PENGUJIAN INTEGRITAS DATA MENGGUNAKAN ALGORITMA MD5

Rendi Gayu Buana [1)]
Program Studi Teknik Informatika STMIK PPKIA Pradnya Paramita [1)]
Jl. Laksda. Adi Sucipto No. 249-A Malang
Telp (0341) 412699

ABSTRACT

Often the integrity of the data did not change as the original caused by many things such as due to a virus and modified by others, and most people do not pay attention to the integrity of the data so that the data is not complete as the original, then the data can not be used. Thus this study to test the integrity of the data using the MD5 algorithm.
The methods in scientific writing using the MD5 algorithm, which is one of the cryptographic functions that can generate a unique code of a file or files.
MD5 can detect changes in the contents of a file even if the change is very small. Changes can be made up of addition, subtraction or change of the contents of a file. Speed MD5 generate code that is the larger the file size the longer the generated code MD5.

***Key words**: Testing, Integrity, Data, Algorithms, MD5.*

.

PENDAHULUAN

Seiring dengan berkembangnya teknologi dan informasi dalam kehidupan kita saat ini telah memberikan kemudahan bagi kita untuk memperoleh maupun menyajikan data dan informasi dalam berbagai bentuk. Perolehan dan penyajiannya pun dapat dilakukan dalam waktu yang singkat. Data dan informasi juga dapat kita kemas dalam bentuk yang lebih kompak (tidak membutuhkan tempat yang besar) dan *mobile* (mudah dipindah dan dipertukarkan). Penggunaan data dan informasi secara bersama (*data and information sharing*) dan pertukarannya, tidak lagi dibatasi oleh ruang dan waktu. Teknologi internet memungkinkan bagi semua orang diseluruh dunia, untuk mengakses dan berbagi data dan informasi, kapanpun dan dimanapun dia berada.

Seringkali keutuhan data berubah tidak seperti pada aslinya yang disebabkan oleh berbagai hal diantaranya disebabkan terserang virus dan juga diubah oleh orang lain, dan kebanyakan orang tidak memperhatikan keutuhan data tersebut sehingga data tidak utuh seperti pada aslinya, maka data tersebut tidak dapat digunakan dengan semestinya.

Dengan semakin pentingnya informasi dan ditunjang oleh kemajuan pengembangan *software*, menarik minat para pembobol (*hacker* / *cracker*) dan penyusup (*intruder*) mencari kelemahan dari suatu konfigurasi sistem informasi. Secara umum, masalah keamanan di *Internet* dapat dipandang dari dua sisi penting. Sisi pertama adalah integritas data yang dikirimkan melalui jaringan *Internet* dan sisi berikutnya adalah tingkat keamanan dalam jaringan komputer itu sendiri. Beberapa

ancaman terhadap keamanan integritas data dalam Internet, yaitu *modification* dan *fabrication*. *Modification* merupakan ancaman terhadap integritas, yaitu: orang yang berhasil mendapatkan akses informasi dari dalam sistem komputer dan dapat melakukan perubahan terhadap informasi, contohnya merubah program dan lain-lain. *Fabrication* merupakan ancaman terhadap integritas, yaitu orang yang tidak berhak meniru atau memalsukan suatu objek ke dalam sistem, contohnya adalah dengan menambahkan suatu record ke dalam *file*.

Integritas data menjadi sangat penting di kala proses download data atau informasi bisa menjadi ancaman besar terhadap sistem komputer. Untuk menjaga integritas data tetap terjaga, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak. Bentuk-bentuk manipulasi yang dapat dilakukan antara lain penyisipan, penghapusan dan pensubstitusian data lain ke dalam data yang sebenarnya.

Algoritma MD5 (*Message Digest 5*) menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit (Munir, 2006:220). MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah data.

Berdasarkan latar belakang tersebut, penelitian ini mengangkat tema mengenai **Pengujian Integritas Data Menggunakan Algoritma MD5.** Hasil dari penelitian ini dapat tercipta pengujian yang dapat digunakan untuk mengetahui keutuhan dan keaslian data.

