

IMPLEMENTASI SANDI HILL DALAM PEMBUATAN PROGRAM APLIKASI PENYANDIAN CITRA

Ardian Adi Wibowo¹

Suhartini²

^{1,2}Fakultas Teknik Industri, Universitas Gunadarma
Jalan Margonda Raya 100.

¹ardian_ia01@yahoo.com, ²tini @staff.gunadarma.ac.id

ABSTRAK

Sandi Hill merupakan salah satu teknik penyandian teks. Dalam penelitian ini, pemakaian sandi Hill diperluas dari teks menjadi citra 24 bit. Matriks yang dipakai adalah matriks yang berordo 2x2 dan 3x3. Hasil percobaan menunjukkan bahwa sandi Hill cocok digunakan untuk enkripsi citra dengan variasi nilai Reda Green Blue (RGB) antar piksel berdekatan yang tinggi (seperti foto), tapi tidak cocok untuk citra dengan variasi nilai RGB yang rendah (seperti gambar kartun) karena pola citra asli masih tampak dalam citra sandi. Sandi Hill juga memiliki kelemahan dalam hal tidak tunggalnya matriks kunci yang dapat dipakai. Akan tetapi untuk pemakaian biasa, dengan pemilihan matriks kunci yang baik, sandi Hill dapat dipakai untuk penyandian karena hanya melibatkan operasi matriks biasa sehingga prosesnya relatif cepat.

Kata Kunci : Sandi Hill, Citra, RGB.

PENDAHULUAN

Dewasa ini, kriptografi dirasakan semakin penting. Keamanan pengiriman informasi melalui komputer menjadi bagian yang tak terpisahkan dalam kehidupan sehari-hari. Seiring dengan peningkatan kepentingannya, banyak metode-metode yang ditemukan maupun diperluas penggunaannya. Diantara metode-metode tersebut terdapat metode yang hanya membutuhkan operasi matematika sederhana, tetapi juga terdapat metode yang melibatkan teori yang rumit dan sulit implementasinya. Salah satu metode yang sangat sederhana dalam kriptografi adalah dengan melakukan penggeseran karakter dalam abjad. Jika enkripsi dilakukan dengan menggeser 3 huruf ke kanan, maka huruf A disandikan dengan D, huruf B dengan E dan seterusnya. Metode tersebut memiliki kelemahan, yaitu apabila satu huruf tertebak, maka semua huruf akan tertebak pula.

Sandi Hill merupakan metode yang diciptakan untuk menutupi kelemahan tersebut. Penyandian dengan sandi Hill dilakukan dengan memanfaatkan operasi matriks biasa. Penyandian dilakukan pada tiap blok teks yang berukuran sama dengan ordo

matriks yang digunakan. Sebagai perluasannya, dalam aplikasi ini sandi Hill diimplementasikan untuk menyandikan sebuah citra. Ini dimungkinkan mengingat sebuah citra merupakan deretan piksel-piksel yang komponen R (Red), G (Green) dan B (Blue) nya merupakan bilangan-bilangan bulat sehingga dapat dioperasikan dalam sebuah matriks.

Penelitian ini dimaksudkan untuk mengimplementasikan sandi Hill dalam penyandian citra yang dapat mendukung keamanan data terutama data citra tersebut.

Dalam pembuatan aplikasi ini, dibatasi format file citra hasil penyandian. Format file yang diperkenankan pada aplikasi ini adalah format png. Sedangkan matriks yang dipakai sebagai matriks kunci adalah matriks bujur sangkar berordo 2x2 dan 3x3 dengan elemen bilangan bulat. Untuk ukuran citra yang dapat dioperasikan disini adalah citra dengan ukuran maksimum 640 x 480.

TINJAUAN PUSTAKA

Sandi Hill. Merupakan salah satu algoritma yang dapat digunakan untuk melakukan enkripsi (penyandian) pada satu teks. Penyandian pada

menggunakan Sandi Hill memanfaatkan matriks sebagai kunci penyandi. Untuk mengembalikan teks yang sudah disandikan, matriks kunci dicari terlebih dahulu inversenya. Matriks inverse tersebut yang akan dijadikan kunci pembuka. Yang perlu diperhatikan adalah matriks kunci harus mempunyai inverse. Untuk memiliki inverse, determinan matriks kunci harus relatif prima dengan modulo tertentu.

