STUDI LITERATUR KRIPTOGRAFI DENGAN ALGORITMA MESSAGE DIGEST 5 (MD5)

Timothy John Pattiasina, ST., M.Kom.*
Adriane Lysandra*

ABSTRAK

Didalam melakukan pengiriman suatu pesan, hingga pesan itu diterima, terdapat persoalan yang sangat penting untuk diperhatikan, kerahasiaan, autentikasi, keutuhan dan tak berbantahkan (non-repudiation). Message Digest 5 (MD5) adalah salah satu algoritma yang dapat memberikan garansi bahwa pesan yang dikirim akan sama dengan pesan yang diterima, hal ini dilakukan dengan cara membandingkan 'sidik jari' atau 'intisari pesan' kedua pesan tersebut. MD5 merupakan pengembangan dari MD4 dimana terjadi penambahan satu ronde. MD5 memproses teks masukan ke dalam blok-blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD5 berupa 4 buah blok yang masing-masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai hash. Studi literatur kriptografi ini memiliki tujuan untuk menganalisa proses keutuhan atau perubahan pesan dengan menggunakan algoritma MD5.

Kata Kunci: Kriptografi, Hash, MD5

1. PENDAHULUAN

Kriptografi merupakan dasar untuk memahami keamanan pada komputer, khususnya keamanan jaringan. Kriptografi sudah digunakan hampir di segala bidang yang terkait dengan penggunaan jaringan komputer. Bahkan kehidupan saat ini dilingkupi oleh kriptografi, mulai dari transaksi di bank, transaksi kartu kredit, percaapan melalui telepon genggam, akses internet hingga pada saat mengaktifkan peluru kendali pun menggunakan kriptografi. Begitu pentingnya kriptografi untuk keamanan informasi, sehingga jika berbicara mengenai masalah keamanan yang berkaitan dengan penggunaan komputer, maka orang tidak bisa memisahkannya dengan kriptografi.

1.1. Latar Belakang

Pada saat kita menerima atau mengirim pesan pada jaringan, terdapat empat buah persoalan yang sangat penting, yaitu : *confidentiality*, *integrity*, *authentication* dan *Non-Repudiation*. *Confidentiality* adalah aspek untuk menjamin bahwa data – data tersebut hanya bisa diakses oleh pihak – pihak tertentu saja. *Integrity* adalah Tuntutan yang berhubungan dengan jaminan setiap pesan yang pasti sampai pada penerimanya tanpa ada bagian dari pesan tersebut yang diganti, diduplikasi, dirusak, diubah urutannya, dan ditambahkan. *Authentication* yang dilakukan baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya seperti yang diklaim. Serta *Non-Repudiation.*

* Staf Pengajar Program Studi S1-Teknik Informatika IKADO
* Mahasiswa Program Studi S1-Teknik Informatika IKADO

Didalam kriptografi, terdapat sebuah fungsi yang sesuai untuk aplikasi keamanan seperti autentifikasi dan integritas pesan. Fungsi tersebut dinamakan fungsi *hash* kriptografi. Salah satu fungsi *hash* yang paling banyak digunakan adalah *message digest 5*(MD5). Algoritma MD5 adalah fungsi hash satu arah yang dibuat oleh Ron Rivest dan merupakan pengembangan dari algoritma MD4. Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan sebuah *message digest* dengan panjang 128 bit.

1.2. Perumusan Masalah

Perumusan masalah untuk penerapan kriptografi dengan algoritma MD5 adalah sebagai berikut:

1. Apakah Kriptografi dengan algoritma MD5 dapat menyandikan pesan dengan baik dibandingkan dengan metode MD2 dan MD4 yang merupakan pendahulunya?
2. Bagaimanakah algoritma MD5 melakukan enkripsi terhadap suatu pesan?

1.3. Tujuan dan Manfaat Penelitian

Penelitian ini bertujuan untuk mengetahui dan membuktikan bahwa kriptografi dengan algoritma MD5 lebih baik dibandingkan dengan pendahulunya, yaitu algoritma MD2 dan MD4.

Sedangkan manfaat yang didapatkan melalui penelitian ini adalah kemampuan untuk menjabarkan proses dan cara kerja algoritma MD5, MD2 dan MD4 pada khalayak umum. Serta menambah wawasan bagi penulis maupun pembaca tentang keanekaragaman algoritma fungsi *hash* satu arah.

