

# Implementasi SHA untuk Komparasi File

Biolardi Yoshogi and 13509035

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

vsio@students.itb.ac.id

**Abstract**—Memeriksa apakah suatu file telah dimodifikasi atau tidak, telah banyak digunakan dalam kehidupan sehari-hari seperti file share (jika hasil modifikasi beda, maka di-commit), compile bahasa pemrograman seperti bahasa ADA (jika hasil modifikasi sama, akan muncul pesan up-to-date), pengecekan sebuah file apakah mengandung virus atau tidak pada file yang tidak perlu dimodifikasi lagi, dan lain-lain. Untuk memeriksa suatu file apakah telah dimodifikasi atau belum, bisa dengan menggunakan fungsi perbandingan string untuk per karakter atau panjang string. Akan tetapi, jika file yang ingin dibandingkan lebih dari satu, tentu saja kompleksitas waktunya akan menjadi lebih tinggi. Hal ini membuat mesin harus beroperasi lebih banyak. Waktu yang digunakan pun semakin lebih lama. Belum lagi kecepatan komputer yang digunakan bisa berbeda-beda untuk tiap komputer. Hal ini membuat metode tersebut menjadi kurang efektif dan efisien, sedangkan ukuran file yang akan dimasuki bisa besar (dari Mega hingga Giga pada umumnya). Masalah ini, bisa diselesaikan dengan memanfaatkan fungsi hash. Salah satunya adalah fungsi SHA (Secure Hash Algorithm). Dengan menggunakan fungsi SHA sebagai hash-nya pada file original, akan didapat message-digest-nya. Message digest tersebut digunakan lagi untuk dibandingkan dengan message-digest file sama namun dicurigai telah dimodifikasi. Message-digest file original tersebut dapat digunakan kembali tanpa perlu file originalnya perlu di-hash lagi. Yang hanya perlu di-hash adalah file yang dicurigai tidak original sehingga waktu dan banyak operasi mesin yang dibutuhkan menjadi relatif lebih singkat. File yang ukurannya sangat besar pun, lama waktu prosesnya bisa dikurangi dengan memanfaatkan hash tersebut.

**Kata Kunci**— sha, kesamaan file, modifikasi, pengujian

## I. PENDAHULUAN

Memeriksa apakah suatu file telah dimodifikasi atau tidak, telah banyak diimplementasikan dalam kehidupan sehari-hari. Dilakukannya pemeriksaan ini, tentunya meningkatkan efisiensi dan efektifitas karena menguji apakah sebaiknya melakukan suatu task atau tidak berdasarkan kesamaan file tersebut. Jika suatu task dilakukan padahal filenya yang diuji sama, tentu saja hal ini mengakibatkan boros waktu karena melakukan task yang seharusnya tidak perlu dilakukan, dan tidak efektif yang mana meningkatkan kinerja komputer yang lebih berakibat boros energi. Hal ini mengakibatkan seberapa pentingnya melakukan pemeriksaan apakah file tersebut

sama atau tidak.

Dengan melakukan pemeriksaan apakah file telah dimodifikasi atau tidak, tentunya banyak manfaat yang bisa diperoleh dengan pemeriksaan tersebut. Di antaranya adalah file share bersistem cloud (jika hasil modifikasi beda, maka di-commit), compile bahasa pemrograman seperti bahasa ADA (jika hasil modifikasi sama, akan muncul pesan up-to-date), pengecekan sebuah file apakah mengandung virus atau tidak pada file yang tidak perlu dimodifikasi lagi, dan lain-lain.

Sekarang, semakin besarnya kebutuhan fungsionalitas suatu program, maka semakin besar pula ukuran file. Hal ini berarti juga meningkatkan panjang string suatu file. Jika file diperiksa per karakter satu persatu persatu, hal ini mungkin tidak masalah jika ukuran filenya sangat kecil. Akan tetapi, jika ukuran filenya sangat besar (hingga beberapa GB), hal ini tentunya akan menjadi sangat masalah untuk dilakukan pengujian kesamaan antar file karena dapat meningkatkan lama waktu yang dibutuhkan dan jumlah kinerja yang harus dilakukan mesin hingga EOF. Belum lagi jika harus dilakukan pengecekan kesamaan lebih dari satu file atau perlu dilakukan pengecekan kesamaan file lebih dari satu kali. Menggunakan cara pengecekan string per karakter tentunya akan membuat hal ini semakin tidak efisien dan tidak efektif. Belum lagi kecanggihan kinerja mesin itu sendiri.

