Pengujian Steganografi untuk *Compresser* Data

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*—Steganografo merupakan metoda untuk menyisipkan suatu teks kedalam sebuah gambar sehingga teks tersebut tidak diketahui isinya. Cara penyisipannya dengan cara merubah teks tersebut dalam bentu ASCII lalu di jadikan biner, setelah itu dibaca sebuat gambar digital setiap pixelnya dimana setiap pixel dibaca dalam biner dan di ubah satu atau dua bit terakhirnya sesuai dengan teks yang disisipkan. Dengan menyisipkan di bit terakhir ini tidak akan mengubah gambar secara signifikan sehingga tidak diketahui apakah gambar memiliki atau terdapa teks di dalamnya atau tidak. Teknik ini digunakan untuk merahasiakan sebuah teks agar tidak di ketahui oleh orang lain.**

Dengan teknik penyisipan teks dalam gambar menggunakan biner dari gambar digital, penulis akan mencoba untuk memanfaatkannya untuk mengkompores data dengan cara membaca bit dari sebuah file lalu menyisipkanya kedalam setiap bit dari gambar. Pada dasarnya gambar secara otomatis akan rusak, tapi penulis ingin menguji apakan dengan teknik tersebut bisa mengecilkan penggunaan memory disk untuk menyimpan beberapa buah file data.

***Index Terms*—file, steganografi, bit, pixel,RGB, bitmap**

I. PENDAHULUAN

Dengan perkembangannya teknologi, hingga saat ini banyak kegiatan – kegiatan kita yang menggunakan komputer. Hampir setiap aktifitas kita berkaitan dengan komputer mulai dari materi pembelajaran dalam bentuk digital, game, musik, video, gambar, pembuatan laporan dan berbagai program yang membantu kita dalam mengerjakan pekerjaan kita sehari-hari. Tentunya untuk menggunakan komputer kita membutuhkan data yang akan di oleh untuk kegiatan tersebut. Data tersebut akan di simpan dalam sebuah media yang biasa kita sebut "*hardisk*" atau pun "*flash disk*". Namun tentunya media tersebut memiliki daya tampung yang terbatas, pada tahun 1956 IBM(salah satu perusahaan komputer yang terkenal di dunia) meluncurkan hardisk pertama yang didunia yang hanya memiliki daya tampung 5Mega Byte.

Namun pada saat ini "*hardisk*" bisa memiliki daya tampung hingga 1Tera Byte. Walaupun daya tampung tersebut sudah sangat besar dan bisa menyimpang sangat banyak data yang dimiliki oleh penggunanya, tetapi manusia tetap saja masih merasa daya tampung tersebut masih belum cukup. Banyak sekali data yang disimpan dikarenakan semakin banyaknya penggunaan komputer sehingga data yang disimpan atau pun di perlukan juga bertambah.

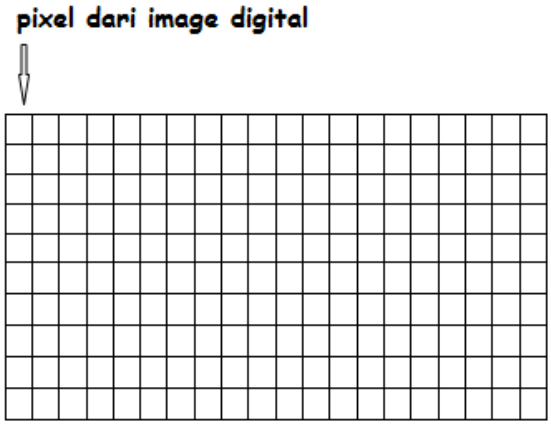
Untuk mengatasi hal tersebut banyak sekali orang-orang dari informatika menggunakan ilmunya untuk meminimalisasi penyimpanan data dalam memory baik itu "*hardisk*" ataupun media penyimpanan lainnya. Dan akhirnya ditemukan cara untuk menghemat daya tampung "*hardisk*" yaitu dengan mengkompressnya kedalam suatu media sehingga menjadi suatu kesatuan yang memiliki daya tampung lebih kecil. Banyak sekali cara dan program yang dibuat untuk mengompres data agar bisa lebih efisien untuk segi penyimpanan data.

