Pengecekan Dokumen – Dokumen Digital Penting Dengan SHA

Yogi Adytia Marsal 13508016
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
adyt_e6_aja@itb.ac.id

***Abstract*—Hash adalah metoda dalam kriptogrfi untuk merahasiakan sebuah data. Namun hash adalah fungsi yang asimetri, yaitu tidak bisa di kembalikan lagi kebentuk semulanya. Namun fungsi ini sangat berguna dalam dunia digital karena bisa merahasiakan password, mengecek data yang berubah dan di gunakan untuk tanda tangan digital. Pada fungsi hash, data dalam ukuran apapun baik itu dari ukuran yang 0 hingga ukuran terbesar data bisa di hashkan menjadi 160 bit. Perubahan satu huruf saja bisa mengakibatkan nilai hashnya berbeda sangat jauh, Sehingga data yang di hashkan sulit untuk ditebak.**

Dengan kemampunan fungsi ini maka semua file bisa di ambil hashnya dengan fungsi SHA. Dengan demikian penulis memanfaatkan kemampuan fungsi ini untuk mengecek perubahan yang terjadi pada dokumen – dokumen penting. Satu huruf saja pada dokumen tersebut berubah maka nilai hash yang di hasilkan akan jauh berubah.

***Index Terms*—file, Hash, SHA-1**

I. PENDAHULUAN

Pada zaman ini banyak orang yang lebih memilih data ataupun dokumen – dokumen penting dalam bentuk digital. Hal ini dikarenkan mudah diakses dan tidak memakan tempat yang besar untuk penyimpanannya. Jika dokumen – dokumen tersebut memiliki jumlah yang sangat banyak, pemilik file tersebut tentuknya akan mengalami kesulitan untuk mengontrolnya jika dalam bentuk hardcopy, namun jika dalam bentuk digital sangat mudah di kelola baik itu pencarian, pengeditan ataupun penambahan data. Namun dengan memanfaatkan data dalam bentuk digital, akan sangat mempermudah penggunaan file tersebut, baik itu pencarian, pengeditan ataupun penambahan data.

Data digital tersebut bisa disimpan didirekori sendiri pada komputer dan bisa juga disimpan pada sebuah server. Tentunya di zaman ini internet sudahlah sangat murah dan oleh sebab itu tempat penyimpanan tersebut di sambungkan dengan internet, dengan tujuan bisa di gunakan dimana saja kita berada. Kemudahan akses ini sangat membantu user untuk memanfaatkan dokumen – dokumen tersebut untuk kepentingannya tersebut.

Tetapi dengan kemudahan tersebut membuka peluang bagi orang lain untuk memanfaatkan hal tersebut juga untuk mengacak – ngacak file atau dokumen penting yang kita miliki tersebut. Tentunya file yang menarik untuk diketahui ataupun dirusak atau diubah bukanlah file sembarangan, pastinya file – file penting yang sangat berpengaruh. Dengan kemampuan teknologi hal tersebut bisa saja terjadi, orang lain bisa mengubah dan menghapus data kita dengan seenaknya melalui internet ataupun jaringan komunikasi lainnya. Bahkan terkadang karena bersaing orang bisa saja mengubah langsung data tersebut dari computer kita langsung tanpa sepengetahuan kita.

Oleh karena itu dengan memanfaatkan fungsi Hash penulis berniat untuk membuat program yang memberikan peringatan kepada user apakah data tersebut telah di ubah ataupun di dihapus oleh orang lain.. idenya sangat sederhana. Hanya menyimpan hasil hash dari tiap file yang berada di direktori yang di tentukan oleh user.

II. DASAR TEORI

Fungsi hash adalah fungsi pada kriptografi yang menerima masukan string dalam ukuran berapa saja dan mengkonversinya menjadi string yang panjangnya tetap. Hasil dari fungsi hash ini disebut nilai hash. Dengan kata lain fungsi ini mengkompresi sembarang pasan yang berukuran bebas menjadi nilai hash yang berukuran selalu tetap.