Langkah-langkah untuk menyandikan teks dengan sandi Hill: (1). Pilih matriks bujur sangkar A yang dipakai sebagai kunci; (2). Transformasikan tiap huruf dalam teks ke bilangan bulat yang sesuai ($A=0, B=1, \dots, Z=25$); (3). Kelompokkan barisan angka yang didapat kedalam beberapa blok vektor p yang panjangnya sama dengan ukuran matriks A ; (4). Hitung $c = A.p \pmod{256}$ untuk tiap vektor p ; (5). Kembalikan tiap angka dalam vektor sandi c ke huruf yang sesuai untuk mendapatkan teks sandi..

METODE PENELITIAN

Penelitian ini diawali dengan analisis kebutuhan sistem perangkat lunak, kegiatannya antara lain mempelajari dan menganalisis algoritma Sandi Hill terhadap kemungkinan aplikasinya dalam proses penyandian data citra, menentukan kebutuhan awal bagi sistem perangkat lunak yang akan dikembangkan.

Langkah selanjutnya adalah membuat model bagi rancangan perangkat lunak menggunakan pendekatan model berorientasi pada objek, yang nantinya akan diperoleh sejumlah kelas (*Class*) yang akan digunakan dalam program berorientasi pada objek.

Berdasarkan kelas yang terbentuk dari hasil rancangan, maka akan dilakukan pemrograman, menggunakan bahasa pemrograman yang mendukung pemrograman berorientasi pada objek, dalam hal ini menggunakan C++.

Setelah proses penulisan program, langkah selanjutnya adalah uji coba program dan dilakukan analisis terhadap hasil uji coba tersebut.

PEMBAHASAN

Implementasi Sandi Hill. Dalam penelitian ini, penggunaan Sandi Hill diperluas untuk melakukan penyandian citra. Algoritma untuk menyandikan citra dengan Sandi Hill adalah sebagai berikut:

Pilih matriks bujur sangkar A yang dipakai sebagai kunci.

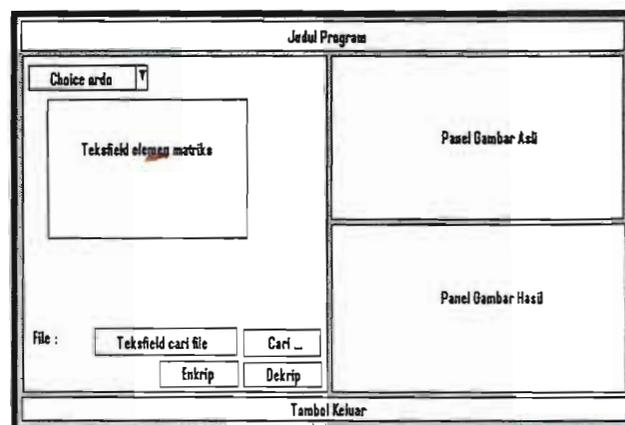
Ambil RGB dari masing-masing piksel pada citra.

Kelompokkan barisan angka yang didapat ke dalam beberapa blok vektor p yang panjangnya sama dengan ukuran matriks A .

Hitung $c = A.p \pmod{256}$ untuk tiap vektor p

Bentuk citra baru dari matriks hasil perkalian matriks.

Rancangan Output. Secara umum tampilan output dari program yang dibuat terlihat seperti pada gambar 1.



Gambar 1. Rancangan tampilan aplikasi Penyandian Citra

Alur Program. Secara umum jalannya program aplikasi Penyandian Citra adalah sebagai berikut; Ketika program pertama kali dijalankan, program akan melakukan inialisasi *aplet*. Program akan menampilkan *frame* beserta objek-objek yang ada di dalamnya.

