# Aplikasi Fungsi *Hash* MD5 untuk Pengecekan Konten Laman *Website*

Akbar Suryowibowo Syam - 13511048
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*Akbar.syam@s.itb.ac.id*

***Abstrak* — Algoritma MD5 adalah salah satu fungsi *hash* yang sering digunakan di bidang kriptografi. MD5 ditemukan oleh Ron Rivest dari MIT pada tahun 1991 untuk menggantikan algoritma *hash* yang sebelumnya, yaitu MD4. Seperti algoritma *hash* lainnya, algoritma ini menerima masukan pesan dan mengeluarkan hasil keluaran yang dinamakan *message digest*. Algoritma MD5 menerima masukan pesan yang berukuran berapapun. Ukuran dari *message digest* yang dihasilkan berukuran 128 bit atau biasanya ditampilkan dalam notasi heksadesimal.**

**Salah satu kegunaan algoritma MD5 adalah untuk melakukan pengecekan apakah konten dari sebuah laman dokumen sudah diubah atau belum. Hal ini berguna untuk mengecek apakah rujukan yang dilakukan oleh seseorang pada sebuah laman *website* masih berlaku kredibilitasnya ataukah sudah berbeda kontennya.**

***Kata Kunci— MD5, hash, konten website***

## I. PENDAHULUAN

Dewasa ini jumlah laman *website* yang ada di internet berjumlah sangat banyak, mencapai angka milyaran. Setiap laman menyimpan informasi yang beragam jenisnya. Hal ini dikarenakan karena semua orang dapat memasukkan informasi apapun yang diinginkannya di internet. Hal tersebut mengakibatkan kredibilitas dari informasi yang disediakan harus dipertanyakan. Setiap informasi yang ada di internet dapat diubah, ditambah, maupun dihilangkan kapanpun. Bahkan beberapa situs penyedia informasi seperti Wikipedia mengizinkan siapapun untuk dapat mengubah konten dari sebuah laman yang tersedia.

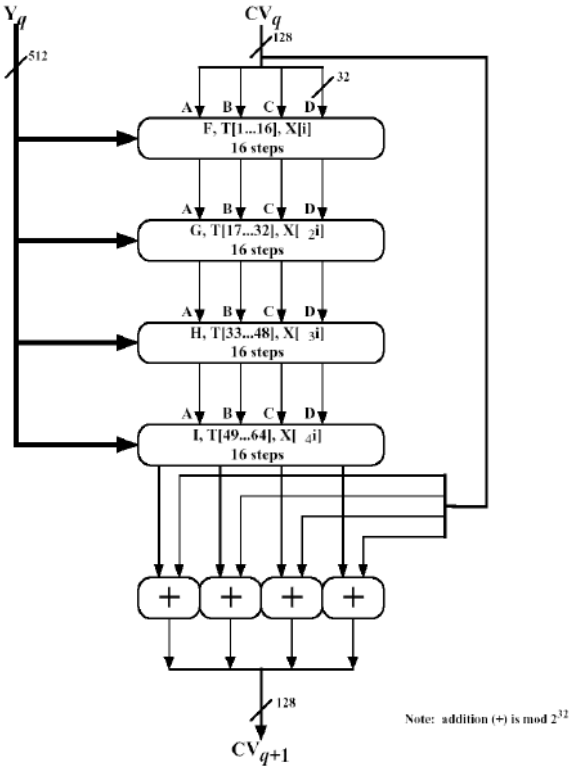
Hal ini menyulitkan bagi seseorang yang ingin merujuk pada sebuah artikel yang berisi informasi yang diinginkan di internet. Tidak ada jaminan bahwa laman yang telah dibuka sebelumnya masih memiliki konten yang sama dengan ketika laman tersebut pertama kali dibuka. Konten informasi yang dirujuk mungkin saja sudah diubah maupun dihapus oleh orang lain sehingga informasi yang dirujuk tidak lagi dapat dipertanggungjawabkan. Selain masalah itu, apabila ingin dirujuk balik oleh orang lain, orang tersebut tidak dapat mengetahui apakah konten yang dimaksud sudah diubah oleh orang lain atau belum. Untuk menyelesaikan permasalahan tersebut, diperlukan sebuah cara untuk memverifikasi konten dari laman yang diakses apakah masih memiliki konten yang sama dari saat pertama kali laman tersebut dirujuk.