2. LANDASAN TEORI

Telah diketahui pada latar belakang penelitian ini, bahwa kriptografi dapat dilakukan dengan menggunakan algoritma MD5. Untuk mengenal lebih lanjut apa itu kriptografi, algoritma MD5, MD2 dan MD4, akan dibahas pada sub bab pada bab ini.

2.1. Kriptografi

Kriptografi sudah lama digunakan oleh tentara Sparta di Yunani pada permulaan tahun 400 SM. Mereka menggunakan alat yang namanya *scytale.* Sebelum komputer ada, kriptografi dilakukan dengan menggunakan pensil dan kertas. Algoritma kriptografi (cipher) yang digunakan saat itu, dinamakan juga algoritma klasik, adalah berbasis karakter, yaitu enkripsi dan dekripsi dilakukan pada setiap karakter pesan (Schneier, 1996). Informasi lengkap mengenai sejarah kriptografi dapat ditemukan di dalam buku David Kahn yang berjudul The Codebreakers. Buku yang tebalnya 1000 halaman ini menulis secara rinci sejarah kriptografi mulai dari penggunaan kriptografi oleh bangsa Mesir 4000 tahun yang lalu (berupa hieroglyph yang tidak standard pada piramid) hingga penggunaan kriptografi pada abad ke-20.

Sampai pada akhir perang dunia I, kriptografi merupakan disiplin ilmu matematika yang hanya dipelajari oleh orang-orang tertentu saja. Penelitian bidang ini tidak pernah sampai kepada umum. Kriptografi juga digunakan di kalangan militer. Pada perang dunia ke 2, pemerintah Nazi Jerman membuat mesin enkripsi yang dinamakan Enigma. Mesin yang menggunakan beberapa buah rotor (roda berputar) ini melakukan enkripsi dengan cara yang sangat rumit. Namun Enigma cipher berhasil

dipecahkan oleh pihak sekutu dan keberhasilan memecahkan Enigma sering dikatakan sebagai faktor yang memperpendek perang dunia ke 2.

Perkembangan komputer dan sistem komunikasi pada tahun 60an berdampak pada permintaan dari sektor-sektor privat sebagai sarana untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan. Dimulai dari usaha Feistel pada IBM di awal tahun 70an dan mencapai puncaknya pada 1977 dengan pengangkatan DES (Data Encryption Standard) sebagai standar pemrosesan informasi federal US untuk mengenkripsi informasi yang unclassified. DES merupakan mekanisme kriptografi yang paling dikenal sepanjang sejarah.

Pengembangan paling mengejutkan dalam sejarah kriptografi terjadi pada 1976 saat Whitfield Diffie dan Martin Hellman mempublikasikan New Directions in Cryptography. Tulisan ini memperkenalkan konsep revolusioner kriptografi kunci publik dan juga memberikan metode baru dan jenius untuk pertukaran kunci, keamanan yang berdasar pada kekuatan masalah logaritma diskret. Walaupun penulis tesis tersebut tidak mempunyai praktek yang nyata akan bentuk skema enkripsi kunci publik pada saat itu, tetapi ide tersebut memicu minat dan aktivitas yang besar dalam komunitas kriptografi. Pada tahun 1978, Rivest, Shamir, dan Adleman menemukan enkripsi kunci publik yang pertama dan sekarang ini dikenal dengan nama RSA (Rivest, Shamir, and Adleman). Skema RSA didasarkan pada permasalahan matematika sulit yang terdiri dari pemfaktoran terhadap bilangan besar. Skema kunci publik lainnya yang kuat dan praktis ditemukan oleh El Gamal. Skema ini juga didasarkan masalah logaritma diskrit.

2.2. Fungsi Hash Satu Arah

Hash function atau fungsi hash adalah suatu cara menciptakan "*fingerprint*" dari berbagai data masukan. *Hash function* akan mengganti atau mentranspose-kan data tersebut untuk menciptakan *fingerprint*, yang biasa disebut *hash value*. *Hash value* biasanya digambarkan sebagai suatu string pendek yang terdiri atas huruf dan angka yang terlihat random (data biner yang ditulis dalam notasi heksadesimal). Suatu *hash function* adalah sebuah fungsi matematika, yang mengambil sebuah panjang variabel string input, yang disebut *pre-image* dan mengkonversikannya ke sebuah string output dengan panjang yang tetap dan biasanya lebih kecil, yang disebut *message digest*5. *Hash function* digunakan untuk melakukan *fingerprint* pada *pre-image*, yaitu menghasilkan sebuah nilai yang dapat menandai (mewakili) *pre-image* sesungguhnya.

