Implementasi MD5 Hash dengan Random Salt serta Analisis Keamanannya

Setia Negara B. Tjaru / 13508054
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Setia.negara.91@gmail.com

***Abstrak*—Password menjadi hal yang kita butuhkan sehari-hari. Banyak kegiatan kita pada internet yang membutuhkan autentikasi password. Oleh karena itu keamanan password menjadi sangat penting untuk dijaga. Bukan hanya oleh pemilik akun tapi juga pemilik website. Sudah banyak kasus bobolnya basis data website-website besar yang menyebababkan keamanan password terganggu. Hal ini disebabkan banyak orang yang memakai password yang sama pada website yang berbeda.**

Karena itu password disimpan pada basis data sebaiknya dalam keadaan ter-hash. Agar tidak berupa plaintext sehingga penyerang meskipun mendapatkan basis data tapi harus melakukan dehash terlebih dahulu. Proses mendapatkan password dari hash ini juga ternyata bisa dilakukan dengan beberapa cara seperti brute force atau rainbow table.

Oleh karena itu, hash mesti diperkuat seperti dengan memakai salt atau melakukan hash berulang kali. Salt berguna untuk mengubah kondisi awal hash, yitu dengan cara menambahkan password dengan teks lain. Teks lain ini bisa saja random atau tetap tapi rahasia. Dengan menggunakan salt maka penyerangan akan menjadi lambat untuk mendapatkan banyak password sekaligus, meskipun untuk mendapatkan satu password masih bisa didapatkan dengan cepat. Salah satu algoritma yang masih sering dipakai adalah MD5. Meskipun beberapa tahun lalu telah ditemukan adanya celah keamanan pada algoritma ini namun masih tetap relevan dipakai dalam proses hash.

Hash berulang kali membuat proses komputasi lebih lama sehingga akan membuat proses pembuatan precomputed table menjadi lebih lama.

Sedangkan salt membuat password tidak bisa didapatkan jika salt atau proses pembuatan saltnya tidak diketahui.

***Index Terms*—Hash, Password, MD5, Basis Data, Website, Salt.**

I. PENDAHULUAN

Pada jaman sekarang ini, teknologi internet sudah berkembang pesat. Banyak hal yang dilakukan dalam internet. Password pun sudah menjadi hal yang sangat akrab dengan kegiatan sehari-hari. Masuk ke sosial media, e-mail, situs kuliah, banyak hal yang membutuhkan password. Oleh karena itu password mesti dijaga kerahasiannya. Bukan hanya oleh sang pemilik password, tapi juga pemilik website yang menyimpan password. Karena banyak orang yang menggunakan password yang sama pada akun berbeda. Hal ini menyebabkan bocornya password akun pada satu situs maka banyak akun pada situs lain yang bocor.

Oleh karena itu selain keamanan website dijaga, password dalam basis data juga dijaga agar tidak hanya berupa plaintext. Salah satu yang bisa dilakukan adalah dengan menerapkan hash pada password. Hash adalah algoritma dimana sebuah text dipetakan ke text yang jumlahnya tetap.

Salah satu algoritma hash yang banyak dipakai pada website-website jaman sekarang adalah MD5. MD5 adalah hash yang menghasilkan 128 bit nilai hash. Dalam hal ini berarti 16 byte. Value ini biasanya direpresentasikan ke dalam heksadesimal sepanjang 32.

Hash membuat basis data atau tabel password menjadi tidak plaintext, namun bukan berarti tidak dapat dipecahkan. Password yang telah di-hash dapat di-dehash dengan precomputed table, biasa disebut dengan rainbow table, yaitu tabel yang berisi hasil hash dari banyak teks yang mungkin menjadi password. Dengan rainbow table, jika kita bisa mendapatkan tabel password, maka proses dehash hanyalah proses look up nilai hash yang sama pada kedua tabel.