## II. DASAR TEORI

### *A. Algoritma Hash MD5*

Algoritma MD5 adalah salah satu fungsi *hash* yang sering digunakan di bidang kriptografi. MD5 ditemukan oleh Ron Rivest dari MIT pada tahun 1991 untuk menggantikan algoritma *hash* yang sebelumnya, yaitu MD4.

Seperti algoritma *hash* lainnya, algoritma ini menerima masukan pesan dan mengeluarkan hasil keluaran yang dinamakan *message digest*. Algoritma MD5 menerima masukan pesan yang berukuran berapapun. Ukuran dari *message digest* yang dihasilkan berukuran 128 bit atau biasanya ditampilkan dalam notasi heksadesimal.




**Gambar II.1. Diagram kerja algoritma MD5 pada sebuah blok pesan**

Diagram mengenai cara kerja algoritma MD5 ditampilkan pada Gambar II.1. Langkah-langkah yang dilakukan untuk membangun *message digest* dari pesan masukan pada algoritma MD5:

1. Menambahkan bit-bit pengganjal (*padding bits*)
2. Menambahkan panjang dari pesan semula pada 64 bit terakhir blok pesan
3. Inisialisasi penyangga (*buffer*) *message digest*
4. Pengolahan pesan dalam blok yang berukuran 512 bit.

Secara teori, berdasarkan langkah-langkah yang telah disebutkan di atas, tidak mungkin didapatkan dua buat pesan yang menghasilkan *message digest* yang sama dari algoritma MD5. Hal ini dikarenakan perubahan satu karakter pada pesan saja sudah mengubah nilai *message digest* secara total. Hal ini dapat dilihat pada Gambar II.2.

```
Pesan : saya suka kriptografi
MD : d2ea936dac81d0d122317502076d85bd

Pesan : saya uka kriptografi
MD : 5fb7dd61031d20954e9292878624ed15
```

**Gambar II.2. Perbedaan nilai *hash* yang berbeda jauh walaupun hanya berbeda satu karakter pesan**

Walaupun secara teori tidak mungkin terjadi kolisi pada algoritma MD5, ditemukan kolisi pada algoritma ini. Pada tahun 1996, Hans Dobbertin berhasil menemukan kolisi untuk algoritma MD5, namun masih belum termasuk serangan terhadap algoritma tersebut. Pada tahun 2005, Arjen Lenstra, Xiaoyun Wang, dan Benner de Weger menemukan kolisi untuk MD5 dengan cara membuat dua sertifikat X.509 dengan kunci publik yang sama namun ternyata menghasilkan *message digest* yang sama. Sejak saat itu algoritma MD5 sudah dianggap tidak aman lagi untuk aplikasi kriptografi seperti sertifikat SSL atau tanda tangan digital. Walaupun begitu, MD5 masih dinilai aman digunakan untuk hal-hal diluar keamanan digital seperti pemerikaan integritas dokumen atau perubahan konten sebuah dokumen.

### *B. Aplikasi Pengecek Konten Laman Website*

Setiap penulis makalah pasti membutuhkan referensi dalam penulisan makalahnya. Tidak jarang rujukan yang dilakukan mengacu pada halaman *website* yang berada di internet. Namun, hal ini memiliki kekurangan yaitu konten dari sebuah laman *website* dapat diubah kapanpun oleh orang-orang yang berwenang untuk mengubah. Oleh karena itu, dibutuhkan sebuah cara untuk mengecek apakah konten dari laman *website* yang dibuka masih sama dengan pertama kali diakses atau sudah diubah oleh pemiliknya.