Fungsi *hash* satu arah (o*ne-way hash function*) adalah *hash function* yang bekerja satu arah, yaitu suatu *hash function* yang dengan mudah dapat menghitung *hash value* dari *pre-image*, tetapi sangat sukar untuk menghitung *pre-image* dari *hash value*. Sebuah fungsi *hash* satu arah, *H(M)*, beroperasi pada suatu *pre-image* pesan *M* dengan panjang sembarang, dan mengembalikan nilai *hash h* yang memiliki panjang tetap. Dalam notasi matematika fungsi *hash* satu arah dapat ditulis sebagai:

```
h = H(M), dengan h memiliki panjang b
```

Ada banyak fungsi yang mampu menerima input dengan panjang sembarangan menghasilkan output dengan panjang tetap, tetapi fungsi *hash* satu arah memiliki karakteristik tambahan yang membuatnya satu arah :

```
Diberikan M, mudah menghitung h.
Diberikan h, sulit menghitung M agar H(M) = h.
Diberikan M, sulit menemukan pesan lain, M', agar H(M) = H(M').
```

2.3. Algoritma MD2

Algoritma MD2 dikembangkan oleh Ron Rivest pada tahun 1989. Algoritma ini dioptimasikan dengan mennggunakan komputer 8-bit. MD2 sebenarnya dispesifikasikan dalam RFC 1319. Algoritma MD2 menghasilkan nilai *hash* yang berukuran 128-bit dan menerima input pesan dengan panjang yang tidak ditentukan. Pesan yang akan dijadikan input untuk fungsi *hash* ini akan terlebih dahulu akan dipadding. Untuk kalkulasi yang sebenarnya.

Misalkan kita memiliki sejumlah b-byte pesan sebagai input, dan kita mengharapkan untuk dapat mendapatkan *message digest* dari pesan tersebut. Dalam hal ini, b merupakan suatu bilangan bulat sembarang yang bernilai positif, bisa juga nol, dan besarnya sembarang. Kita nyatakan bahwa byte dari pesan ditulis dalam bentuk :

```
m_0 m_1 . . . m_{b-1}
```

2.4. Algoritma MD4

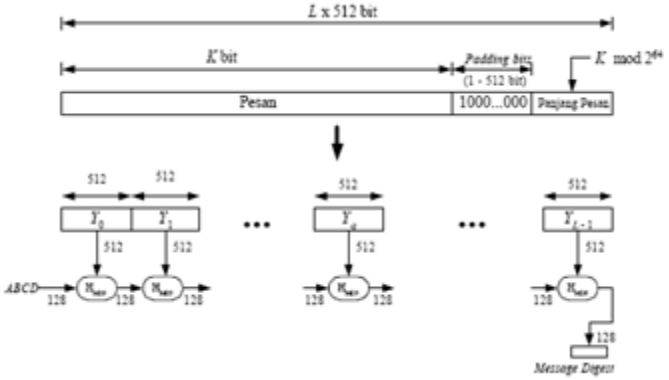
Algoritma MD4 atau *Message Diggest 4* merupakan salah satu seri algoritma Message Diggest yang dibuat oleh Ronald Rivest dari MIT pada tahun 1990, 2 tahun sebelum pembuatan MD5 yang menjadi perbaikannya. Proses fungsi hash MD4 terdiri atas 3 buah putaran dengan masing-masing putaran terdiri atas 16 buah operasi dasar.

Pertama-tama kita misalkan pesan memiliki sejumlah b-bit pesan sebagai input, dan kita menginginkan untuk menghasilkan *message digest* dari pesan tersebut. Dalam hal ini, b merupakan sebuah bilangan bulat positif, mungkin nol, dan bilangan tersebut harus kelipatan dari 8. Kita membagi pesan tersebut sebagai berikut :

```
m_0 m_1 . . . m_{b-1}
```

2.5. Algoritma MD5

MD5 dibuat oleh Ronald Rivest pada tahun 1991. MD5 merupakan fungsi *hash* satu arah yang merupakan perbaikan dari *MD4* setelah *MD4* berhasil ditemukan kelemahannya oleh kriptanalis. Algoritma *MD5* menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit. Gambar skema pembuatan *message digest* diperlihatkan pada gambar di bawah di bawah ini.