Oleh karena itu diperlukan cara untuk memperkuat hash, diantaranya adalah hash berulang dan salt.

II. MD5 HASH

Algoritma hash MD5 adalah salah satu algoritma hash yang banyak dipakai pada website-website dan aplikasi signature. MD5 didesain oleh profesor Ron Rivest pada tahun 1991 untuk menggantikan MD4 yang akan kadaluarsa. MD5 menghasilkan hash value 128 bit yang direpresentasikan denganhexadecimal sepanjang 32 digit.

Meski populer, MD5 sudah ditemukan "ketidakamanannya". Pada tahun 2005, dibuktikan bahwa MD5 tidaklah collision resistant.

Collision resistant adalah properti yang dimiliki sebuah algoritma hash yang merepresentasikan sulitnya menemukan hash value yang sama dari dua text berbeda. Ini bukan berarti bahwa tidak ada collision, karena pada dasarnya algoritma hash pasti memiliki collition, tapi apakah ada cara yang lebih mudah untuk menemukan collition dibanding dengan brute force.

Dengan ditemukannya serangan collition pada tahun

2004 tersebut, banyak ahli yang menyaranan untuk beralih ke SHA-1, algoritma has yang lain. Namun saat ini SHA-1 juga telah ditemukan kelemahannya.

Pada hal-hal seperti sertifikat SSL misalnya, MD5 sudah tidak dipakai lagi. Meskipun begitu MD5 masih banyak dipakai untuk melindungi password pada website-website di internet.

A. Algoritma MD5

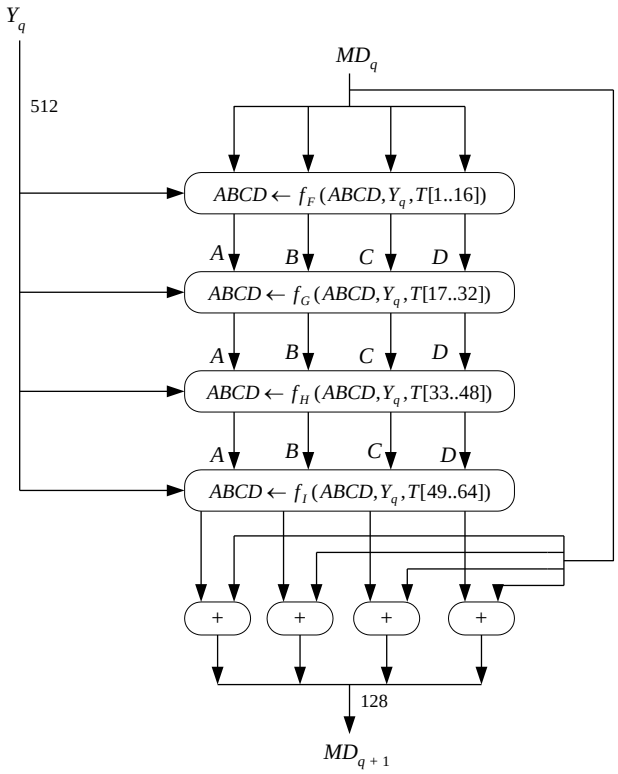
Langkah-langkah yang dibutuhkan untuk MD5 adalah sebagai berikut:

1. Penambahan bit-bit pengganjal (padding bits).
   Pesan ditambah dengan sejumlah bit pengganjal sedemikian rupa sehingga didapatkan panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. 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 dari sebuah bit 1 diikuti dengan sisanya bit 0.