## II. TEORI DASAR

SHA adalah sejenis fungsi hash satu arah yang dikembangkan oleh NIST. SHA didasarkan pada MD4. SHA lebih aman karena message digest yang dihasilkan tidak mungkin berkoresponden dengan pesan asli.

Terdapat beberapa jenis SHA yang standar:

- SHA-0 dengan panjang 160 bit
- SHA-1 dengan panjang 160 bit
- SHA-2 yang terdiri dari
  - SHA-224 dengan panjang 224 bit
  - SHA-256 dengan panjang 256 bit
  - SHA-384 dengan panjang 384 bit
  - SHA-512 dengan panjang 512 bit

Beberapa langkah penambahan bitnya yaitu:

1. Padding bits

2. Ditambahnya panjang pesan masukan
3. Inisialisasi Buffer
4. Proses pesan

### III. METODE PEMECAHAN MASALAH

Untuk mengatasi masalah tersebut, maka digunakan cara, yaitu menghash file yang akan dibandingkan. Dalam makalah ini, hash yang akan digunakan adalah SHA (termasuk keluarganya yang sejenis seperti SHA0, SHA1, dan SHA2). Dengan dihashnya kedua file tersebut, maka akan dihasilkan message digest yang mana akan digunakan sebagai perbandingan.

Karena hash umumnya memiliki sensitifitas yang lumayan tinggi, message digest yang dihasilkan pun umumnya bisa berbeda meskipun hanya beda satu karakter atau beda satu jarak antar indeks ASCII. Hal ini dapat dimanfaatkan untuk pengujian apakah kedua file tersebut sama atau tidak yang mana jumlah kerjanya dapat lebih dihemat.

Message digest yang dihasilkan oleh sebuah file yang menjadi inti pengujian suatu file pun disimpan (bisa di dalam file yang berbeda, dalam program (sementara), registry, atau metode lainnya selama message digest simpanan tersebut bisa dipanggil kembali). Dari message digest tadi, dapat dibandingkan dengan perbandingan jumlah nilai atau string (jika dikonversi ke string). Jika dibandingkan sebagai string, maka pemeriksaan per karakter pun hanya berdasarkan panjang bitnya. Seperti SHA-512, maka pemeriksaan karakter hanya 512 kali. Berbeda drastis jika file yang sangat besar, misal panjangnya 10000 karakter, dengan dihashnya file tersebut, maka penghematan bisa mencapai sekitar 20 kali lipat.

### IV. EKSPERIMEN DAN HASILNYA

Untuk menguji apakah dengan menghash file tersebut dapat menghemat jumlah eksekusi dan menghemat waktu, akan dilakukan eksperimen. Akan digunakan sebuah algoritma (yang mana source codenya dalam bahasa ruby ) yang dilengkapi lama waktu eksekusi dan jumlah eksekusi pada perbandingan filenya. Untuk jumlah eksekusi, hanya akan dipasang di saat perbandingan filenya saja. Jika jumlah eksekusi dirasa kurang sesuai karena tidak diawasinya jumlah iterasi pada hash yang dilakukan, masih bisa dibandingkan dengan waktu.

File yang akan diujipun, inputnya adalah jenis file dan ukuran file inti. Outputnya adalah apakah sama atau tidak, jumlah iterasi, dan lama waktu perbandingan yang direpresentasikan dalam bentuk gambar screenshot. File yang akan diuji dengan file inti ada tiga. Masing-masing file mempresentasikan atribut-atribut output yang

telah disebutkan tadi. Dari output tadi, akan dibagi menjadi dua bagian: apakah file tersebut tidak dihash dan file tersebut dihash.

Asumsi yang digunakan sebagai berikut:

- Jumlah iterasi dan lama waktu perbandingan hanya dilakukan saat perbandingan per-karakter, bukan saat dihash, menerima masukan file, dan di luar perbandingan tadi
- Asumsi jenis file masukan dengan tiga file untuk dibandingkan adalah sama
- File yang sama hanya satu karena pada intinya sama saja jumlah iterasi dan lama waktu jika file tersebut sama
- Jika dalam proses iterasi ada satu karakter saja yang tidak sama, iterasi dihentikan dan mengembalikan jumlah iterasi tersebut
- Message digest berbasis 16