Skema Pembuatan *Message Dig*est dengan Algoritma MD5

3. METODE PENELITIAN

Pada bab ini, akan dijabarkan metode yang digunakan di dalam penelitian ini, serta prosedur-prosedur dalam pengumpulan dan analisa data. Dimana data-data yang dipakai di dalam penelitian ini diambil dari berbagai sumber, seperti artikel, *paper*, serta dari internet.

3.1. Metode Analisis

Dalam analisis yang dilakukan dengan cara pengumpulan data, akan dirangkum dan dipelajari sebagai bahan kelanjutan analisa penelitian ini. Selain itu, analisa juga dilakukan dengan mengamati cara kerja *message digest 5*, untuk selanjutnya dibandingkan dengan algoritma pendahulunya, MD2 dan MD4. Dalam pengujian integritas serang *user* dapat membandingkan MD5 *sum* yang dipublikasikan dengan *checksum* dari file yang diambil. Dengan asumsi bahwa *checksum* yang dipublikasikan dapat dipercaya akan keasliannya, seorang *user* dapat secara yakin bahwa file tersebut adalah file yang sama dengan file yang dirilis oleh para *developer,* jaminan perlindungan dari *Trojan Horse* dan virus komputer yang ditambahkan pada perangkat lunak.

3.2. Prosedur Pengumpulan Data

Proses pengumpulan data dalam penelitian ini dapat dijabarkan sebagai berikut:

1. Pesan akan ditambahkan bit-bit tambahan sehingga panjang bit akan kongruen dengan 448 mod 512. Hal ini berarti pesan akan mempunyai panjang yang hanya kurang 64 bit dari kelipatan 512 bit.
2. Setelah penambahan bit, pesan masih membutuhkan 64 bit agar kongruen dengan kelipatan 512 bit. 64 bit tersebut merupakan perwakilan dari b (panjang pesan sebelum penambahan bit dilakukan). Bit-bit ini ditambahkan ke dalam 2 *word* (32 bit) dan ditambahkan dengan *low-order* terlebih dahulu.
3. Pada MD5 terdapat 4 buah *word* (32 bit) register yang berguna untuk menginisialisasi *message digest* pertama kali. Register-register ini diinisialisaskikan dengan bilangan heksadesimal.
4. Hasil dari MD5 adalah 128 bit dari *word* terendah A dan tertinggi *word* D, masing-masing 32 bit.

4. ANALISIS DATA

Berdasarkan hasil dari metode analisis yang telah dilakukan, akan dijabarkan hasil-hasil analisa kriptografi dengan metode MD5, MD2, dan MD4 , dilihat dari cara kerja masing-masing algoritma tersebut.

4.1. Cara Kerja MD2

5 tahap proses untuk menghasilkan *message digest* untuk algoritma *MD2*:

1. Memasukkan *padding byte*
   Pertama-tama dilakukan *padding* pada pesan awal sehingga panjangnya (dalam satuan bit) kongruen dengan 448 modulo 512. Maka pesan tersebut kekurangan 64 bit untuk mencapai kelipatan 512. *Padding* selalu dilakukan sehingga pesan tersebut kongruen dengan 512.

Padding bit awal yaitu bit “1”, selanjutnya ditambahkan bit “0” sehingga pesan tersebut memiliki panjang yang kongruen dengan 448 modulo 512. Jadi, panjang *padding* paling sedikit adalah satu bit hingga 512 bit.