Pada kesempatan ini penulis akan merancang metoda mengompressan data yang di harapkan akan bisa efisien untuk digunakan dalam penyimpanan data. Penulis memanfaatkan maka kuliah kriptografi yaitu pada materi tentang "Steganografi". Namun metoda ini belum diketahui apakah akan lebih efisien atau malah akan memperbesar penggunaan media penyimpanan pada nantinya.

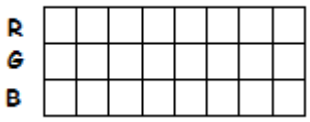
II. DASAR TEORI

Steganografi adalah teknik untuk menyembunyikan teks tanpa diketahui oleh orang lain teks tersebut ada atau tidak. Penyisipannya dengan membaca bit dari setiap pixel gambar dan diganti bit terakhirnya dengan bit teks yang telah di ubah menjadi biner. Semakin besar atau panjang teks tersebut semakin besar juga gambar yang di butuhkan.

Teks yang akan disisipkan diubah semuanya menjadi ASCII lalu di jadikan biner agar bisa disisipkan kedalam bit gambar. Sedangkan gambar terdiri dari sekumpulan pixel yang terdiri dari 3 bagian yang itu Red, Green, Blue. Setiap RGB memiliki delapan bit yang bisa disisipkan oleh teks tersebut. Seperti pada gambar berikut



Gambar 1 ilustrasi gambar digital



Gambar 2 ilustrasi bit dalam 1 pixel

Dengan demikian pada teknik stegano kita bisa menyisipkan 3 bit atau 6 bit dalam setiap pixelnya jika disisipkan di akhir atau dua terakhir bit saja yang digunakan. Hal ini akan membuat teks tersebar dalam gambar digital tanpa merusak gambar secara signifikan sehingga secara kasat mata teks tidak tidak terlihat dan tidak merusak gambar.

Teks yang disisipkan di ubah menjadi biner yang dimana setiap teks yang sebelumnya adalah ASCII dijadikan biner 4 bit. Jika hanya 64 ASCII nya maka akan dijadikan 8 bit yaitu 01000000 agar bisadi baca kembali dengan mudah untuk proses dekripsinya atau di keluarkan dari gambar. Lalu setiap bit dari teks tersebut di sisipkan satu persatu secara random yang tetap jika seednya diketahui atau tetap juga.

Prinsip penyisipan inilah yang akan digunakan untuk kompreser data atau file. Jadi setiap file dibaca bitnya dan disisipkan satu persatu-satu di setiap element gambar, namun konsep ini akan merusak gambar secara total karena menggunakan semua bit gambar misalkan gambar ukuran 600x800. Jadi gambar tersebut bisa menampung 600x800x3x8 = 11.520.00 bit sehingga ini di prediksi akan memperkecil ukuran file. Namun hal ini akan dicoba apakah akan memperkecil atau malah akan memperbesar ukuran file karena media yang digunakan adalah bitmap yang tentunya ukurannya cukup besar dibandingkan dengan gambar-gambar format lainnya.

III. PEMBACAAN FILE

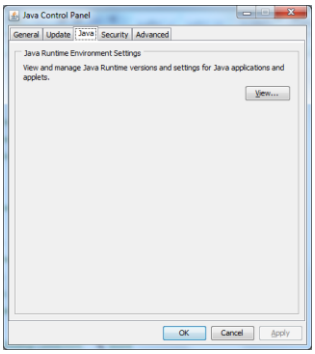
Sebelumnnya telah disebutkan dan di ilustrasikan bagaimana sebuah gambar digital dalam komputer yang nantinya akan di manfaatkan sebagai media untuk penyimpanan. Gambar digital memiliki kurang lebih 11juta bit untuk gambar berformat "*Bitmap*" dengan dimensi 800x600 pixel.