2. Penambahan nilai panjang pesan semula.
   Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan > 264 maka yang diambil adalah panjangnya dalam modulo 264. Dengan kata lain, jika panjang pesan semula adalah K bit, maka 64 bit yang ditambahkan menyatakan K modulo 264. 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 □ 32 = 128 bit. Keempat penyangga ini menampung hasil antara dan hasil akhir. 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

4. Pengolahan pesan dalam blok berukuran 512 bit.
   Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-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 pada gambar berikut:



Gambar 1. Skema MD5

B. Collition MD5

Pada tahun 1996, collition ditemukan pada fungsi kompresi MD5. Kemudian pada tahun 2005, peneliti dapat membuat dua dokumen yang memiliki hash yang sama. Pada tahun yang sama Ron Rivest mengumumkan bahwa MD5 “broken” dalam hal collition resistance. Pada tahun 2007 ditemukan cara penyerangan chosen prefix collition, yaitu meng-append dua value berbeda yang telah dihitung sebelumnya kepada dua dokumen berbeda. Dua dokumen baru ini pun akan menghasilkan hash yang sama.

Contoh MD5 collition yang terkenal adalah pesan dengan perbedaan 6 bit berikut, perbedaan bitnya di garis bawahi dan ditebalkan:

Pesan 1:
d131dd02c5e6eec4 693d9a0698aff95c
2fcab5***8***712467eab 4004583eb8fb7f89
55ad340609f4b302 83e4888325***7***1415a
085125e8f7cdc99f d91dbd***f***280373c5b
d8823e3156348f5b ae6dacd436c919c6
dd53e2***b***487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080***a***80d1e
c69821bcb6a88393 96f965***2***b6ff72a70

Pesan 2:

d131dd02c5e6eec4 693d9a0698aff95c
2fcab5***0***712467eab 4004583eb8fb7f89
55ad340609f4b302 83e4888325***f***1415a
085125e8f7cdc99f d91dbd7280373c5b
d8823e3156348f5b ae6dacd436c919c6
dd53e2***3***487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080***2***80d1e
c69821bcb6a88393 96f965***a***b6ff72a70

kedua pesan ini menghasilkan MD5 hash yang sama yaitu:
79054025255fb1a26e4bc422aef54eb4.

III. CRACKING PASSWORD

Ada banyak cara untuk mendapatkan password. Berikut adalah cara yang umum digunakan

A. Brute Force

Brute force adalah mencoba semua kemungkinan yang ada untuk mendapatkan password. Jika misalnya diketahui maksimal password adalah 8 karakter, dengan domain karakter a-z. Maka brute force adalah mencoba semua kemungkinan dari a hingga zzzzzzzz hingga menemukan password yang benar.
Pada praktiknya, serangan seperti ini suda hampir tidak mungkin dilakukan karena banyaknya cara yang dapat digunakan untuk mencegahnya. Contohnya adalah misalnya password hanya boleh salah sebanyak 3 kali sebelum di blokir sementara, memberikan waktu delay tiap input password yang salah, atau yang populer sekarang yaitu dengan captcha, text verifikasi bahwa yang melakukan input adalah manusia, bukan machine.

B. Dictionary Attack

Mirip seperti brute force yaitu mencoba semua kemungkinan. Namun Dictionary attack lebih terarah karena menggunakan dictionary atau kamus. Kamus berisi kombinasi kata-kata yang lazim dipakai sebagai password atau perpaduan kata dari kamus sebenarnya.Isi dictionary ini pun diinputkan satu persatu ke form login.

Jika penyerang bisa mendapatkan database atau basis data website, maka dictionary attack dapat dilakukan dengan precomputed table. Isi dictionary semuanya di-hash satu persatu kemudian dilakukan lookup antara hash yang didapatkan pada database dibandingkan dengan hash pada precomputed table.

C. Precomputed Table

Pada skema penyerangan Precomputed Table, penyerang harus mendapatkan basis data website yang ter-hash. Kemudian penyerang akan membuat banyak kemungkinan password kemudian melakukan hash kedalamnya. Setelah tabel selesai dibuat maka dilakukan lookup perbandingan satu persatu, apakah ada hash yang sama antara tabel password dengan tabel precomputed.