Salah satu cara untuk mengecek apakah konten sebuah laman *website* sudah berubah atau belum adalah dengan menggunakan algoritma *hash*. Sebuah algoritma *hash* yang baik akan memastikan bahwa setiap masukan pesan yang berbeda akan menghasilkan nilai *hash* yang berbeda pula, sehingga pemeriksaan dapat dilakukan dengan menggunakan algoritma *hash*.

## III. PERANCANGAN


**Gambar III.1. Diagram kerja aplikasi pengecek laman *website***

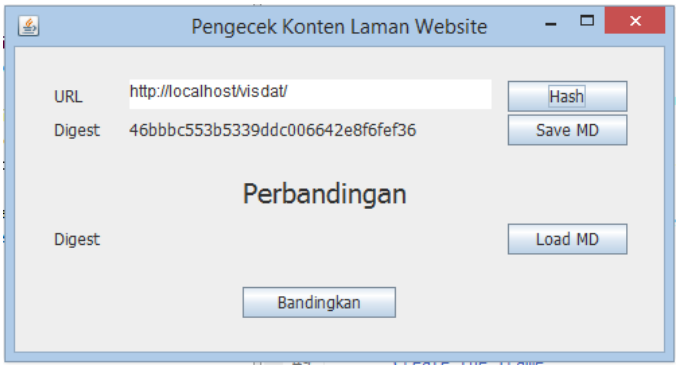
Diagram proses aplikasi pengecekan konten laman *website* dapat dilihat pada Gambar III.1. Secara umum, terdapat dua fungsi utama dari aplikasi pengecek konten laman *website* ini. Fungsi yang pertama adalah untuk menghitung nilai *hash* dari konten sebuah laman *website* dengan masukan URL-nya dan kemudian menyimpan nilai *hash* tersebut. Fungsi yang kedua adalah untuk membandingkan nilai *hash* dari konten laman yang telah disimpan sebelumnya dengan nilai *hash* dari konten laman saat ini. Apabila nilai *hash* dari kedua konten laman tersebut sama, maka dapat diambil kesimpulan bahwa dari dua kali pengaksesan laman tersebut, kontennya belum berubah. Apabila nilai *hash* dari kedua konten tersebut berbeda, maka kesimpulannya adalah kedua konten tersebut memiliki perbedaaan, sesedikit apapun perbedaannya.

Program yang dibuat memiliki fitur-fitur antara lain:

1. Mampu menerima masukan URL dan mendapatkan konten dari situs tersebut
2. Mampu menghitung nilai *hash* dari konten laman yang telah dimuat
3. Mampu menyimpan dan memuat ulang nilai *hash* dari konten laman
4. Mampu membandingkan nilai *hash* dari dua pengaksesan berbeda dari situs yang sama

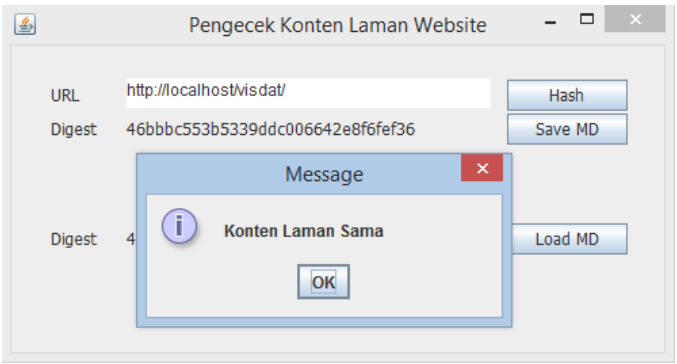
## IV. IMPLEMENTASI

Aplikasi dibuat dalam bahasa Java, menggunakan IDE Eclipse Luna. Proses *hashing* MD5 dan pengambilan konten HTML dari sebuah laman *website* menggunakan *library* bawaan dari Eclipse Luna.