METODE PENELITIAN

Adapun metode-metode penelitian yang digunakan adalah sebagai berikut :

- Analisa yaitu dari permasalahan yang ada atau objek dibuat pemodelan yang nanti digunakan dalam pembuatan program.
- Desain yaitu setelah melakukan pemodelan selanjutnya dilakukan perancangan sistem sesuai dengan pemodelan perancangan yang sudah ditetapkan sebelumnya.
- Implementasi yaitu desain yang telah dibuat direpresentasikan kedalam kode-kode pemrograman berdasarkan perancangan

KAJIAN TEORI

Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi. Sedangkan menurut Ariyus (2006:9), berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua kripto dan graphia, kripto berarti *secret* (rahasia) dan graphia berarti *writting* (tulisan). Menurut teminologinya kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ketempat yang lain.

Integritas Data

Integritas data (*data integrity*), adalah layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman. Dengan kata lain, aspek keamanan ini dapat diungkapkan sebagai pertanyaan:"Apakah pesan yang diterima masih asli atau tidak mengalami perubahan (modifikasi)?". Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi pesan oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam pesan yang sebenarnya. Di dalam kriptografi, layanan ini direalisasikan dengan menggunakan tanda-tangan digital (*digital signature*). Pesan yang telah ditandatangani menyiratkan bahwa pesan yang dikirim adalah asli (Munir,
2006:9).

Analisis MD5

Langkah-langkah pembuatan message digest 5 adalah sebagai berikut:

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

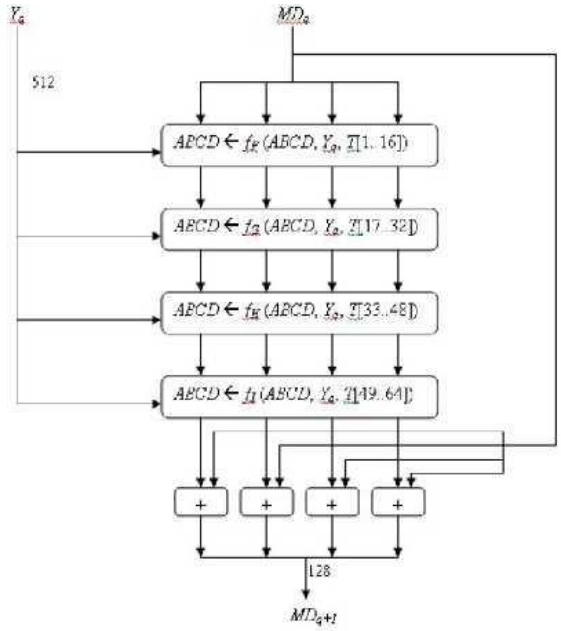
Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. Angka 512 ini muncul karena MD5 memproses pesan dalam blok-blok yang berukuran 512. Pesan dengan panjang 448 bit pun tetap 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 > 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 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:



Gambar 1 Inisialisasi Penyangga MD5

Proses H_{MD5} terdiri dari 4 buah putaran, dan masing-masing putaran melakukan operasi dasar MD5 sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen *T*. Jadi setiap putaran memakai 16 elemen Tabel *T*. Pada gambar diatas, Y_q menyatakan blok 512-bit ke-q dari pesan yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. *MDq* adalah nilai *message digest* 128-bit dari proses H_{MD5} ke-q. Pada awal proses, *MDq* berisi nilai inisialisasi penyangga MD.

PEMBAHASAN

Fungsi Hash

Hash adalah suatu teknik "klasik" dalam ilmu komputer yang banyak digunakan dalam praktek secara mendalam. *Hash* merupakan suatu metode yang secara langsung mengakses *record-record* dalam suatu tabel dengan melakukan transformasi aritmatik pada *key* yang menjadi alamat dalam tabel tersebut. *Key* merupakan suatu input dari pemakai dimana pada umumnya berupa nilai atau *string* karakter.

