sumber pesan. Dua pihak yang saling berkomunikasi harus dapat mengotentikasi satu sama lain sehingga ia dapat memastikan sumber pesan. Otentikasi sumber pesan secara implisit juga memberikan kepastian integritas data, sebab jika pesan telah dimodifikasi berarti sumber pesan sudah tidak benar. Gambar 2 memperlihatkan skema *password*.



Gambar 2: Skema Password

2.3 Fungsi Hash

Fungsi *hash* adalah fungsi yang menerima masukan *string* yang panjangnya 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*). Pesan ringkas dinyatakan dalam kode heksadesimal yang panjangnya 128 bit. Satu karakter heksadesimal = 4 bit. Dua pesan yang berbeda akan selalu menghasilkan nilai *hash* yang berbeda pula. Sifat-sifat fungsi *hash*:

1. Fungsi *H* dapat diterapkan pada blok data berukuran berapa saja.
2. *H* menghasilkan nilai (*h*) dengan panjang tetap.
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.

2.4 Algoritma MD5

Fungsi *hash* yang banyak digunakan dalam kriptografi MD5 dan SHA. Dalam artikel ini fungsi *hash* yang digunakan algoritma MD5. MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.

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

1. Penambahan bit-bit pengganjal (*padding bits*).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (*buffer*) MD.
4. Pengolahan pesan dalam blok berukuran 512 bit.

2.5 Kompleksitas Serangan

Kompleksitas serangan dapat diukur dengan beberapa cara, antara lain kompleksitas data, kompleksitas waktu, dan kompleksitas memori.

1. Kompleksitas data (*data complexity*)
Jumlah data (plainteks dan cipherteks) yang dibutuhkan sebagai masukan untuk serangan. Semakin banyak data yang dibutuhkan untuk melakukan serangan, semakin kompleks

serangan tersebut, yang berarti semakin bagus sistem kriptografi tersebut.

2. Kompleksitas waktu (*time complexity*)
Waktu yang dibutuhkan untuk melakukan serangan. Semakin lama waktu yang dibutuhkan untuk melakukan serangan, berarti semakin bagus sistem kriptografi tersebut.

Tabel 1: Waktu yang Diperlukan
Exhaustive Key Search

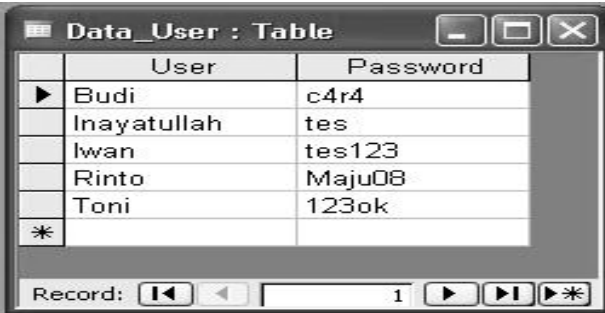
Ukuran Kunci	Jumlah Kemungkinan Kunci	Lama Waktu Untuk 106 Percobaan per Detik	Lama Waktu Untuk 1012 Percobaan per Detik
16 bit	$2^{16} = 65536$	32.7 milidetik	0.0327 mikrodetik
32 bit	$2^{32} = 4.3 \times 10^9$	35.8 menit	2.15 milidetik
56 bit	$2^{56} = 7.2 \times 10^{16}$	1142 tahun	10.01 jam
128 bit	$2^{128} = 4.3 \times 10^{38}$	5.4×10^{24} tahun	5.4×10^{18} tahun

3. Kompleksitas ruang memori (*space/storage complexity*)
Jumlah memori yang dibutuhkan untuk melakukan serangan. Semakin banyak memori yang dibutuhkan untuk melakukan serangan, berarti semakin bagus sistem kriptografi tersebut.

3 PEMBAHASAN

3.1 Sumber Data

Sumber data yang akan digunakan dalam tulisan ini berupa data alfanumerik.



	User	Password
▶	Budi	c4r4
	Inayatullah	tes
	Iwan	tes123
	Rinto	Maju08
	Toni	123ok
*		

Gambar 3: Sumber Data yang Digunakan

3.2 Aplikasi

Bahasa pemrograman yang digunakan *Microsoft Visual Basic* versi 6.0, *database* yang digunakan *MS. Access* 2003.



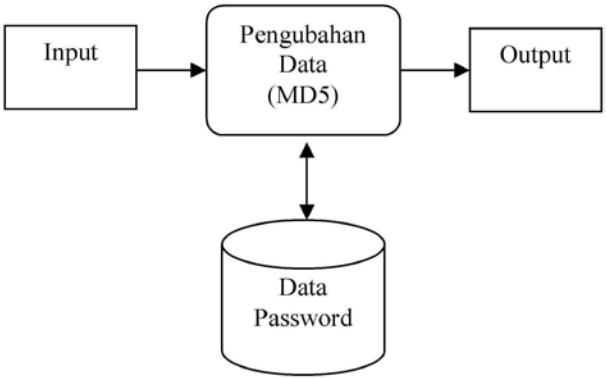
Gambar 4: Tampilan Aplikasi

3.3 Perangkat

Komputer dengan spesifikasi *Pentium* 3, RAM 128, sistem operasi *Windows* XP SP 1, dan *Harddisk* 20 GB.