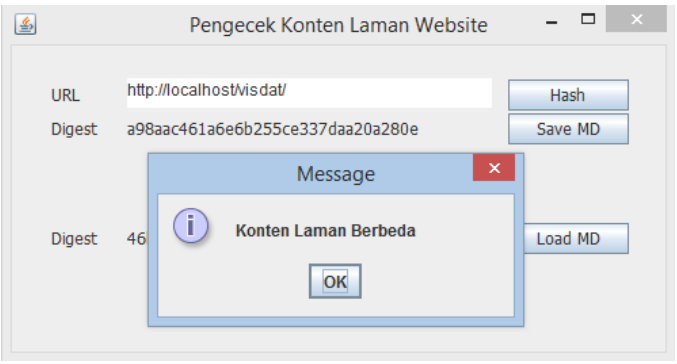



**Gambar IV.1. Tampilan antarmuka utama aplikasi**

Antarmuka aplikasi dapat dilihat pada Gambar IV.1. Secara umum, antarmuka aplikasi dibuat dengan tampilan minimalis yang tetap mencakup fungsi-fungsi yang sudah dijelaskan pada Bab III. Bagian pertama aplikasi adalah *label* dan *textarea* untuk memasukkan URL dari laman yang ingin dijadikan tujuan dan kemudian mendapatkan nilai *hash* dari konten laman tersebut. Selain itu, terdapat juga tombol simpan untuk menyimpan nilai *hash* ke dalam *file* eksternal. Bagian kedua dari aplikasi ini adalah bagian perbandingan antara nilai *hash* yang dimiliki pada bagian pertama dan dibandingkan dengan nilai *hash* yang sudah pernah disimpan sebelumnya. Oleh karena itu, pada bagian kedua ini terdapat tombol untuk memuat *file* eksternal yang berisi nilai *hash*. Selain itu, pada bagian kedua ini terdapat tombol yang berfungsi untuk membandingkan nilai hash dari bagian pertama dan bagian kedua. Apabila nilai *hash* sama, maka dapat diambil kesimpulan bahwa konten dari laman *website* yang dibuka masih sama, dan menampilkan kotak dialog yang mengatakan hal serupa, seperti yang ditampilkan pada Gambar IV.2. Apabila nilai *hash* berbeda, maka konten dari kedua laman memiliki perbedaan. Tampilan kotak dialog untuk nilai *hash* yang berbeda ditampilkan pada Gambar IV.3.




**Gambar IV.2. Kotak dialog ketika nilai *hash* sama**




**Gambar IV.3. Kotak dialog ketika nilai *hash* berbeda**

Berikut ini adalah kode implementasi untuk mendapatkan *message digest* dari sebuah pesan masukan yang bertipe *String*.

```
public static String generateMD5(String
pesan) {

String original = pesan;
StringBuffer sb = new StringBuffer();
MessageDigest md;
try {
   md=MessageDigest.getInstance("MD5");
   md.update(original.getBytes());
   byte[] digest = md.digest();
   for (byte b : digest) {
      sb.append(String.format("%02x", b &
0xff));}
} catch (NoSuchAlgorithmException e){
   e.printStackTrace();
}
return (sb.toString());
}
```

Berikut ini adalah kode untuk mendapatkan konten dari sebuah URL laman *website* yang akan mengembalikan konten laman dalam bentuk HTML. Untuk mendapatkan konten dari laman tersebut dibutuhkan koneksi untuk mencapai server dari laman tersebut.

```
public static String URLtoHTML(String url) {

String content = null;
URLConnection connection = null;
try {
connection =  new URL(url).openConnection();
scanner=newScanner(connection.getInputStream
());
scanner.useDelimiter("\\Z");
content = scanner.next();
}catch ( Exception ex ) {
    ex.printStackTrace();
}
return(content);
}
```