Fungsi hash adalah fungsi dalam kriptografi yang bersifat asimetrik. Jadi setelah di hash kan sebuah string, tidak dapet di kembalikan lagi ke bentuk semulanya. Sifat – sifat fungsi hash satu arah adalah sebagai berikut:

1. Fungsi H dapat diterapkan pada blok data berukuran berapa saja.
2. H menghasilkan nilai (h) dengan panjang tetap (*fixedlength output*).
3. $H(x)$ mudah dihitung untuk setiap nilai x yang diberikan.
4. Untuk setiap h yang dihasilkan, tidak mungkin dikembalikan nilai x sedemikian sehingga $H(x)$ =

h. Itulah sebabnya fungsi *H* dikatakan fungsi *hash* satu-arah (*oneway hash function*).

5. Untuk setiap *x* yang diberikan, tidak mungkin mencari *y* ¹*x* sedemikian sehingga *H*(*y*) = *H*(*x*).
6. Tidak mungkin mencari pasangan *x* dan *y* sedemikian sehingga *H*(*x*) = *H*(*y*

Fungsi hash sudah sangat banyak dibuat oleh orang dengan berbagai cara pembuatan nilai hash yang berbeda – beda, beriku beberapa nama fungsi hash yang telah ada :

1. MD2, MD4,MD5
2. SHA
3. Snefru
4. N-Hash
5. RIPE-MD

Pada kesempatan ini user akan memanfaatkan fungsi hash SHA yang telah di buat pada tugas kriptografi untuk fungsi yang digunakan untuk mengetahui perubahan data yang ada pada dokumen – dokumen penting tersebut.

Prinsip pengecekannya sangat sederhana yaitu menyimpan nilai hash dari setiap dokumen kedalam bentuk file ekternal. Jadi awalnya user setelah selesai menggunakan dokumen – dokumen tersebut harus mengesave atau membuat file eksternal tersebut terlebih dahulu, jika tidak ada perubahan saat program dijalankan untuk pengecekan maka tidak ada file yang berubah. Namun, dikarenakan menggunakan hash, perubahan 1 file saja akan memperlihatkan atau menghasilkan nilai hash yang sangat berbeda jauh oleh karena itu jika ada file yang berubah maka bisa diketahui dimana file tersebut berubahnya.

III. PEMBACAAN FILE

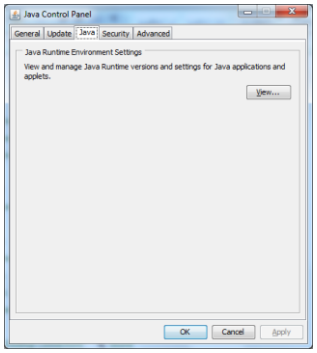
File yang akan dihash kan bisa dalam ukuran apa sajam namun string memiliki batasan tertentu yang akan membatasi ukuran file yang bisa dibaca untuk diambil nilai hashnya, string hanya memiliki panjang sekitar 2,147,483,647 atau jika dalam ukuran file hanya berukuran kurang lebih 2Giga Byte. Pembacaan file tersebut dapat dilihat caranya sebagai berikut, jika kita menggunakan java.

```
// Returns the contents of the file
in // a byte array.
public static byte[]
getBytesFromFile(File file) throws
IOException {

InputStream is = new
FileInputStream(file);

// Get the size of the file
long length = file.length();
```

```
if (length > Integer.MAX_VALUE) {
// File is too large
    }

// Create the byte array to hold
the data
    byte[] bytes = new
byte[(int)length];

// Read in the bytes
int offset = 0;
int numRead = 0;

while (offset < bytes.length
            &&
(numRead=is.read(bytes, offset,
bytes.length-offset)) >= 0)
{
        offset += numRead;
}


// Ensure all the bytes have been
read in
if (offset < bytes.length)
{
   throw new IOException("Could not
completely read file
"+file.getName());
}

// Close the input stream and
return bytes
    is.close();
    return bytes;
}
```

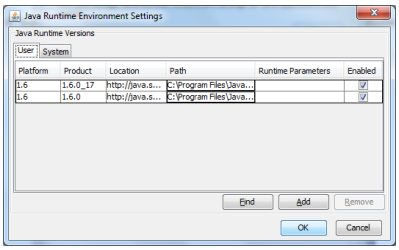
Namun ada sedikit permasalahan yang terjadi saat pembacaan file, file yang dibaca tidak bisa terlalu besar jika file telah lebih besar dari kurang lebih 64 MB maka harus di setting terlebih dahulu memory javanya dengan cara :