2. Memasukkan *check sum*
   Sebanyak 16 byte *checksum* dari pesan akan ditambahkan pada hasil dari tahap sebelumnya. Pada langkah ini digunakan sebuah 256-byte yang dibangkitkan secara acak yang dibuat dengan nilai digit dari pi. Jika S[i] menotasikan untuk elemen ke-i pada tabel.
   Pseudocode untuk *checksum*:

```
/* kosongkan checksum. */
For i = 0 to 15 do:
Set C[i] to 0
end
/* dari loop i */
Set L to 0
/* Proses tiap blok 16-word.
*/
For i = 0 to N/16-1 do
/* Checksum block i. */
For j = 0 to 15 do
Set c to M[i*16+j]
Set C[j] to S[c xor L]
Set L to C[j]
end
end
```

   selanjutnya 16 *byte checksum* tersebut dimasukkan ke dalam pesan.
3. Inisialisasi penyangga MD
   Sebuah 48-byte penyangga X digunakan untuk menghasilkan *message digest*, nilai penyangga diinisialisasi dengan nol.
4. Proses pesan dalam blok 16 *byte*
   Langkah ini menggunakan angka hasil pembangkitan sebanyak 256 *byte* yang sama, yang dihasilkan pada proses 2.
   Pseudocode untuk proses ini adalah sebagai berikut:

```
/* Proses tiap blok 16-word */
For i = 0 to N’/16-1 do
/* Menyalin blok i pada X */
For j = 0 to 15 do
Set X[16+j] to M[i*16+j]
Set X[32+j] to (X[16+j]
xor
X[j])
end /* dari loop j */
Set t to 0
/* Lakukan 18 round */
For j = 0 to 17 do
/* Round j */
For k = 0 to 47 do
Set t and X[k] to (X[k]
xor
S[t]).
end /* dari loop k */
Set t to (t+j) modulo 256.
end /* dari loop j */
end /* dari loop i */
```

4.2. Cara Kerja MD4

5 tahap proses untuk menghasilkan *message digest* untuk algoritma *MD4*:

1. Memasukkan *padding bit*
   Pertama-tama dilakukan *padding* pada pesan awal sehingga panjangnya (dalam satuan bit) kongruen dengan 448 modulo 512. Maka pesan tersebut kekurangan 64 bit untuk mencapai kelipatan 512. *Padding* selalu dilakukan sehingga pesan tersebut kongruen dengan 512.
   Padding bit awal yaitu bit "1", selanjutnya ditambahkan bit "0" sehingga pesan tersebut memiliki panjang yang kongruen dengan 448 modulo 512. Jadi, panjang *padding* paling sedikit adalah satu bit hingga 512 bit.
2. Memasukkan Panjang Pesan
   Sebuah 64-bit yang direpresentasikan oleh b (panjang pesan awal sebelum dilakukan *padding bit*) dimasukkan pada hasil pesan untuk tahap satu di atas. Panjang pesan yang digunakan selalu lebih kecil dari 264. Bila panjang pesan melebihi 264, maka order terendah yang direpresentasikan oleh 264 yang akan digunakan. Oleh karena itu, setelah proses ini, pesan akan memiliki panjang kelipatan dari 512 bit. Dan pesan dibagi dalam kelipatan 32 bit dalam M[0 . . . N-1] yang menotasikan pesan, dimana N adalah kelipatan dari 16.selanjutnya 16 *byte checksum* tersebut dimasukkan ke dalam pesan.
3. Inisialisasi penyangga MD
   Ada empat peubah penyangga yang digunakan, yaitu *A, B, C, D* yang digunakan untuk menghasilkan *message digest*. Masing-masing variable ini merepresentasikan sebuah nilai 32-bit register. Register tersebut diinisialisasi dengan nilai dalam bentuk heksadesimal, dalam order terendah terlebih dahulu) :

```
A = 01 23 45 67
B = 89 ab cd ef
C = fe dc ba 98
D = 76 54 32 10
```

Inisialisasi Penyangga

4. Proses pesan dalam blok 32 bit
   Untuk tahap ini pertama-tama di definisikan terlebih dahulu fungsi-fungsi pelengkap yang dibutuhkan untuk menghasilkan *message digest*, Terdapat tiga fungsi pelengkap yang dalam hal ini tiap-tiap fungsi ini menerima input 32-bit dan menghasilkan output 32 bit juga.