Berikut adalah kode implementasi untuk penyimpanan nilai *hash* ke dalam *file* eksternal untuk nantinya dibandingkan dengan nilai *hash* dari pengaksesan laman yang sama pada waktu yang berbeda.

```
btnSaveMD.addActionListener(newActionListene
r(){

private BufferedWriter o;
public void actionPerformed(ActionEvent e) {

final JFileChooser fc = new JFileChooser();
int returnVal =
fc.showSaveDialog(getParent());
if (returnVal ==
JFileChooser.APPROVE_OPTION) {
File file = fc.getSelectedFile();

try {
o = new BufferedWriter(newFileWriter(file));
o.write(lblHash.getText());
o.flush();
o.close();
JOptionPane.showMessageDialog(getParent(),
"File berhasil disimpan");
} catch (IOException e1) {
      e1.printStackTrace();
}}}});
```

Berikut ini adalah kode implementasi untuk pembangkitan nilai *hash* dari *file* eksternal untuk dibandingkan dengan nilai *hash* pada bagian pertama. Nilai *hash* yang dibangkitkan ini menggambarkan konten dari laman *website* pada waktu pengaksesan yang sebelumnya.

```
JButton btnLoadMd = new JButton("Load MD");
btnLoadMd.addActionListener(newActionListene
r() {

public void actionPerformed(ActionEvent
arg0) {

JFileChooser fc = new JFileChooser();
FileInputStream inputStream = null;
int returnVal =
fc.showOpenDialog(getParent());
File file = null;
if (returnVal==JFileChooser.APPROVE_OPTION){
   file = fc.getSelectedFile();
   try {

inputStream=new FileInputStream(file);
String everything =
IOUtils.toString(inputStream);
label_1.setText(everything);
      } catch (FileNotFoundException e) {
      e.printStackTrace();
      } catch (IOException e) {
      e.printStackTrace();
      }}
}
});
```

Berikut ini adalah kode implementasi untuk perbandingan kedua nilai *hash* yang dimiliki.

```
JButton btnBandingkan = new
JButton("Bandingkan");

btnBandingkan.addActionListener(new
ActionListener() {
public void actionPerformed(ActionEvent
arg0) {

if(lblHash.getText().equals(label_1.getText(
))){
JOptionPane.showMessageDialog(getParent(),
"Konten Laman Sama");
} else {
JOptionPane.showMessageDialog(getParent(),
"Konten Laman Berbeda");
}}
});
```

## V. PERCOBAAN

Percobaan dilakukan sebanyak dua kali. Percobaan pertama dilakukan dengan tidak mengubah konten dari laman ketika sebuah halaman kedua kalinya diakses. Percobaan kedua dilakukan dengan mengubah konten dari laman ketika halaman tersebut ingin diakses kedua kalinya.

Sebelum diubah, konten HTML dari laman yang digunakan untuk percobaan ditunjukkan di bawah ini.

```
<body>
	<h1>Ini adalah laman untuk tugas makalah kriptografi</h1>
</body>
```

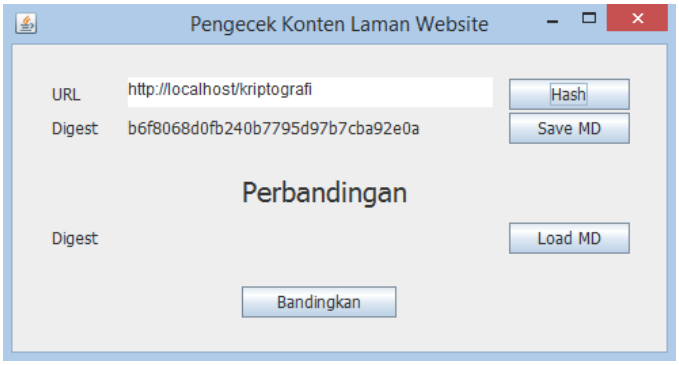
Nilai *hash* MD5 dari laman tersebut ditampilkan pada Gambar V.1.


**Gambar V.1. Nilai *hash* laman percobaan sebelum kontennya diubah**

Setelah diubah kontennya, teks HTML laman percobaan adalah sebagai berikut.