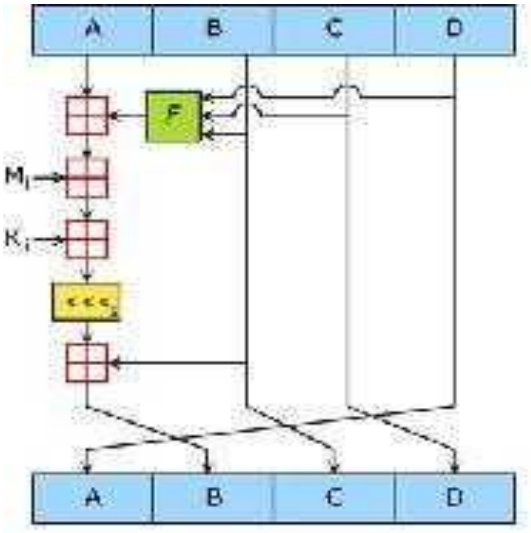
Pelacakan dengan menggunakan *Hash* terdiri dari dua langkah utama, yaitu:

- Menghitung fungsi *Hash*. Fungsi *hash* adalah suatu fungsi yang mengubah *key* menjadi alamat dalam tabel. Fungsi hash memetakan sebuah key ke suatu alamat dalam tabel. Idealnya, *key-key* yang berada seharusnya dipetakan ke alamat-alamat berbeda juga. Pada kenyataannya, tidak ada fungsi *hash* yang sempurna. Kemungkinan besar yang terjadi adalah dua atau lebih *key* yang berada dipetakan kealamat yang sama dalam tabel. Peristiwa ini disebut dengan *collision* (tabrakan). Karena itulah diperlukan langkah berikutnya, yaitu *collision resolution* (pemecahan tabrakan).
- *Collision resolution. Collision resolution* merupakan proses untuk menangani kejadian atau lebih *key* di-*hash* ke alamat yang sama. Cara yang dilakukan jika terjadi *collision* adalah mencari lokasi yang kosong dalam tabel hash secara terurut. Cara lainnya adalah menggunakan fungsi hash yang lain untuk mencari lokasi kosong tersebut.

Cara Kerja MD5

- MD5 mengolah blok 512 bit, dibagi kedalam 16 sub blok berukuran 32 bit.

Keluaran algoritma diset menjadi 4 blok yang masing-masing berukuran 32 bit yang setelah digabungkan akan membentuk nilai hash 128 bit.



Gambar 2 Cara Kerja MD5

MD5 terdiri atas 64 operasi, dikelompokkan dalam empat putaran dari 16 operasi.

F : Adalah fungsi nonlinear, satu fungsi digunakan pada tiap-tiap putaran.

M_i : Menunjukkan blok 32-bit dari masukan pesan, dan

K_i : Menunjukkan konstanta 32-bit, berbeda untuk tiap-tiap operasi.

$<<<_s$: menunjukkan perputaran bit kiri oleh s; s bervariasi untuk tiap-tiap operasi. ⊞ : Menunjukkan tambahan modulo 2^{32}

Berikut dapat dilihat satu buah operasi MD5 dengan operasi yang dipakai sebagai contoh adalah FF(a,b,c,d,Mj,s,ti) menunjukkan a = b ((a + F(b,c,d) + Mj + ti) <<<s). bila Mj menggambarkan pesan ke-j dari sub blok (dari 0 sampai 15) dan <<<s menggambarkan bit akan digeser ke kiri sebanyak s bit, maka keempat operasi dari masing-masing ronde adalah:

FF(a,b,c,d,Mj,s,ti) menunjukkan a = b + ((a + F(b,c,d) + Mj + ti) <<<s) GG(a,b,c,d,Mj,s,ti) menunjukkan a = b + ((a + F(b,c,d) + Mj + ti) <<<s) HH(a,b,c,d,Mj,s,ti) menunjukkan a = b + ((a + F(b,c,d) + Mj + ti) <<<s) II(a,b,c,d,Mj,s,ti) menunjukkan a = b + ((a + F(b,c,d) + Mj + ti) <<<s) Konstanta ti didapat dari integer 2^{32}. abs(sin(i)), dimana i dalam radian.

Setiap pesan yang akan dienkripsi, terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan. Kita anggap sebanyak b bit. Disini b adalah bit non negatif integer, b bisa saja nol dan tidak harus selalu kelipatan delapan. Pesan dengan panjang b bit dapat digambarkan sebagai berikut :

m_0 m_1 ..... m_(b-1)

Langkah-langkah pembuatan message digest dengan MD5 seperti yang disampaikan oleh Munir (2006:220-224), adalah sebagai berikut:

- 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 ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512.

- 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 > 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

- Inisialisasi penyangga (buffer) MD.

MD5 membutuhkan 4 buah penyangga yang masing-masing panjangnya 32bit.

Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut:

A = 67452301
B = EFCDAB89
C = 98BADCFE
D = 10325476

- 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}. Proses H_{MD5} terdiri dari 4 putaran, dan masing-masing putaran melakukan 16 kali operasi dasar MD5 dan tiap operasi dasar memakai sebuah elemen T yang didapatkan dari perhitungan fungsi $T(i)=2^{32}$ x abs(sin(i)) dimana i dalam radian. Berikut adalah tabel fungsi untuk tiap putaran MD5.

- Keluaran MD5

Keluaran dari MD5 adalah 128 bit dari *word* terendah A dan tertinggi *word* D masing-masing 32 bit.

MD5 memproses variasi panjang pesan kedalam keluaran 128-bit dengan panjang yang tetap. Pesan masukan dipecah menjadi dua gumpalan blok 512-bit, pesan ditata hingga panjang pesan dapat dibagi 512. penataan bekerja sebagai berikut: bit tunggal pertama, 1, diletakkan pada akhir pedan. Proses ini diikuti dengan serangkaian nol (0) yang diperlukan agar panjang pesan lebih dari 64-bit dan kurang dari kelipatan 512. Bit-bit sisa diisi dengan 64-bit integer untuk menunjukkan panjang pesan yang asli. Sebuah pesan selalu ditata setidaknya dengan 1-bit tunggal, seperti jika panjang pesan adalah kelipatan 512 dikurangi 64-bit untuk informasi panjang (panjang mod(512) = 448), sebuah blok baru dari 512-bit ditambahkan dengan 1-bit diikuti dengan 447 bit-bit nol (0) diikuti dengan panjang 64-bit.

Algoritma MD5 yang utama beroperasi pada kondisi 128-bit, dibagi menjadi empat *word* 32-bit menunjukkan *A,B,C* dan *D*. Operasi tersebut diinisialisasi dijaga untuk tetap konstan. Algoritma utama kemudian beroperasi pada masing-masing blok pesan 512-bit, masing-masing blok melakukan pengubahan terhadap kondisi. Pemrosesan blok pesan terdiri atas empat tahap, batasan *putaran*; tiap putaran membuat 16 operasi serupa berdasar pada fungsi non-linear *F*, tambahan modular, dan

rotasi ke kiri. Gambar satu mengilustrasikan satu operasi dalam putaran. Ada empat macam kemungkinan fungsi *F,* berbeda dari yang digunakan pada tiap-tiap putaran:

F(X,Y,Z) = (X∧Y)∨(¬X∧Z)

G (X,Y,Z)

(X∧Z)∨(Y∧¬Z) H(X,Y,Z)

= (X⊕Y⊕Z) I(X,Y,Z) = Y⊕(X∨¬Z)

dan operasi XOR, AND, OR, dan NOT adalah sebagai berikut :

⊕,∧,∨,¬

Kelebihan MD5

Kadang-kadang kita menginginkan isi arsip tetap terjaga keasliannya (tidak diubah oleh orang yang tidak berhak). Perubahan kecil pada isi arsip sering tidak terdeteksi, khususnya pada arsip yang berukuran besar. Fungsi *hash* dapa digunakan untuk menjaga keutuhan (integritas) data. Caranya, bangkitkan *message digest* dari isi arsip dengan menggunakan algorit*ma MD5. message digest* dari arsip tersebut dapat disimpan dalam basis data. Verifikasi isi arsip dapat dilakukan secara berkala dengan membandingkan *message digest* dari isi arsip sekarang dengan *message digest* dari arsip asli (yang disimpan di dalam basis data). Jika terjadi perbedaan, maka disimpulkan ada modifikasi terhadap isi arsip (atau terhadap *message digest* yang disimpan). Aplikasi ini didasarkan pada kenyataan bahwa perubahan 1 bit pada pesan akan mengubah secara rata-rata setengah dari bit-bit *message digest.* Dengan kata lain, fungsi *hash* sangat peka terhadap perubahan sekecil apapun pada data masukan.