#### Percobaan 1

##### Input:

- Jenis File: .txt
- Ukuran File: 1 KB

##### Output:

```
Non-Hashed Result:
File #1
isSame: true
Iteration: 20
Time: 0.00000000

File #2
isSame: false
Iteration: 1
Time: 0.00000000

File #3
isSame: false
Iteration: 1
Time: 0.00000000
```

```
C:\WINDOWS\system...
Hashed Result:
File #1
SHA type: SHA-0
isSame: true
Iteration: 40
Time: 0.0000000

File #1
SHA type: SHA-1
isSame: true
Iteration: 40
Time: 0.0000000

File #1
SHA type: SHA-224
isSame: true
Iteration: 56
Time: 0.0000000

File #1
SHA type: SHA-256
isSame: true
Iteration: 64
Time: 0.0000000

File #1
SHA type: SHA-384
isSame: true
Iteration: 96
Time: 0.0000000

File #1
SHA type: SHA-512
isSame: true
Iteration: 128
Time: 0.0000000
```

```
C:\WINDOWS\system...
File #3
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

**Percobaan 2**

**Input:**

- Jenis File: jpg
- Ukuran File: 10 KB

**Output:**

```
C:\WINDOWS\system...
File #2
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

```
Non-Hashed Result:
File #1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
isSame: true
Iteration: 9357
Time: 0.0156250

File #3
isSame: false
Iteration: 1
Time: 0.0000000
```

```
C:\WINDOWS\s...
Hashed Result:
File #1
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-512
isSame: false
Iteration: 2
Time: 0.0000000
```

```
C:\WINDOWS\s...
File #2
SHA type: SHA-0
isSame: true
Iteration: 40
Time: 0.0000000

File #2
SHA type: SHA-1
isSame: true
Iteration: 40
Time: 0.0000000

File #2
SHA type: SHA-224
isSame: true
Iteration: 56
Time: 0.0000000

File #2
SHA type: SHA-256
isSame: true
Iteration: 64
Time: 0.0000000

File #2
SHA type: SHA-384
isSame: true
Iteration: 96
Time: 0.0000000

File #2
SHA type: SHA-512
isSame: true
Iteration: 128
Time: 0.0000000
```

```
C:\WINDOWS\s...
File #3
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-224
isSame: false
Iteration: 2
Time: 0.0000000

File #3
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

### Percobaan 3

#### Input:

- Jenis File: rb
- Ukuran File: 5 KB

#### Output:

```
Non-Hashed Result:
File #1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
isSame: false
Iteration: 1
Time: 0.0000000

File #3
isSame: true
Iteration: 4417
Time: 0.0000000
```

```
C:\WINDOWS\syst...
Hashed Result:
File #1
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

```
C:\WINDOWS\syst...
File #2
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

```
C:\WINDOWS\syst...
File #3
SHA type: SHA-0
isSame: true
Iteration: 40
Time: 0.0000000

File #3
SHA type: SHA-1
isSame: true
Iteration: 40
Time: 0.0000000

File #3
SHA type: SHA-224
isSame: true
Iteration: 56
Time: 0.0000000

File #3
SHA type: SHA-256
isSame: true
Iteration: 64
Time: 0.0000000

File #3
SHA type: SHA-384
isSame: true
Iteration: 96
Time: 0.0000000

File #3
SHA type: SHA-512
isSame: true
Iteration: 128
Time: 0.0000000
```



#### Percobaan 4

##### Input:

- Jenis File: gmk
- Ukuran File: 17 KB

##### Output:

```
Non-Hashed Result:  
File #1  
isSame: true  
Iteration: 17354  
Time: 0.0156250  
  
File #2  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #3  
isSame: false  
Iteration: 1  
Time: 0.0000000
```

```
C:\WINDOWS\system3...  
Hashed Result:  
File #1  
SHA type: SHA-0  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #1  
SHA type: SHA-1  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #1  
SHA type: SHA-224  
isSame: true  
Iteration: 56  
Time: 0.0000000  
  
File #1  
SHA type: SHA-256  
isSame: true  
Iteration: 64  
Time: 0.0000000  
  
File #1  
SHA type: SHA-384  
isSame: true  
Iteration: 96  
Time: 0.0000000  
  
File #1  
SHA type: SHA-512  
isSame: true  
Iteration: 128  
Time: 0.0000000
```

```
C:\WINDOWS\system3...  
File #2  
SHA type: SHA-0  
isSame: false  
Iteration: 2  
Time: 0.0000000  
  