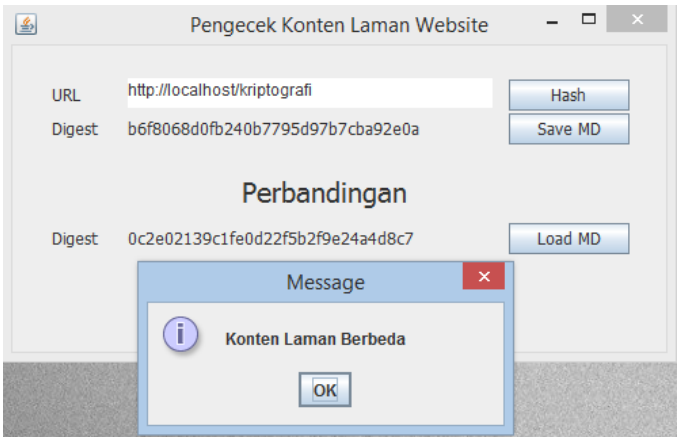
```
<body>
	<h1>Ini adalah laman untuk tugas makalah kriptografi</h1>
	<h2>Tambahan informasi baru</h2>
</body>
```

Konten HTML tersebut menghasilkan nilai *hash* MD5 yang ditampilkan pada Gambar V.2.



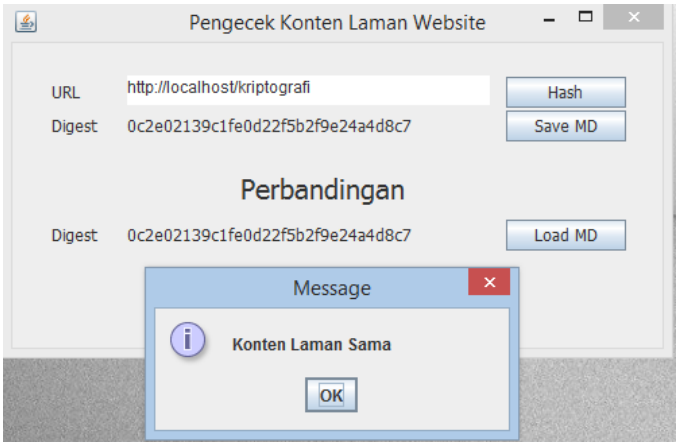

**Gambar V.2. Nilai *hash* laman percobaan setelah kontennya diubah**

Hasil percobaan menunjukkan bahwa ketika kontennya diubah maka kotak dialog akan menyatakan bahwa kontennya sudah berubah, dilihat dari nilai *hash*-nya yang berbeda, seperti yang dapat dilihat pada Gambar V.3.




**Gambar V.3. Hasil percobaan dengan konten yang diubah**

Ketika percobaan dilakukan dengan menggunakan konten yang tidak diubah, kedua *message digest* bernilai sama. Hal ini menyebabkan algoritma perbandingan mengembalikan kotak dialog yang menyatakan bahwa kedua konten yang dibandingkan sama, seperti yang ditunjukkan pada Gambar V.4.




**Gambar V.4. Hasil percobaan dengan konten yang tidak diubah**

## VI. KESIMPULAN

Pengecekan perubahan konten dari sebuah laman *website* untuk memastikan konten yang dirujuk pada sebuah rujukan belum berubah dapat dilakukan dengan menggunakan algoritma MD5. Hal tersebut dapat dilakukan dengan membandingkan nilai *hash* MD5 dari kedua pengaksesan situs tersebut. Jika nilai *hash* MD5 dari kedua pengaksesan tersebut menghasilkan nilai *hash* yang sama maka dapat diambil kesimpulan bahwa konten dari situs tersebut belum berubah. Selain untuk pengecekan integritas dari *file*, algoritma MD5 merupakan salah satu teknik efektif untuk mengecek integritas dari konten laman *website*.

## REFERENSI


[1] R. Munir, "Algoritma MD5 (2013).ppt", Informatics Engineering Institute Technology of Bandung, 2013

[2] S.G. Tanuwijaya, "Penerapan MD5 untuk Pencarian File Duplikasi", Informatics Engineering Institute Technology of Bandung, 2013.