Setiap file dalam komputer bisa dibaca dalam bentuk stringnya ataupun dibaca dalam bentuk bit nya. Pada saat ini penulis memanfaatkan dari "*library java*" untuk membaca bit dari sebut file seperti sebagai berikut.

```
// 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;
}
```

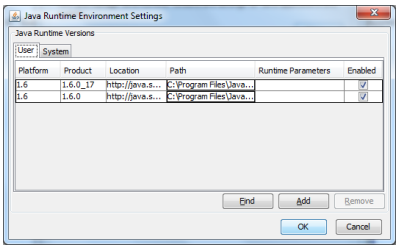
pembacaan file, file yang dibaca tidak bisa terlalu besar jika file telah lebih besar dari kurang lebih 64 MB maka harus di setting terlebih dahulu memory javanya dengan cara :

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.

Pada rancangan ini diharapkan akan menggunakan 8 file, namun dibatasi penggunaan file adalah 10MB jadi disetting memory javanya menjadi 100MB untuk persoalan ini. Karena media yang akan digunakan sangat besar dan untuk komputer standard hanya bisa membuat maksimal 9999x4992 pixel file bitmap dan saat file tersebut di buat, akan memakan memory komputer yang cukup besar sehingga akan agak lama untuk membuat dan menyimpanny dalam direktory.

Dimisalkan file yang dibuka adalah typenya .ppt dengan ukuran kurang lebih 10MB maka setelah dibaca bytenya, file tersebut memiliki panjang file 11.187.200 byte. Dikarenakan masih dalam bentuk byte dan yang akan diisisipkan kedalam gambar adalah dalam bentuk bit maka byte tersebut dirubah dalam bentuk bit.

Ada sedikit masalah dalam byte yang dibaca, jadi byte yang dibaca memiliki rentang dari -128 sampai 127, jadi agar rentangnya positif dan mudah untuk dijadikan 8bit maka di tambah dengan 128 setiap bitnya dan nanti saat di “*export*” filenya akan dikurang kembali dengan 128.

Jadi misalkan salah satu bytenya ada -48 maka akan ditambah 128 dan akan menjadi 80 dan jika di jadikan 8bit akan menjadi 01010000. 8 bit tersebut lah yang nantinya akan disisipkan kedalam gambar bitmap yang telah disiapkan sebelumnya.

Tadi file 10MB dibaca bytenya maka diperoleh panjang atau jumlah byte yang dibaca adalah 11 juta lebih, namun setelah dijadikan 8bit maka:

8bit/byte x 11juta byte = 88 juta bit

Jika dibulatkan akan kurang lebih membutuhkan 90 juta bit. Dikarenakan satu pixel terdapat 3 tempat yang bisa di isi agar bisa diisi oleh 8 file maka dibutuhkan

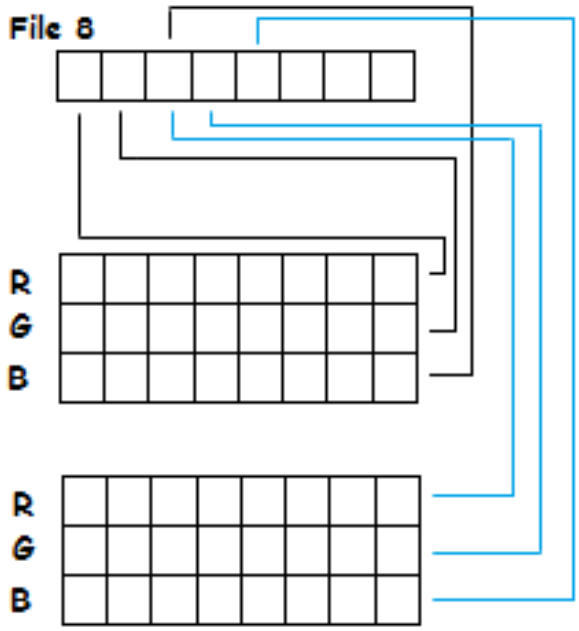
90juta bit : 3pixel/bit = 30 juta pixel

Jadi, jika ditotalkan pixel yang dibutuhkan, diperkirakan dibutuhkan gambar yang kurang lebih berukuran 6000x5000 pixel. Pada intinya dibutuhkan gambar yang memiliki total pixel 30juta pixel.