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$
$$G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$

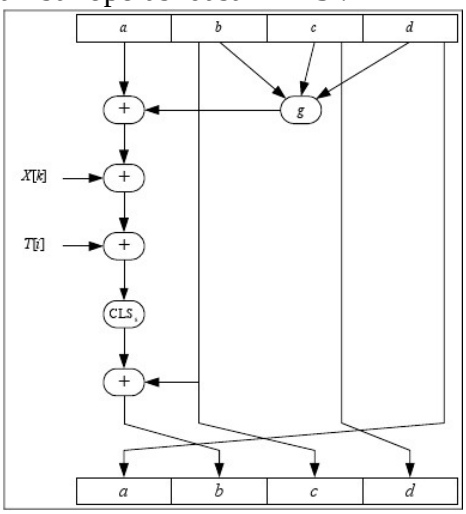
Inisialisasi Penyangga

   Dalam tiap bit posisi F pada fungsi berlaku hubungan kondisional berikut : if X then Y else Z. Fungsi F seharusnya didefinisikan menggunakan atau tanpa v selama XY dan not(X)Z tidak memiliki bit "1" pada posisi yang sama. Dalam tiap bit posisi G berlaku hubungan berikut : jika paling tidak dua dari X, Y, Z sedang digunakan, maka G memiliki sebuah bit "1" dalam posisi tersebut, dan jika tidak G memiliki sebuah bit "0". Ini menarik untuk diperhatikan bahwa jika

bit-bit X, Y, dan Z saling bebas tidak saling mempengaruhi satu sama lain, tiap-tiap bit dari f(X,Y,Z) akan menjadi saling bebas pula. Fungsi H merupakan operasi bit XOR atau fungsi *parit*, dengan atribut yang mirip dengan F dan G.

4.3. Cara Kerja MD5

Cara kerja kriptografi algoritma MD5 adalah menerima input berupa pesan dengan ukuran sembarang dan menghasilkan *message diggest* yang memiliki panjang 128 bit. Berikut ilustrasi gambar operasi dasar MD5 :



Operasi Dasar MD5
dengan Pergeseran Penyangga ke Kanan Secara Sirkuler

Operasi dasar MD5 yang diperlihatkan gambar diatas dapat dituliskan dengan persamaan berikut ini :

$$a \leftarrow b + \mathrm{CLS}_s(a + g(b, c, d) + X[k] + T[i])$$

Dimana

a, b, c, d = empat buah peubah penyangga 32-bit (berisi nilai penyangga A, B, C, D)

g = salah satu fungsi F, G, H, I

CLS_s = *circular left shift* sebanyak s bit

$X[k]$ = kelompok 32-bit ke-k dari blok 512 bit *message* ke-q. Nilai $k = 0$ sampai 15.

$T[i]$ = elemen Tabel T ke-i (32 bit)

$+$ = operasi penjumlahan modulo 2^{32}

4.4. Percobaan MD2, MD4 dan MD5 Menggunakan Perangkat Lunak

Untuk menguji coba MD5, MD2, dan MD4, maka dilakukan percobaan dengan bermacam-macam input.

- *Test case -1:*
  Digunakan dengan mencoba *input string "hello World"* pada ketiga algoritma
  Input: *hello World*
  MD2: d9cce882ee690a5c1ce70beff3a78c77
  MD4: aa010fbc1d14c795d86ef98c95479d17
  MD5: 5eb63bbbe01eeed093cb22bb8f5acdc3

- *Test case -2:*
  Pada percobaan ini dilakukan dengan mencoba sebuah *string* kosong pada ketiga algoritma. Dapat dilihat pada hasilnya ketiga algoritma ini menghasilkan nilai *hash* yang berbeda untuk sebuah *string* kosong.
  Walaupun nilai inisialisasi *variabel* penyangga untuk ketiga algoritma sama.
  Input:
  MD2: 8350e5ae24c153df2275c9f80692773
  MD4: 31d6cfe0d16ae931b73c59d7e0c089c0
  MD5: d41d8cd98f00b204e9800998ecf8427e
- *Test case -3:*
  Tes ini akan mencoba untuk menginputkan sebuah *string "test"* pada ketiga algoritma.
  MD2: dd34716876364a02d0195e2fb9ae2d1b
  MD4: db346d691d7acc4dc2625db19f9e3f52
  MD5: 098f6bcd4621d373cade4e832627b4f6

4.5. Implementasi MD5 Pada Database

Langkah-langkah di dalam mengimplementasikan metode MD5 dilakukan dengan menggunakan 2 buah file PHP yang berfungsi sebagai file form inputan dan koneksi *database. Database* yang digunakan untuk implementasi ini adalah MySQL, dengan alasan bahwa MD5 kriptografi merupakan bagian yang terintegrasi dengan MySQL. Dengan menggunakan *syntax:*

```
$encripted_password=md5{$password}
```

Maka *password* pada *database* akan terenkripsi dengan sendirinya.