1. Klik “Control Panel”
2. Pilih dan klik icon yang bernama “Java” maka akan muncul tampilan seperti berikut :



Gambar 3 tampilan Java Control Panel

3. Lalu pilih tab "java" dan klik tombol "view..." maka akan muncul tampilan sebagai berikut:



Gambar 4 tampilan Java Runtime Environment Setting

4. Lalu setting "Runtime Parameters" sesuai dengan yang dibutuhkan jika ingin menyetting batas maksimum memory java adalah 300MB maka ketikkan di tabel tersebut "`-Xmx300m`" yang artinya maksimum memory adalah 300MB

Walaupun telah disetting seperti itu namun java memiliki juga memiliki batas adalah kurang lebih 1,2 GB dan angka tersebut adalah total ukuran file yang bisa dibuka dalam satu waktu program java di jalankan.

Jika file yang dibaca dalam ukuran kecil tapi banyak itu tidak masalah, namun jika file tersebut dalam ukuran yang besar ini merupakan kasus khusus yang harus diatasi oleh program ini. Dengan mengubah batas memory yang digunakan oleh java maka maksimal kita bisa membaca 1,2 Giga byte namun hal ini akan memlambat kerja java tersebut. Oleh karena itu kita membatasi pembacaan file menjadi beberapa bagian sehingga walaupun ukurannya besar tidak berpengaruh kepada kecepatan program.

Program ini membatasi pembacaan file dengan ukuran 64MB, jadi setiap file yang berukuran lebih dari 64MB akan di potong – potong menjadi beberapa potongan. Dengan kata lain file yang memiliki ukuran lebih dari 64 MB akan memiliki beberapa nilai hash, tergantung kepada berapa banyak potongannya tersebut. Untuk pembacaan file yang lebih besar, penulis menggunakan sintaks berikut ini:

```
Public byte[] readPeacesFile(int
Position, File file){

 RandomAccessFile fp=new
RandomAccessFile(file);

//Seek Position of read
fp.seek(position);
byte[] buff= new  byte[len];

//Read File with length = len = 64MB
fp.read(buff, 0, len);
 //Close read file
 fp.close;

Retrun buff;
}
```

Dengan fungsi ini kita bisa mendapatkan potongan dari file yang nantinya akan kita hash kan. Jadi file akan didapat dalam ukuran 64MB tiap pembacaan jika ukuran file lebih besar. Jika ukuran file bukan kelipatan dari 64MB maka sisanya akan diisi dengan byte null.

IV. FUNGSI HASH

SHA (*Secure Hash Algorithm*) adalah sebuah standar fungsi *hash* satu arah. Fungsi *hash* satu arah adalah fungsi yang bekerja dalam satu arah; menghasilkan string yang tidak dapat diubah kembali menjadi pesan asli. SHA telah mengalami banyak perkembangan, dimulai dari rilisnya SHA-0 pada tahun 1993 hingga dipublikasikannya SHA-512. Namun, fungsi *hash* SHA yang paling umum digunakan adalah SHA-1 yang telah diimplementasikan di dalam berbagai aplikasi dan protokol keamanan seperti TSS, SSL, PGP, SSH, S/MIME, dan Ipsec.

Keluaran fungsi *hash* sering disebut sebagai nilai *hash* atau *message digest* (MD). Fungsi *hash* satu arah harus memenuhi sejumlah kriteria berikut :

1. *Preimage resistant*, yaitu tidak mungkin menemukan pesan masukan berdasarkan sebuah *message digest*.
2. *Second preimage resistant*, yaitu tidak mungkin menemukan dua masukan berbeda yang dapat menghasilkan *message digest* yang sama.
3. *Collision resistant*, yaitu tidak mungkin menemukan dua pesan masukan dengan nilai *hash* yang sama.
4. Fungsi *hash* mudah dihitung.
5. Panjang *message digest* tetap.
6. Fungsi *hash* dapat diterapkan paada pesan masukan dengan panjang sebarang.

Meskipun fungsi *hash* satu arah harus memenuhi enam kriteria tersebut, masih dapat ditemukan sejumlah kolisi pada fungsi *hash*. Tabel berikut menggambarkan kolisi yang mungkin terjadi secara lebih detil.