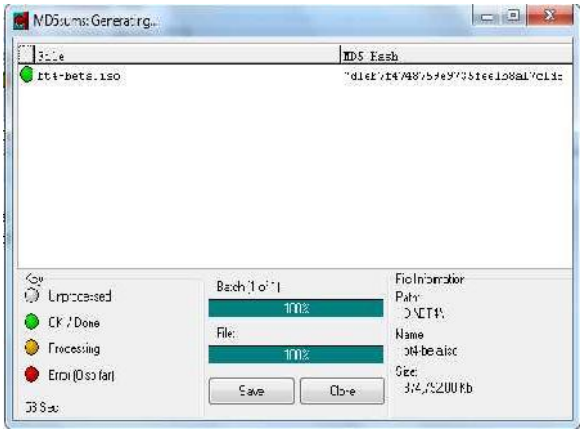
Pseudocode MD5

```
//Catatan: Semua peubah adalah 32-bit dan penjumlahan modulo 2^32 saat melakukan perhitungan
//Mendefinisikan r sebagai berikut
var int[64] r, k
r[ 0..15] := {7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22}
r[16..31] := {5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20}

r[32..47] := {4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23}
r[48..63] := {6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21}
//Menggunakan bagian fraksional biner dari integral sinus sebagai konstanta:
for i from 0 to 63
k[i] := floor(abs(sin(i + 1)) × 2^32)
//Inisialisasi variabel:
var int h0 := 0×67452301
var int h1 := 0xEFCDAB89
var int h2 := 0x98BADCFE
var int h3 := 0×10325476
//Pemrosesan awal:
append "1" bit to message
append "0" bits until message length in bits ≡ 448 (mod 512)
append bit length of message as 64-bit little-endian integer to message
//Pengolahan pesan pada kondisi gumpalan 512-bit:
Bagi pesan ke dalam blok-blok berukuran 512-bit.
Bagi tiap blok ke dalam 16 buah sub-blok 32-bit, w(i), 0 ≤ i ≤ 15
//Inisialisasi nilai hash pada gumpalan ini:
```

```
var int a := h0
var int b := h1
var int c := h2
var int d := h3
//Kalang utama:
for i from 0 to 63
  if 0 ≤ i ≤ 15 then
     f := (b and c) or ((not b) and d)
     g := i
  else if 16 ≤ i ≤ 31
     f := (d and b) or ((not d) and c)
     g := (5×i + 1) mod 16
  else if 32 ≤ i ≤ 47
     f := b xor c xor d
     g := (3×i + 5) mod 16
  else if 48 ≤ i ≤ 63
     f := c xor (b or (not d))
     g := (7×i) mod 16
  temp := d
temp := d
d := c
c := b
b := ((a + f + k(i) + w(g)) leftrotate r(i)) + b
a := temp
//Tambahkan hash dari gumpalan sebagai hasil:
h0 := h0 + a
h1 := h1 + b
h2 := h2 + c
h3 := h3 + d
var int digest := h0 append h1 append h2 append h3
```

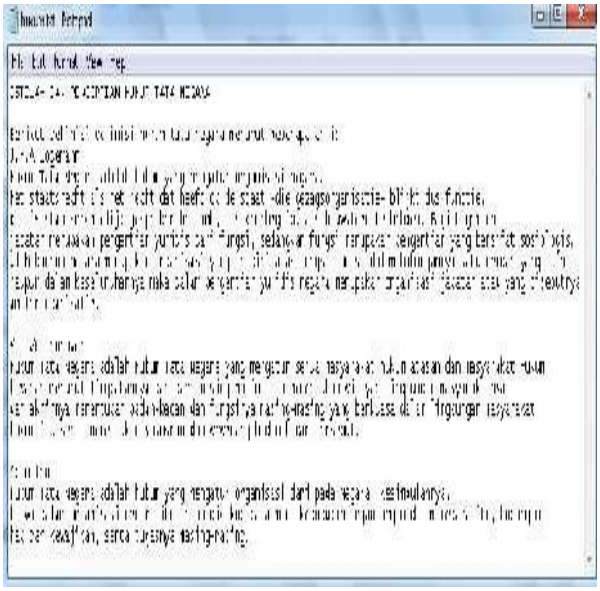
Implementasi MD5 Pada Pengujian Integritas Data