Berdasarkan beberapa kali implementasi yang dilakukan, dapat ditemukan kelemahan dari algoritma MD5, dimana dalapm pengaplikasiannya, apabila seorang *user* ingin melihat datanya sendiri, didapatkan bahwa data tersebut masih terenkripsi. Akan tetapi dapat ditarik kesimpulan singkat bahwa dengan menggunakan algoritma MD5 akan lebih terjamin keamanannya dibandingkan dengan pemakaian algoritma MD@ atau MD4.

5. KESIMPULAN DAN SARAN

Pada akhir dari penelitian ini, dapat ditarik beberapa kesimpulan dan saran yang dapat dilakukan dalam rangka mengembangkan hasil penelitian ini di kelak kemudian hari.

5.1. Kesimpulan

Dari pembahasan pada bab-bab sebelumnya, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Fungsi *hash* adalah fungsi yang bisa digunakan untuk berbagai keperluan. Fungsi *hash* dalam kriptografi memiliki beberapa sifat tambahan yang dapat digunakan dalam pengamanan data. Jika dalam *database* fungsi *hash* digunakan untuk memudahkan penyimpanan *key* dalam tabel *hash*, pada kriptografi, fungsi *hash* digunakan untuk memastikan kebenaran pesan yang dikirim dengan cara membandingkan nilai-nilai *hash* yang diperoleh.

2. Untuk kompleksitas waktu algoritma MD4 yang memang di desain untuk meningkatkan performansi kecepatan, memiliki hasil yang lebih baik. Namun demikian dalam segi keamanan MD4 lebih buruk dibandingkan dua algoritma lainnya.
3. *Message Digest 5* (MD5) adalah sebuah fungsi *hash* satu arah yang mengubah masukan dengan panjang variabel menjadi keluaran dengan panjang tetap yaitu 128 bit.
4. Algoritma MD5 merupakan pengembangan lebih lanjut dari MD4.
5. *Simplicity*, algoritma MD5 mudah untuk diimplementasikan karena tidak membutuhkan program yang besar dan panjang.
6. Kecepatan enkripsi pada sistem kriptografi MD5 sangat bergantung kepada spesifikasi komputer yang digunakan.
7. MD5 akan menghasilkan *output* berupa 4 buah blok yang masing-masing terdiri atas 32 bit sehingga menjadi 128 bit yang disebut sebagai nilai *hash*.
8. Untuk memenuhi kebutuhan terhadap algoritma *hash* dalam berbagai aplikasi masih cukup dapat dipercaya.

5.2. Saran

Adapun saran dari penelitian ini dapat dijabarkan sebagai berikut:

1. Bentuk-bentuk *signature* dari algoritma MD5 tidak beresiko, sehingga kedepannya dapat dimanfaatkan pada berbagai macam aplikasi di masa yang akan datang.
2. MD4 sebaiknya tidak digunakan, dikarenakan maasih banyak resiko dalam penyandiaannya.
3. Bentuk-bentuk *signature* dari algoritma MD2 tidak begitu beresiko. Tetapi algoritma ini tidak dapat dijadikan rekomendasi untuk aplikasi-aplikasi dimasa mendatang.

6. DAFTAR PUSTAKA

Kaliski, B, *The MD2 Message-Digest Algorithm*, RSA Laboratories, 1992.

Kurniawan, Yusuf. *Kriptografi, Keamanan Internet dan Jaringan Komunikasi: text*. Informatika 2004

Munir, Rinaldi. *Kriptografi: text.* Informatika. 2006

Rivest, Ron. *RFC 1320 – The MD4 Message-Digest Algorithm*, MIT Laboratory for Computer Science and RSA Data Security, Inc, 1992

http://www.ilmu-komputer.net, *keamanan informasi dan kriptografi* (diakses pada: 20 Oktober 2010, pukul 18.25 WIB)

http://www.ilmu-komputer.net, *MD5 and 1 hash function cryptography* (diakses pada: 20 Oktober 2010, pukul 19.05 WIB)

http://www.id2.php.net/manual /en/function.md5_md4_md2.php (diakses pada: 21 Oktober 2010, pukul 17.30 WIB)