IV. PENYISIPAN BIT

Seperti yang telah dibahas sebelumnya sebuah kita harus bisa membaca bit file yang akan di baca yang nanti akan disisipkan dan kita juga harus bisa membaca gambar digital dalam bentuk RGB ditiap pixelnya. Oleh karena itu gambar harus dalam format “bitmap” saja.

Pada makalah ini di rancang maksimal file yang akan disisipkan adalah delapan file dengan ukuran 10MB dan harus menyetting maksimum memory javanya terlebih dahulu. Setelah itu untuk awalnya file pertama ditelakkan di urutan bit-1 dalam gambar, urutan ke dua juga dilekakan dalam urutan bit ke-2 pada gambar dan begitu seterusnya hingga setiap file bisa di alokasikan dengan baik dan teratur didalam gambar digital tersebut seperti di ilustrasikan pada gambar berikut ini untuk penyisipan pada salah satu pixel pada gambar digital.



Gambar 5 ilustrasi penyisipan file ke-8 yang dimasukkan kedalam bit trakhir dalam pixel

Untuk segi penggunaan memory sebelum dan sesudah file disisipkan tidak merubah ukuran gambar digital tapi gambar digital yang bisa dijadikan media berukuran sangat besar untuk file yang disisipkan dengan ukuran 10MB. file tersebut disisipkan dengan Urutan R,G,B untuk tiga bit pertama file dan tiga bit kedua dimasukkan ke R,G,B pixel berikutnya yang dimiliki oleh gambar digital yang dijadikan media tersebut.

Penyisipan file kedalam gambar digital tidak akan merubah ukuran file gambar digital, tapi tentunya semakin besarnya file yang disisipkan semakin besar juga resolusi gambar digital yang akan dibutuhkan. File digital tersebut dimasukkan kedalam gambar digital secara random sehingga susah untuk dipecahakan. Walaupun tidak mengurangi atau menambah ukuran file tapi gambar digital yang dijadikan media akan rusak sehingga orang yang melihat gambar tersebut bisa saja mengetahui ada sesuatu file yang disisipkan dengan gambar digital tersebut, namun akan sulit untuk mengetahui gambar tersebut.

Oleh karena itu dibutuhkan fungsi random yang akan menentukan pixel berikutnnya yang dijadikan tempat untuk menyisipkan data tersebut. Random tersebut harus bisa menghasilkan angka yang sama setiap kali dimasukkan suatu angka yang sama juga. Angka tersebut disebut "*seed*". "*Seed*" dapat diperoleh dari masukan user ataupun dari pengolahan kunci yang diinput menjadi angka.

Setelah seed diperoleh, dalam java digunakan fungsi random dalam java untuk memperoleh "*array of integer*" yang merupakan urutan dari penyisipan file dalam pixel gambar digital tersebut. Dalam java dapat digunakan fungsi random seperti pada fungsi berikut.

```
public static ArrayList<Integer>
CreateRandomizeArray(int size, int
seed) {

//Start random number
Random rand = new Random(seed);


//Create variable List of interger
ArrayList<Integer> array = new
ArrayList<Integer>();


//Looping
for (int i = 0; i < size; ++i) {
            array.add(i);
        }


// shuffle array
java.util.Collections.shuffle(array,
rand);


return array;

}
```

Jika fungsi tersebut dijalankan maka akan terbentuk beberapa angka random yang unik dan teracak. Awalnya dibuat serangkaian angka yang berurutan. Lalu diacak sesuai dengan angka random yang keluar. Sehingga akan terbentuk serangkaian angka random yang nantinya akan digunakan untuk menentukan pixel mana dari gambar digital yang merupakan media dari penyisipan yang akan dijadikan tempat.