File #2  
SHA type: SHA-1  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-224  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-256  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-384  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-512  
isSame: false  
Iteration: 1  
Time: 0.0000000
```

```
C:\WINDOWS\syst...  
File #3  
SHA type: SHA-0  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #3  
SHA type: SHA-1  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #3  
SHA type: SHA-224  
isSame: true  
Iteration: 56  
Time: 0.0000000  
  
File #3  
SHA type: SHA-256  
isSame: true  
Iteration: 64  
Time: 0.0000000  
  
File #3  
SHA type: SHA-384  
isSame: true  
Iteration: 96  
Time: 0.0000000  
  
File #3  
SHA type: SHA-512  
isSame: true  
Iteration: 128  
Time: 0.0000000
```

## Percobaan 5

### Input:

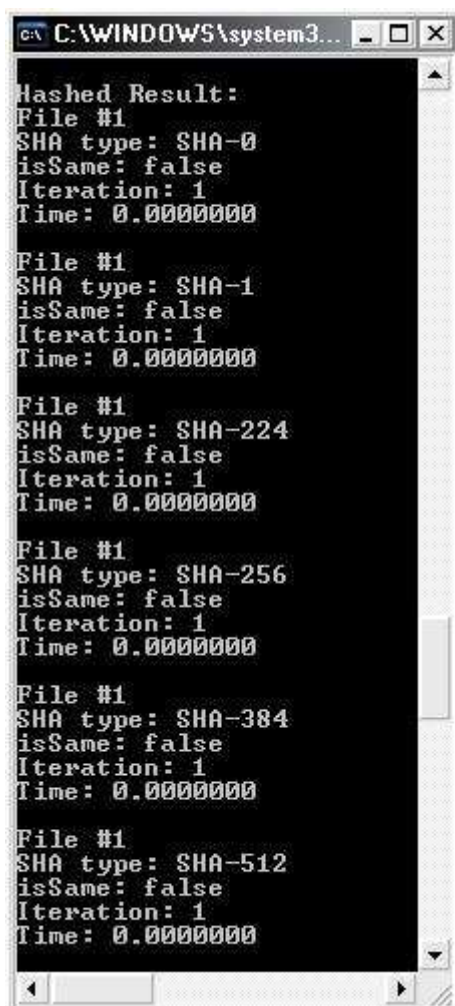
- Jenis File: wav
- Ukuran File: 217 KB

### Output:

```
Non-Hashed Result:
File #1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
isSame: true
Iteration: 221915
Time: 0.1875000

File #3
isSame: false
Iteration: 1
Time: 0.0000000
```



```
C:\WINDOWS\system32\cmd.exe
Hashed Result:
File #1
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

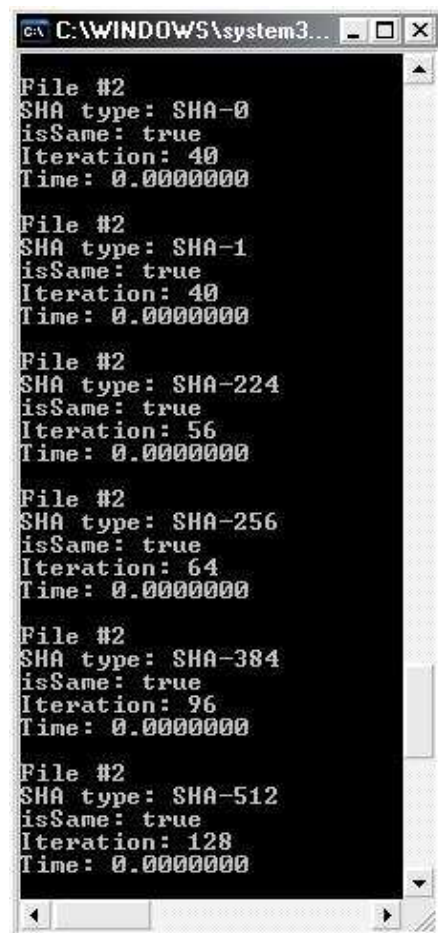
File #1
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #1
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```



```
C:\WINDOWS\system32\cmd.exe
File #2
SHA type: SHA-0
isSame: true
Iteration: 40
Time: 0.0000000

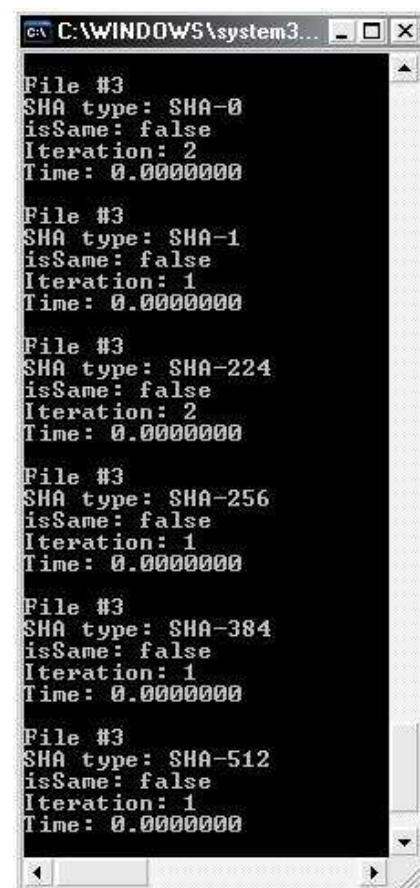
File #2
SHA type: SHA-1
isSame: true
Iteration: 40
Time: 0.0000000

File #2
SHA type: SHA-224
isSame: true
Iteration: 56
Time: 0.0000000

File #2
SHA type: SHA-256
isSame: true
Iteration: 64
Time: 0.0000000

File #2
SHA type: SHA-384
isSame: true
Iteration: 96
Time: 0.0000000

File #2
SHA type: SHA-512
isSame: true
Iteration: 128
Time: 0.0000000
```



```
C:\WINDOWS\system32\cmd.exe
File #3
SHA type: SHA-0
isSame: false
Iteration: 2
Time: 0.0000000

File #3
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-224
isSame: false
Iteration: 2
Time: 0.0000000

File #3
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

### Percobaan 6

#### Input:

- Jenis File: pdf
- Ukuran File: 152 KB

#### Output:

```
Non-Hashed Result:  
File #1  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #3  
isSame: true  
Iteration: 154714  
Time: 0.1249998
```

```
Hashed Result:  
File #1  
SHA type: SHA-0  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #1  
SHA type: SHA-1  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #1  
SHA type: SHA-224  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #1  
SHA type: SHA-256  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #1  
SHA type: SHA-384  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #1  
SHA type: SHA-512  
isSame: false  
Iteration: 1  
Time: 0.0000000
```

```
File #2  
SHA type: SHA-0  
isSame: false  
Iteration: 2  
Time: 0.0000000  
  
File #2  
SHA type: SHA-1  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-224  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-256  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-384  
isSame: false  
Iteration: 1  
Time: 0.0000000  
  
File #2  
SHA type: SHA-512  
isSame: false  
Iteration: 1  
Time: 0.0000000
```

```
File #3  
SHA type: SHA-0  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #3  
SHA type: SHA-1  
isSame: true  
Iteration: 40  
Time: 0.0000000  
  
File #3  
SHA type: SHA-224  
isSame: true  
Iteration: 56  
Time: 0.0000000  
  
File #3  
SHA type: SHA-256  
isSame: true  
Iteration: 64  
Time: 0.0000000  
  
File #3  
SHA type: SHA-384  
isSame: true  
Iteration: 96  
Time: 0.0000000  
  
File #3  
SHA type: SHA-512  
isSame: true  
Iteration: 128  
Time: 0.0000000
```



## Percobaan 7

### Input:

- Jenis File: rar
- Ukuran File: 13 KB

### Output:

```
Non-Hashed Result:
File #1
isSame: true
Iteration: 13242
Time: 0.0156250

File #2
isSame: false
Iteration: 1
Time: 0.0000000

File #3
isSame: false
Iteration: 1
Time: 0.0000000
```

```
C:\WINDOWS\syst...
Hashed Result:
File #1
SHA type: SHA-0
isSame: true
Iteration: 40
Time: 0.0000000

File #1
SHA type: SHA-1
isSame: true
Iteration: 40
Time: 0.0000000

File #1
SHA type: SHA-224
isSame: true
Iteration: 56
Time: 0.0000000

File #1
SHA type: SHA-256
isSame: true
Iteration: 64
Time: 0.0000000

File #1
SHA type: SHA-384
isSame: true
Iteration: 96
Time: 0.0000000

File #1
SHA type: SHA-512
isSame: true
Iteration: 128
Time: 0.0000000
```

```
C:\WINDOWS\syst...
File #2
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #2
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

```
C:\WINDOWS\syst...
File #3
SHA type: SHA-0
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-1
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-224
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-256
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-384
isSame: false
Iteration: 1
Time: 0.0000000

File #3
SHA type: SHA-512
isSame: false
Iteration: 1
Time: 0.0000000
```

## V. ANALISIS

Dari hasil eksperimen yang didapat, terdapat beberapa hal yang bisa dianalisis. Hal ini termasuk yang diharapkan maupun yang tidak diharapkan.