Algoritma	**Ukuran MD (bit)**	**Ukuran Blok Pesan (bit)**	**Kolisi**
MD2	128	128	Ya
MD4	128	512	Hampir
MD5	128	512	Ya
SHA-0	160	512	Ya
SHA-1	160	512	Hampir
SHA-	256/224	512	Tidak

256/224			
SHA-512/384	512/384	1024	Tidak
WHIRLPOOL	512	512	Tidak

SHA sangatlah beragam yang telah diciptakan oleh orang dengan cara pembuatan yang berbeda – beda. Pada kesempatan ini program yang dibuat oleh penulis menggunakan SHA-1. SHA-1 memiliki hasil nilai hash dalam ukuran 160bit atau 32byte dan hamper tidak memiliki kolisi atau persamaan nilai hash untuk inputan yang sama. SHA-1 memiliki enam proses utama yang harus dilakukan, proses tersebut adalah sebagai berikut :

1. Inisialisasi variabel kunci
2. Penambahan bit 1
3. Pemecahan pesan ke dalam kelompok berukuran 512-bit
4. Ekstensi pesan
5. Iterasi utama
6. Pembangkitan hash value.

Berikut merupakan *pseudocode* yang dapat lebih menggambarkan proses yang terjadi dalam algoritma SHA-1

```
Initialize variables:
h0 = 0x67452301
h1 = 0xEFCDAB89
h2 = 0x98BADCFE
h3 = 0x10325476
h4 = 0xC3D2E1F0


Pre-processing:
append the bit '1' to the message
   append 0 ≤ k < 512 bits '0', so that the
resulting message length (in bits)

is congruent to 448 ≡ −64 (mod 512)

append length of message (before pre-
processing), in bits, as 64-bit big-endian
integer

Process the message in successive 512-bit
chunks:

break message into 512-bit chunks

for each chunk
break chunk into sixteen 32-bit big-endian
words w[i], 0 ≤ i ≤ 15

Extend the sixteen 32-bit words into eighty
32-bit words:

for i from 16 to 79
w[i] = (w[i-3] xor w[i-8] xor w[i-14] xor
w[i-16]) leftrotate 1

Initialize hash value for this chunk:
       a = h0
       b = h1
       c = h2
       d = h3
       e = h4


       Main loop:
       for i from 0 to 79

           if 0 ≤ i ≤ 19 then
               f = (b and c) or ((not b)
and d)
               k = 0x5A827999

           else if 20 ≤ i ≤ 39
               f = b xor c xor d
               k = 0x6ED9EBA1

           else if 40 ≤ i ≤ 59
               f = (b and c) or (b and d)
or (c and d)
               k = 0x8F1BBCDC

           else if 60 ≤ i ≤ 79
               f = b xor c xor d
               k = 0xCA62C1D6




           temp = (a leftrotate 5) + f + e +
k + w[i]
           e = d
           d = c
           c = b leftrotate 30
           b = a
           a = temp
Add this chunk's hash to result so far:
h0 = h0 + a
h1 = h1 + b
h2 = h2 + c
h3 = h3 + d
h4 = h4 + e
Produce the final hash value (big-endian):
   digest = hash = h0 append h1 append h2
append h3 append h4
```

dengan mengubah *pseudocode* tersebut menjadi bahasa java maka kita bisa memperoleh nilai hash dari string yang kita masukkan kedalam fungsi tersebut. Contoh dari nilai string dan hasil nilai hash yang dimasukkan.

Input	***Message Digest***
Hasil fungsi hash.	0c51eddeac6f352aa9ff53 d230c866a736e67011
Hasil fungsi hash!	4cdfc4730a95127fcdb8f3 ca9746300b71427aa8
Hasil fungsi hash	3b531aa1a78fa48a3d9d77 da4aa4d4e63c0c812c
Hasil	19e1039427aac9c59aea86 19d0c66545eb5ebdf6

Semua masukan hampir mirip, dari masukan pertama

hingga ketiga hanya berbeda akhiran dan hasil hash yang diperoleh sangatlah berbeda jauh. Begitu juga pada inputan yang ke empat, walaupun ukurannya hanya lima char tetap menghasilkan nilai hash 32byte. Hal ini yang akan dimanfaatkan oleh penulis untuk menyimpan keterangan dari file atau dokumen – dokumen penting yang disimpan user untuk dilakukan pengecekan file tersebut.