IV. MEMPERKUAT HASH MD5

A. Hash Berulang

Salah satu cara untuk memperkuat hash MD5 adalah dengan cara menerapkan hash berulang. Yang sering dipakai pada website-website adalah perulangan tiga kali, atau $MD5^3$.
Perulangan ini berguna untuk memperlambat penghitungan nilai hash, sehingga jika penyerang membuat precomputed table maka prosesnya akan bertambah lama.
Kelemahan dari cara ini adalah server website juga akan terbebani oleh penghitungan hash yang begitu banyak. Oleh karena itu biasa disiati dengan melakukan MD5 pada client-side, tentu saja dengan fallback server-side untuk berjaga-jaga jikaclient-side tidak berjalan.

B. Salt

Salt adalah data atau teks yang dipakai untuk menyulitkan penyerang passsword. Salt dimasukkan kedalam proses hash sebagai tambahan input. Hal ini menyebabkan nilai hash akan berubah jauh dari hash sebelumnya tanpa salt. Salt dapat dipilih tetap atau acak.
Salt ini berguna untuk mengurangi keefisienan precomputed table yang mencoba mpendapatkan banyak password sekaligus dalam satu table. Dengan salt maka penyerangan tidak dapat dilakukan secara paralel dengan lookup password dalam satu tabel. Tapi penyerang harus terlebih dahulu menggenerate tabel untuk tiap-tiap salt. Tentu saja, penerang terlebih dauhulu harus mengetahui salt-nya apa atau bagaimana cara menggenerate salt-nya

V. IMPLEMENTASI

Implementasi dilakukan pada bahasa pemrograman Java. Pada implementasi MD5 mengikuti algoritma aturan sesuai spesifikasi RSA.
Untuk MD5 tiga kali, dilakukan sebagai berikut: MD5(MD5(MD5(“pesan”))).
Sementara untuk MD5 berantai 100 kali. Dilakukan implementasi sama seperti MD5 tiga kali hanya berantai 100 kali.

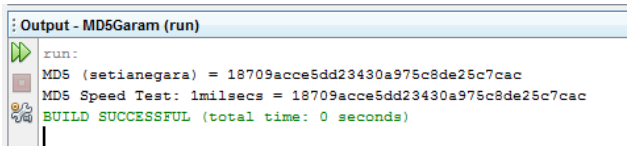
Sementara untuk salt, impleentasi dilakukan dengan penambahan string alfanumerik bergantung pada password yang dimasukkan. Seperti berikut:

```
masukan = new String("setianegara");
SecureRandom random = new SecureRandom(masukan.getBytes());
String salt = new BigInteger(35, random).toString(32);
System.out.println(salt);
```

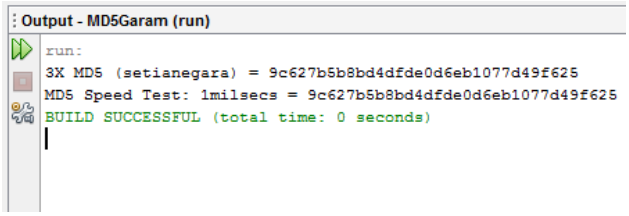
Kemudian MD5 dilakukan dengan mengappend salt pada akhir pesan masukan.

```
masukan = new String("setianegara" + salt);
chrTestData = masukan.toCharArray();
md5Test.update(chrTestData, chrTestData.length);
md5Test.md5final();
System.out.println("MD5 (" + masukan + " + " +  salt + ") = " + md5Test.toHexString());
```

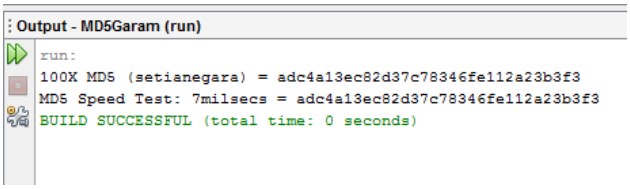
Berikut contoh output-outputnya:



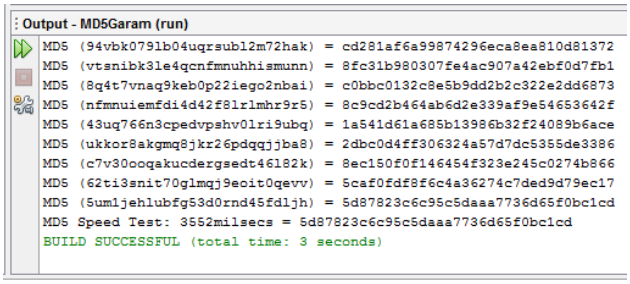
Gambar 2. MD5 dengan 1 inputan



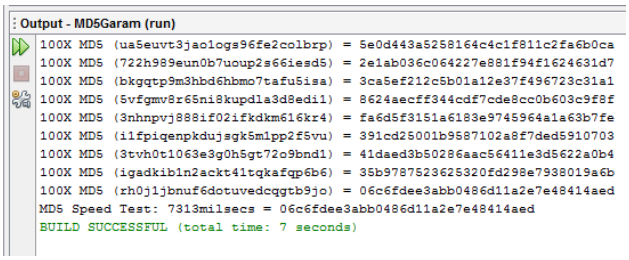
Gambar 3. MD5 berantai 3 kali



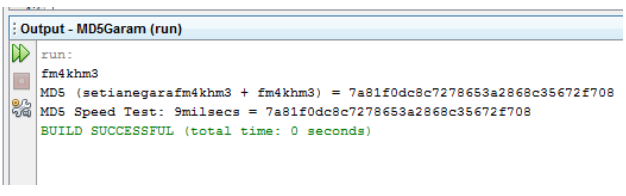
Gambar 4. MD5 berantai 100 kali



Gambar 5. MD5 10000 Input Random



Gambar 6. MD5 Berantai 100kali dengan 10000 Input Random

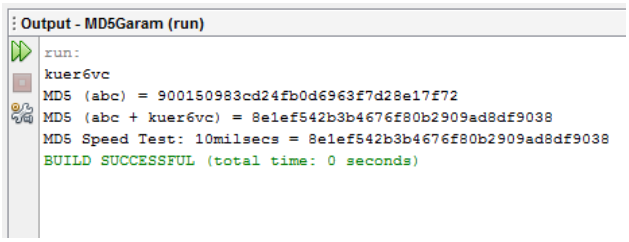


Gambar 7. MD5 dengan Random Salt dengan Seed Masukan

VI. ANALISIS

Terlihat pada implementasi, untuk melakukan MD5 pada string “setianegara” dibutuhkan 1 milisecond. Sementara pada MD5 berantai 100 kali, dalam hal ini penerapan hash berulang, membutuhkan waktu 7 milisecond.
Untuk 10000 input random dibutuhkan waktu 3.552 milisecond, sedangkan untuk berantai 100 kali untuk 10000 input dibutuhkan waktu 7.313 miliseconds
Disini terlihat bahwa MD5 berantai meningkatkan waktu komputasi hash, meskipun tidak signifikan.

Kemudian untuk Hash dengan salt, dibutuhkan waktu 9 milisecond untuk prosesnya. Hal ini disebabkan fungsi secure random yang sebenarnya perhitungannya mahal yang menyebabkan waktu komputasi bertambah. Hal ini menjadi kelebihan tersendiri karena semakin lambat komputasi hash akan membuat penyerang harusmembuang waktu lebih lama.