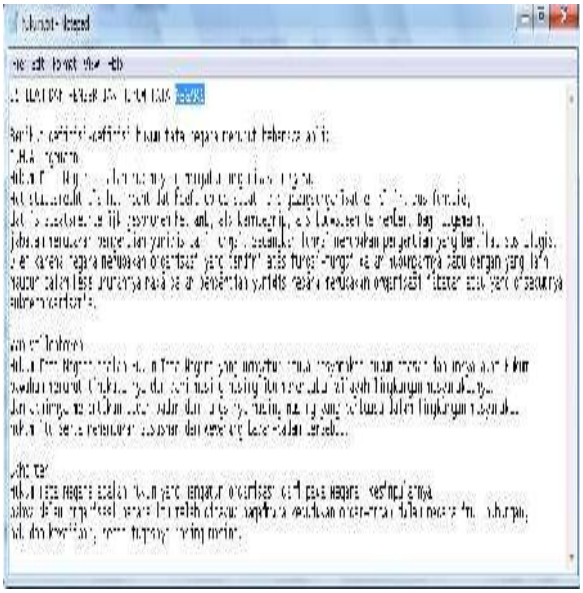
Gambar 3 Pengujian Integritas Data

Dari gambar 3 dapat dilihat bahwa verifikasi terhadap data yang sebelumnya telah diunduh dari *ftp://repo.ugm.ac.id/iso/backtrack/bt4/* dengan kode *MD5* (7d1eb7f4748759e9735fee1b8a17c1d8) yang diberikan berhasil dilakukan. Ini berarti data yang telah diunduh tersebut tidak mengalami kerusakan atau perubahan dikarenakan kode MD5 tetap.

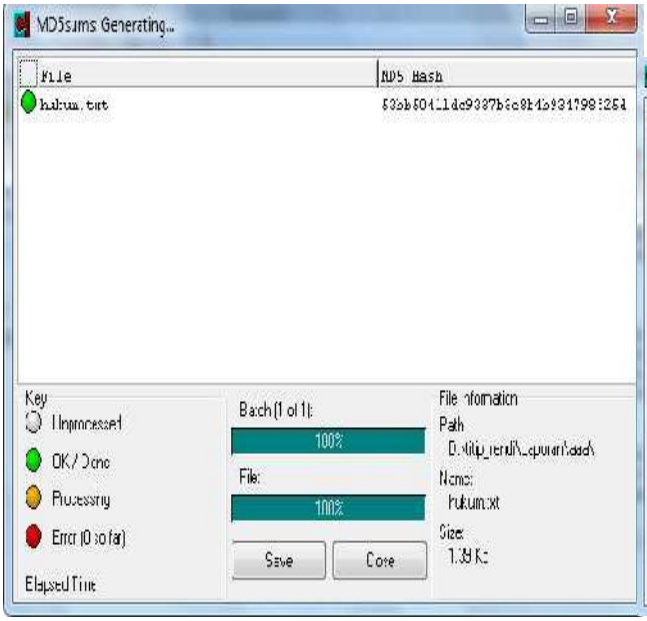
Pengujian berikutnya adalah untuk menguji sensitifitas dari perubahan data. Proses ini membandingkan dua buah data yang sama dengan perubahan yang sangat kecil. Gambar berikut menunjukkan data yang masih asli dengan *hash MD5* (53bb50411dc9337b3e8b4b934798f25d) yang telah dibangkitkan:



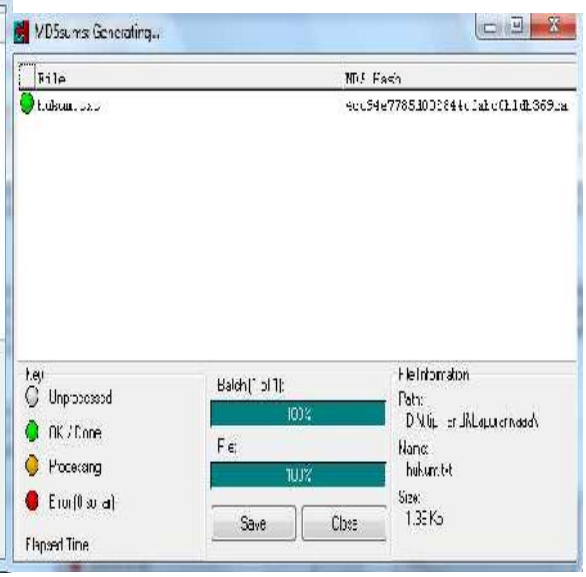
Gambar 4 Data Dengan *hash* MD5



Gambar 6 Data dengan perubahan pada salah satu kata



Gambar 5 Form Pembangkitan data dengan MD5



Gambar 7 Form Pembangkitan data dengan MD5

Pada percobaan pertama dilakukan pengujian dengan melakukan perubahan pada salah satu kata. Gambar berikut adalah data yang telah mengalami perubahan pada salah satu kata dengan *hash MD5* yang telah dibangkitkan:

Dari dua data yang isinya hampir sama dengan perubahan yang sangat kecil dimana pada data yang kedua perubahan terjadi pada kata “NEGARA”, diperoleh kode *MD5* yang berbeda. *Message digest* yang dihasilkan adalah:

Sebelum diubah : *MD1* = 53bb50411dc9337b3e8b4b934798f25d

Sesudah diubah : *MD2* = 4cd94e7785d002844cfabc0b1db369ca

Verifikasi : *MD1* ≠ *MD2*

Kesimpulan : *File* telah diubah.

KESIMPULAN DAN SARAN

Kesimpulan yang diambil dari pembahasan terdahulu adalah sebagai berikut :

a. MD5 dapat mendeteksi perubahan yang terjadi pada suatu file walaupun perubahan tersebut sangat kecil. Perubahan dapat terdiri dari penghapusan, penggantian atau penambahan dari suatu file.
b. *MD5* dapat membangkitkan kode dari segala format file termasuk zip, exe, iso dan lain-lain.
c. *MD5* merupakan fungsi *hash* satu arah dimana kode yang telah dibangkitkan sangat sulit bahkan tidak mungkin untuk dikembalikan ke kode sumber awalnya. Hal ini menjadi kelebihan dari *MD5* dalam aplikasi keamanan data termasuk integritas data.

Saran

- Penelitian dengan MD5 ini dapat dikembangkan dengan bahasa pemrograman lain yang dapat bermanfaat bagi keamanan data seperti: pemrograman java, visual basic.net dan lain-lain.
- MD5 dapat dikombinasikan dengan metode-metode lain untuk lebih memperkuat keamanan integritas data seperti: SHA-1 dan lain-lain.

DAFTAR PUSTAKA

Ariyus, Dony. 2006. ***Kriptografi Keamanan Data dan Komunikasi.*** Yogyakarta: Graha Ilmu.

Febriani. 2005 (online) (*http://gunadarma.ac.id* Diakses 10 juli 2011). ***Flowchart,*** *SUMBER http://febriani.staff.gunadarma.ac.id/Downloads/files/5616/Flowchart.pdf*

Kadir, Abdul. 2003. ***Pengenalan Sistem Informasi.*** Yogyakarta: Andi.

Kurniawan, Yusuf. 2004. ***Kriptografi Keamanan Internet dan Jaringan Telekomunikasi***. Bandung Informatika.

Masyarakat Digital Gotong Royong (MDGR). 2003-2006 ***Pengantar Sistem Operasi Komputer : plus ilustrasi kernel,*** (online) (. Diakses 11 juni 2011). *SUMBER .v06/Kuliah/SistemOperasi/BUKU/SistemOperasi-4.0.pdf*

Misky, Dudi. 2005. ***Kamus Informasi & Teknologi.*** Jakarta: EDSA Mahkota.

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

Pascoe, Luke. 2003. ***MD5summer,*** (online) (*http://md5summer.org* Diakses 2 juli 2011) *SUMBER http://md5summer.org/source/Source_md5v12005.zip*

Taufik, Muhammad. 2011. ***Kriptografi Password dengan MD5.*** Sekolah Tinggi Teknologi Nurul Jadid, Probolinggo, 2011.

Williams, Laurie. 2006. ***Testing Overview***

and Black-Box Testing Techniques, (online) (Diakses 10 juli 2011) *SUMBER http://docs.google.com/viewer?a=v&q=cache:pl-padW3eKgJ:agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf+filetype:pdf+blackbox+testing&hl=en&pid=bl&srcid=ADGEESiDc6UdTl2b9VX2B4wNuJIwjGawo7xRMb29jbbgcT6iTCGVkI13Lj9xfUGuu1LAzaOPIUyhPHnQ-czSCNgDe9T9GmdsrZVrkbS_ahbEoR_FiiWbBe-sXv-xXxbP0MVtjHTWpyVK&sig=AHIEtbS5MUGIFoYAk3t-E-pMtK4YrtVxJw*