V. STRUKTUR FILE EKSTERNAL

Hasil hash atau nilai hash hanya berukuran 32byte sehingga jika dibuat file eksternal yang menyimpan data tersebut ukurannya tidaklah akan memakan space yang besar oleh karena itu penulis lebih memilih untuk menggunakan file eksternal. Selain itu file ekternal mudah untuk di pindah – pindahkan ataupun dibawa oleh user (bisa dengan flashdisk ataupun media penyimpanan lainnya) untuk menjaga keamanan file tersebut.

File ekternal yang dimiliki memiliki struktur tersendiri yang telah di rancang oleh penulis agar bisa mengetahui perubahan file yang terjadi. Secara umum strukturnya sebagai berikut :

Path File	###	Nilai Hash	###

Mofidied Date

Jadi kita akan menyimpan path dari file tersebut dari root direktorinya, lalu nilai hashnya untuk mengecek perubahan file dan modified date untuk verifikasi perubahan file. Modified date yang digunakan dalam bentuk "long" jadi agar mudah dibandingkan daripada harus dalam bentuk format tanggal maka kita harus membandingkan tahun, bulan, tanggal, jam, menit dan detiknya.

Dalam java mengubah date menjadi long sangatlah mudah, yaitu dengan menggunakan fungsi yang disediakan oleh java seperti berikut ini :

```
File file = new File(path file);

Date date = new Date(file.lastModified());

Long longdate = date.getTime();
```

Disini penulis menggunakan modified date dan nilai hash dengan tujuan untuk mengetahui apakah terjadi perubahan file oleh orang lain, namun jika orang lain tersebut mengesave kembali file tersebut ke keadaan semua otomatis modified datenya akan berubah namun isi filenya tidak berubah. Untuk itu perlu di bandingakn modified date dan nilai hashnya juga.

Untuk menangani ukuran file yang lebih besar maka format penulisan di file ekternal tidak mengalami perubahan namun, pada pathfile di berikan lambing tambahan ya itu "*n*" yang nilai n adalah bagian ke berapa dari potongan file tersebut.

Berikut adalah contoh tampilan file ekternal yang dibuat oleh program tersebut

```
D:\Kuliah\Semester 6\AI\MateriMinggu2a11_AI_ProblemSolving.pdf###458472e93843894a84758e9854f9485a###1302765236000
D:\Kuliah\Semester 6\AI\MateriMinggu2b11_AI.pdf###312351123f12359e92392b0394309d3###1302366124000
D:\Installer\netbeans-6.9.1-ml-windows*1*###1239382e9483a8984b0239d9384398f3###1300291426000
D:\Installer\netbeans-6.9.1-ml-windows*2*###ad3413245ec45425493e099c8775b122###1300291426000
D:\Installer\netbeans-6.9.1-ml-windows*3*###a647947d859754e85916b90757f85586###1300291426000
D:\Installer\netbeans-6.9.1-ml-windows*4*###947e874932b93743764e897347a84587###1300291426000
D:\Installer\netbeans-6.9.1-ml-windows*5*###30c9483a845749f94574d94854e58498###1300291426000
D:\Installer\netbeans-6.9.1-ml-windows*6*###23928475e0c974a9434e847f8347f873###1300291426000
```

Diatas ada 3 file yang dimasukkan kedalam program dan menghasilkan file eksternal seperti pada tampilan diatas. Untuk file "netbeans-6.9.1-ml-windows" dipotong menjadi 6 bagian yang berbeda dengan tiap bagian dibagi dalam ukuran 64MB. Sedangkan 2 file lainnya berukuran dibawah 64MB sehingga tidak perlu dipotong lagi oleh program dan dituliskan dalam file ekternal.

VI. CARA KERJA PROGRAM

Setelah membuat format file eksternal, proses pembacaan file juga telah di tentukan dan fungsi hash telah dibuat, maka berikutnya kita akan masuk pada proses kerja program utama.