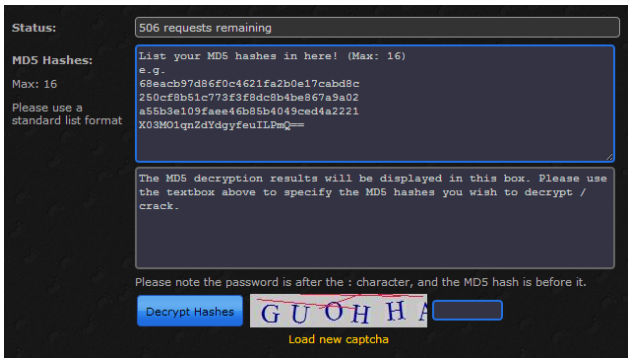
Gambar 8. Perbandingan Tanpa Salt dengan Salt

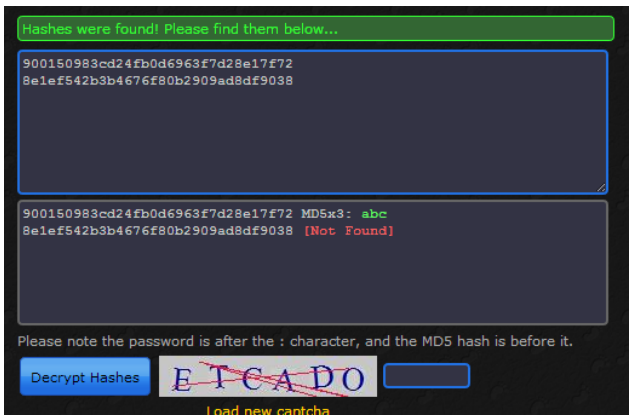
Kita periksa apa salt tadi meningkatkan keamanan:

Teks: abc
MD5: 900150983cd24fb0d6963f7d28e17f72
MD5 + Salt: 8e1ef542b3b4676f80b2909ad8df9038

Kita cek dengan rainbow table yang tersedia di internet. Pada kasus ini diambil situs www.md5decrypter.co.uk.

Berikut tampilannya:





Setelah dimasukkan ke rainbow table, hanya hash pertama yang ditemukan, yaitu sama dengan aslinya yaitu string "abc". Sedangkan hasil dengan salt tidak ditemukan.

Dari segi keamanan, MD5 dengan tambahan salt ini terbukti tahan dari precomputed table.

VII. SIMPULAN

1. Menerapkan hash seperti MD5 bukan berarti keamanan sudah terjamin. MD5 memiliki collition sehingga sudah tudak aman dipakai pada signature atau certficate. Precomputed table atau rainbow table sudah banyak bertebaran di internet untuk password-pasword yang sudah umum.

2. Ada beberapa cara untuk memperkuat keamanan pada hash yaitu: 1. menerapkan hash berulang, untuk meperlambat proses komputasi hash, sehingga pembuatan rainbow tabel menjadi lama, dan 2. Dengan Salt, untuk "mengotori" plaintext password sehingga hasil hash menjadi jauh berbeda.

3. Salt secara tidak langsung sudah menjadi kebutuhan dalam penerapan keamanan pada hash. Salt memperkuat password dengan cara menambahkan teks secara random dan atau rahasia pada password sebelum di-hash, sehingga hasil hash yang diberikan akan berbeda jauh. Proses random salt juga memperlambat proses komputasi sehingga menerapkan skema yang sama seperti hash berulang, namun lebih efisien.

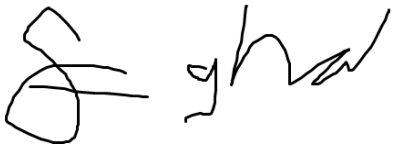
PUSTAKA

[1] The MD5 Message-Digest Algorithm, http://tools.ietf.org/html/rfc1321
[2] ISC Diary | Hashing Passord, http://www.dshield.org/diary/Hashing+Passwords/11110
[3] Why You Should Always Salt Your Hashes, http://www.addedbytes.com/blog/why-you-should-always-salt-your-hashes
[4] Rinaldi Munir. Slide kuliah "Fungsi Hash".
[5] Rinaldi Munir. Slide kuliah "Algoritma MD5".

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, 20 Mei 2013



Setia Negara/13508054