Pada perbandingan apakah sebuah file tersebut sama atau tidak, bisa terlihat bahwa baik yang tidak dihash maupun dihash, didapat bahwa hasil perbandingannya sama.

Pada jumlah iterasi yang dilakukan jika hasilnya sama, jumlah iterasi yang dilakukan jika file tidak dihash adalah panjang string file tersebut (dengan 1 karakter = 8 bit). Untuk jumlah iterasi didapat bahwa pada percobaan 1, jumlah iterasi untuk yang tidak dihash lebih kecil daripada yang dihash. Sedangkan pada percobaan lainnya, jumlah iterasi untuk yang tidak dihash lebih besar daripada jika dihash. Jumlah iterasi hashnya bergantung pada jenisnya (dalam hal ini berdasarkan jumlah bitnya yang mana jumlah bit dibagi 8).

Tambahan untuk jumlah iterasi yang jika hasilnya sama, untuk tiap jenis hash, jumlah iterasi yang dilakukan cenderung berdasarkan jenis SHA masing-masing yang mana dalam hal ini berdasarkan jumlah bitnya dibagi delapan.

Untuk jumlah iterasi jika file tidak sama, jika tidak dihash, jumlah iterasi yang dilakukan adalah mendekati jumlah panjang string isi file (1 karakter = 8 bit). Sedangkan jika dihash, panjangnya adalah panjang message digest berdasarkan jenis SHA-nya. Kasus terburuk jika tidak dihash adalah panjang string dikurangi 1, sedangkan jika dihash, panjang string message digest (berdasarkan masing-masing jenis SHA-nya) dikurangi 1. Dalam percobaan tadi, cenderung terdapat 1 dan 2 pada pemeriksaan message-digest. Hal ini menandakan bahwa karakter mulai tidak sama saat iterasi pertama atau kedua.

Untuk lama waktu yang didapat, rata-rata waktu yang didapat cenderung dibawah 1 sekon. Hal ini ada kemungkinan karena kecepatan CPU sehingga hampir tidak terlihat perbedaannya. Akan tetapi, pada percobaan yang iterasinya sangat banyak, terlihat bahwa waktu yang dibutuhkan lebih lama dibandingkan dengan jumlah iterasi yang lebih sedikit. Dalam percobaan yang didapat, terlihat pada pesan yang tidak dihash memiliki lama waktu lebih lama daripada pesan yang dihash. Hal lain yang menyebabkan lebih lama adalah ketika ada karakter yang menyebabkan iterasi menjadi berhenti sementara atau berbunyi "beep".

## VI. KESIMPULAN

Berdasarkan hasil analisis yang didapat, dapat disimpulkan bahwa dengan melakukan hash, jumlah iterasi dan lama waktu yang dibutuhkan dapat dibuat lebih singkat.

Hash pun berfungsi untuk digunakan sebagai perbandingan apakah file tersebut sama atau tidak dengan jumlah iterasi dan lama waktu yang lebih singkat. Hal ini membuat perbandingan lebih efektif dan efisien.

Meskipun hasil lebih baik dengan dihash, ada saatnya hash tidak perlu digunakan jika panjang bit isi file yang tidak dihash kurang dari panjang bit message digest hasil file. Hal ini perlu dilakukan penyaringan lagi apakah sebuah file perlu atau tidak dihash berdasarkan panjang bit yang dimilikinya.

Dari beberapa hasil eksperimen tadi, karena lama waktu dan jumlah iterasi pada proses hashing, pengambilan file, dan di luar perbandingan file tidak dihitung, ada kemungkinan saat digabung dengan fungsi dan program lain, akan ada sedikit perbedaan yaitu jumlah iterasi dan lama waktu mungkin lebih besar. Akan tetapi, abaikan hal tadi, terlihat bahwa dengan dihash secara selektif, maka perbandingan file lebih efisien dan efektif.

## REFERENCES

- [1] Munir, Rinaldi. Ir. M.T. Diktat Kuliah IF5054 Kriptografi.

## 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, 14 Mei 2012



Biolardi YOSHOGI, 13509035