Program ini memiliki dua vitur yaitu "create" dan "check". Untuk create yang dilakukan oleh program adalah membuat file eksternal seperti pada format yang telah dijelaskan sebelumnya. Jadi program membuat hash dari file yang dibaca dan menuliskannya kedalam file

eksternal. Program tidak langsung membaca semua isi direktori, namun program menuliskan satu persatu kedalam file eksternal tiap satu file yang dibaca oleh program tersebut. Sehingga memory yang dibutuhkan tidaklah terlalu banyak.

Untuk proses “check”prinsip kerjanya sangatlah simple yaitu hanya membuat kembali file eksternal dari direktori yang dipilih dan membandingkannya dengan file eksternal yang telah dibuat sebelumnya. Namun file eksternal saat melakukan pengecekan ditulisakan pada direktori program. Nanti file tersebut yang akan dibandingkan dengan file eksternal yang dimilki oleh user yang telah dibuat oleh program ini sebelumnya.

Setelah melalui proses pengecekan, maka file dummy tersebut yang digunakan untuk mengecek, akan dihapus. Pada program akan keluar list file yang berubah. Jika tidak ada file yang berubah pada direktori yang ditentukan user untuk menyimpan file tersebut. Jika tidak ada perubahan maka table hanya akan terisi tulisan “Nothing Change”.

Pada pengecekan ini ada beberapa hal yang diperhatikan, hal – hal tersebut adalah :

1. File eksernal dicocokkan secara langsung yaitu line pertama pada file eksternal dummy sama atau tidak dengan line pertama pada file ekternal yang dimilki oleh user.
2. Jika terjadi berbedaan maka yang akan dilakukan adalah pencarian pada file eksternal user, path dari file yang ada didummy yang tidak ditemukan. Untuk mendapatkan pathny harus displit terlebih dahulu dengan “###” dan “*”.
3. Jika file tersebut ditemukan maka ambil hashnya dengan parsing String (di split) dan dibandingkan. Jika hasil hashnya sama tapi modified timenya sama maka akan keluar pada table “File x have been open but nothing change”, tetapi jika file tersebut berbeda hashnya maka pada table ”File x have been change”.
4. Jika ada bagian dari file ekternal yang dimiliki user namun tidak dimiliki oleh file ekternal dummy maka file tersebut dikatan di delete.
5. Jika ada bagian dari file eksternal dummny tapi tidak dimiliki oleh file ekternal pada user maka file tersebut dikatakan telah ditambah.

Dengan aturan diatas maka user akan mengetahui apakah file tersebut berubah, ditambah dan dihapus. Program ini sangatlah simple, dan untuk keamanan file ekternal yang disimpan oleh user, user bisa menggunakan enkripsi dan dekripsi agar file tersebut tidak bisa dibaca orang jika ia membukanya dengan notepad. Tentunya akan membutuhkan kunci – kunci tertentu untuk menggunakan enkripsi dan dekripsi sehingga user harus mengingat kunci tersebut.

VI. KESIMPULAN

Pembuatan file eksternal berhasil dibentuk dan bisa dimanfaatkan untuk melakukan pengecekan perubahan file. Jika terjadi perbuhan file maka program akan mengecek file eksternal tersebut dan membandingkannya dengan keadaan sekarng. Dari file eksternal tersebut bisa diketahui penambahan file, pengahapusan file dan bahkan pengubahan isi file tersebut. Walaupun ukuran file tersebut lebih besar namun bisa digunakan pemotongan file menjadi beberapa bagian yang di jadikan hash pada berikutnya. Dan untuk file eksternal, diberikan kebebasan user untuk menentukan format dan letak file tersebut dalam direktori agar susah untuk ditemukan oleh orang lain jika orang tersebut mengetahui user menggunakan program ini. Selain itu, untuk keamanan file, maka file ekternal tersebut dienkripsi terlebih dahulu dengan kunci yang ditentukan oleh user.

REFERENSI

[1] http://www.velocityreviews.com/forums/t141739-max-length-of-a-string.html
[2] http://www.javadb.com/change-last-modified-time-of-a-file-or-directory
[3] http://stackoverflow.com/questions/4375118/reading-buffered-binary-file-with-seek
[4]

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, 23 Maret 2011

Yogi Adytia Marsal
13508016