Tentunya jika kita menyisipkan data atau file dalam sebuah gambar digital untuk bisa dikembalikan lagi ke bentuk semua, filer atau data tersebut harus diketahui ekstensinya, oleh karena itu sebelum dimasukkan ke dalam gambar file tersebut diubah formatnya menjadi serangkaian bit yang tersediri atas header dan isi, seperti pada gambar berikut

Header	1111	Ekstensi	111111	Isi

Nama File

Gambar 6 Format file yang akan di masukkan

Header berisi seberapa panjang isi file yang akan dibaca, formatnya adalah satu angka sama dengan 4 bit hingga ditemukan 1111. Contoh:

123 = 0001 0010 0011

Jadi setiap angka diubah jadi bit dan tidak mungkin ada satu angka yang jika dijadikan bit akan berubah menjadi 1111. Header berisi dua buah angka yang dibatasi oleh bit 1111, angka pertama adalah untuk panjang nama file dan yang kedua adalah angka untuk panjang isi file sebenarnya yang disisipkan

Ekstensi berisi serangkaian huruf yang diprediksi hanya huruf saja oleh karena itu kisarannya 1-26. Jadi untuk satu huruf dijadikan 5bit. Dan tidak akan mungking huruf dengan urutan ke 11111 atau 31. Contoh :

Pdf = 100000 000100 000110

Dan ditutupkan dengan 111111. Lalu berikutnnya akan diisi dengan 8bit file yang akan disisipkan dan sisanya ditambah dengan nama file. Kerusakan gambar digital akan menyebabkan file tidak bisa dibaca ataupun dibuat kembali kebentuk semula.

Setelah format file yang akan disisipkan telah dipenuhi maka kita tinggal memanfaatkan algoritma steganografi untuk memasukkan file kedalam gambar digital sesuai dengan angka random yang telah kita peroleh sebelumnya.

```
//class for image
class RGB{
        String red;
        String green;
        String blue;
    }

//method to combine image n file
public static RGB[] Import(RGB[]
rgb,String file,int number,int[]
np)
{
RGB[] x= new RGB[rgb.length];
int j = 0;

for(int i = 0; i <
file.length();i++)

  {

  String temp = "";

  temp +=
x[np[j]].red.substring(0,number-
2) +
x[np[j]].red.substring(number-1,
7);

 x[np[j]].red = temp;
  i++;


//if file not yet finished insert
  if (i < file.length()){
    temp +=
x[np[j]].green.substring(0,number-2)
+ x[np[j]].green.substring(number-1,
7);

    x[np[j]].green = temp;
    i++;

//if file not yet finished insert
    if(i < file.length()){
      temp +=
x[np[j]].blue.substring(0,number-2)
+ x[np[j]].blue.substring(number-1,
7);

      x[np[j]].blue = temp;
      i++;
      }
    }

  }

    return x;
}
```

Data yang dibaca dari gambar digital yang dijadikan media dimasukkan kedalam class RGB dimana bit tiap warna sebelumnya telah dijadikan string biner.

Pada fungsi ini terdapat 4 parameter yang harus dimasukkan ya itu pertama

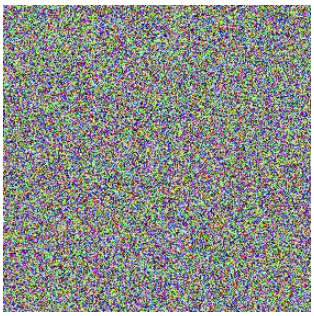
1. rgb
   merupakan variabel yang berikan data RGB gambar digital yang akan dijadikan media.