3.4 Proses Pengisian dan Penyimpanan data Password

Untuk proses pengisian dan penyimpanan data *password*, melalui tahap sebagai berikut:



Gambar 5: Alur Proses Pembuatan Password

- Penjelasan:
1. Jika data *password* belum ada, pengguna melakukan registrasi. Jika data sudah ada pengguna bisa langsung melakukan *login*.



Gambar 6: Pengguna Melakukan Registrasi

2. Data *password* yang diisikan mengalami perubahan dalam bentuk pesan ringkas dengan fungsi *hash*.

M = Data *password*
H = Fungsi *Hash*
h = Pesan ringkas
h = H(M)

Data *password* yang disimpan sudah merupakan *message digest* (pesan ringkas) dari hasil pengolahan fungsi *hash* terhadap *password* yang diisikan oleh pengguna atau *user*. Pesan ringkas yang dihasilkan dengan panjang 128 bit atau 32 karakter heksadesimal.

registrasi : Table		
	User	Password
+	Budi	78c2212328663337589020946a9b4f3
+	Inayatullah	28b662d883b6d76fd96e4ddc5e9ba780
+	Iwan	b93939873fd4923043b9dec975811f66
+	Toni	dbfDecbe1a3d12d50699158bf6370431
*		

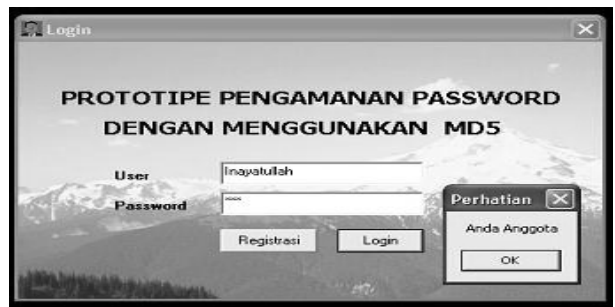
Gambar 7: Data *Password* yang Telah Diubah

3.5 Pengujian Data

Pengujian data dilakukan untuk mengetahui keakuratan program dalam implementasinya.

Pengguna memasukan *password* sesuai dengan data pada waktu registrasi. Data *password* yang diisikan diubah oleh aplikasi menjadi pesan ringkas (*message digest*). Pesan ringkas dari data *password* yang telah diisikan disamakan dengan data *password* yang telah disimpan dalam *database*. Jika $h = h'$ maka *password* yang diisikan benar.

h = pesan ringkas
h' = pesan ringkas dalam *database*
 $h = h' \Rightarrow$ *password* benar



Gambar 8: Tampilan *Login* Dengan *Password* yang Benar

Jika $h \neq h'$ maka *password* yang diisikan tidak benar atau salah.

h = pesan ringkas
h' = pesan ringkas dalam *database*
 $h \neq h' \Rightarrow$ *password* salah

3.6 Analisis

1. Data *password* yang digunakan dapat berupa data alfanumerik, sehingga tidak mengganggu pengguna dalam membuat *password*. Pengguna dapat membuat *password* dengan alfabet, numerik atau gabungan keduanya.
2. Data *password* yang disimpan bukan lagi data *password* yang sama dengan data *password* yang diisikan sehingga data *password* yang dimiliki oleh pengguna hanya diketahui oleh pengguna itu sendiri jadi untuk kerahasiaannya lebih terjamin.
3. Kompleksitas waktu yang dimiliki baik. Berdasarkan teknik yang digunakan dalam menemukan kunci (*exhaustive attack* atau

brute force attack), semakin panjang kunci semakin lama waktu yang dibutuhkan untuk pencarian kunci sehingga semakin bagus sistem kriptografi tersebut (Tabel 1 halaman 3).

4 KESIMPULAN

Berdasarkan hasil dari ujicoba dan pembahasan di atas, maka kesimpulan yang didapat sebagai berikut: Algoritma MD5 dapat digunakan atau diterapkan untuk pengamanan *password* dari pengguna dalam suatu *database*. Semakin besar ukuran kunci, semakin lama waktu yang dibutuhkan untuk melakukan pencarian terhadap kunci. Pengguna harus mengingat *password* yang telah dibuat, karena *password* yang disimpan telah mengalami perubahan menjadi pesan ringkas (*message digest*) yang berbeda dengan aslinya.

DAFTAR PUSTAKA

- [1] Halvorson, M., 2001. *Visual Basic 6*, Jakarta: Elex Media Komputindo.
- [2] Kurniawan, Yusuf, 2004. *Kriptografi Keamanan Internet dan Jaringan Komunikasi*, Bandung: Informatika.
- [3] Menezes, A., Oorschot, P. van & Vanstone, S., 1997. *Handbook of Applied Cryptography*, USA: CRC Press INC.
- [4] Munir, R., 2006. *Kriptografi*, Bandung: Informatika.
- [5] Stalling, W., 1985. *Data and Computer Communication*, Fourth Edition, USA: Prentice Hall.