2. file
   merupakan variabel yang berisi format file yang sebelumnya telah dijelaskan yang telah lengkap dengan headernya dalam bentung string.

3. number
   merupakan urutan file yang akan dimasukkan, berkisar antara 1-8.
4. np
   merupakan hasil dari fungsi **`CreateRandomizeArray()`** yang menghasilkan "*array of integer*".

Hasil dari fungsi import adalah data berformat RGB

yang telah di edit atau di masukkan file. Jika ke delapan file tersebut dimasukkan maka gambar akan menjadi rusak total seperti gambar dibawah



Gambar 7 gambar yang telah di masukkan data

Setelah dimasukkan ukuran file tidak berubah namun setelah dihitung ukuran total dari file dan dibandingkan dengan ukuran media yang digunakan, ternyata ukurannya tidak jauh beda dan ukuran gambar akan lebih besar jika terjadi ada salah satu gambar digital yang besar sedangkan yang lainnya tidak terlalu besar dimana rentang ukuran filenya jauh.

V. PENARIKAN BIT

Setelah file dimasukkan maka untuk bisa membaca atau menggunakan file tersebut lagi kita harus meng"*export*" file tersebut dari gambar digital tersebut. Prinsip pembacaannya sangat simple, Yaitu membagi RGB menjadi 8 bagian, lalu diparsing kembali sesuai dengan format sebelumnya.

```
//method to export file from image

public static String[] Export(RGB[],
int[] np) {

String[] x= new
String[rgb.length*3];

//intialization x
for(int i = 0;i<8;i++){
        x[i] = "";
    }

for(int i = 0; i < rgb.length ; i++)
{
  x[0] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;


  x[1] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[2] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[3] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[4] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[5] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[6] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;

  x[7] += rgb[np[i]].red +
rgb[np[i]].green + rgb[np[i]].blue;


    }

    return x;
}
```

Setelah file diperoleh dari gambar kita harus melakukan parsing lagi untuk bisa memperoleh file utuh lagi. Awalnyakita harus membaca headernya lalu diparsing headernya menjadi dua angka yaitu angka untuk nama file dan angka kedua untuk ukuran bit file yang dimasukkan. Setelaih itu dari bit 1111 hingga ditemukan 11111 dimasukkan menjadi ekstensi file. Dan berikutnya dibaca isi file sesuai dengan panjang bit file yang telah dibaca sebelumnya. Dan terakhir membaca nama file sesuai dengan panjang nama file yang telah dibaca sebelumnnya. Dengan demikian kita bisa membuat file yang dimasukkan ke gambar digital sebelumnya lagi, tanpa harus berubah karena ada bit tambahan yang disebebkan ukuran file tidak sama.

VI. KESIMPULAN

gambar digital berhasil digunakan sebagai media untuk penyisipan data. Tetapi setelah dicoba dan dianalisi ternyata dikarenakan ukuran media yang diperlukan untuk penyisipan data file yang diperlukan lebih besar dari pada ukuran total dari delapan file yang akan disisipkan, file memiliki total 80MB tapi gambar digital yang diperlukan memiliki ukuran kurang lebih 87MB. Jadi apa yang diharapkan yaitu menjadikan gambar digital sebagai media untuk menyisipkan file tidak efektif, tapi untuk merahasiakan file kedalam gambar digital cukup baik, karena ada sebanyak n! (n = banyak bit file dibagi 3) kemungkinan untuk menyusun kembali bit file menjadi file yang utuh lagi.

REFERENSI

[1] http://aneh22.blogspot.com/2009/01/hardisk-pertama-di-dunia-dengan-berat.html
[2] http://aneh22.blogspot.com/2009/01/hardisk-pertama-di-dunia-dengan-berat.html
[3] http://www.duckware.com/pmvr/howtoincreaseappletmemory.html
[4] http://stackoverflow.com/questions/171205/java-maximum-memory-on-windows-xp
[5]

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