Setelah *frame* ditampilkan, pengguna dipersilahkan untuk memasukkan berbagai masukan yang diperlukan dalam proses penyandian citra. Masukan tersebut diantaranya matriks kunci penyandi. Matriks kunci penyandi haruslah matriks yang memiliki determinan dan determinan dari matriks tersebut memiliki inverse dalam modulo 256. Masukan lain yang dibutuhkan yaitu file gambar yang ingin dilakukan operasi penyandian. Citra yang dapat dioperasikan pada aplikasi ini hanyalah citra yang memiliki ukuran maksimal 640 x 480.

Setelah itu, pengguna dapat melakukan penekanan pada tombol enkrip atau dekrip. Sebelum proses dijalankan, program akan memeriksa masukan terlebih dahulu. Apabila masukan dari pengguna salah, program akan menampilkan pesan kesalahan melalui *messageBox*. Pesan kesalahan yang muncul

berbeda-beda, tergantung kesalahan apa yang terjadi. Apabila masukan sudah benar, program akan melakukan proses penyandian.

Setelah proses penyandian selesai, program akan menampilkan citra hasil penyandian. Sampai disini pengguna dapat keluar dari aplikasi dengan menekan tombol keluar atau dapat mengulang proses penyandian dengan matriks kunci atau citra yang berbeda.

Kelas-kelas Pada Aplikasi Penyandian Citra.

Kelas TugasAkhir : Kelas ini merupakan kelas utama dari aplikasi Penyandian Citra ini. Pada kelas ini penulis membuat *frame* dengan judul *frame* "Tugas Akhir" dan dengan ukuran 700 x 500. Setelah itu *Layout* dari *frame* tersebut diset menjadi *BorderLayout*. Setelah itu, diletakkan panel judul di sebelah atas *frame*, panel tengah di sebelah tengah *frame* dan panel keluar di sebelah bawah *frame*. Ketiga *panel* tersebut masing-masing merupakan objek dari kelas *PanelJudul*, *PanelTengah* dan *PanelKeluar*.

Kelas PanelJudul : Kelas ini merupakan cetak biru untuk membuat objek *panel* yang dibuat pada kelas *TugasAkhir*. Kelas ini digunakan untuk menuliskan judul program pada panel judul, dimana penulisannya memanfaatkan *Label* yang telah disediakan bahasa pemrograman java. Hal tersebut terlihat dari potongan program berikut :

```
Label teks = new Label ("IMPLEMENTASI SANDI
HILL PADA PENYANDIAN CITRA");
```

Pertama, dibuat objek teks yang dideklarasikan sebagai *Label*. Setelah objek teks dibuat, objek teks tersebut diletakkan pada *panel* *PanelJudul*.

```
add(teks);
```

Kelas PanelTengah : Kelas ini merupakan cetak biru untuk membuat objek *panel* yang dibuat pada kelas *TugasAkhir*. Pada kelas ini dibuat objek *panelproses* dan *panelkanvas* yang masing-masing dibentuk dari kelas *PanelProses* dan *PanelKanvas*. Untuk menentukan tata letak *panel-panel* tersebut, digunakan *GridLayout* dengan jumlah baris 1 dan jumlah kolom 2. *Panel* *panelproses* diletakkan di sebelah kiri sedangkan *panel* *panelkanvas* diletakkan di sebelah kanan.

Kelas PanelProses : Kelas ini merupakan cetak biru untuk membuat objek *panel*, dimana pada objek tersebut akan diletakkan *panel* *panelmatriks*

dan *panel* enkrip. Objek *panel* *panelmatriks* dibentuk dari kelas *PanelMatriks*, sedangkan objek *panel* enkrip dibentuk dari kelas *Enkripsi*. Untuk menentukan tata letak dari *panel-panel* tersebut, digunakan *BorderLayout*. Objek *panelmatriks* diletakkan di tengah, sedangkan objek enkrip diletakkan di sebelah bawah.

Kelas PanelMatriks : Kelas ini merupakan cetak biru untuk membuat *panel*, dimana pada *panel* tersebut akan diletakkan objek *panelordomatriks*, *panelemenmatriks* dan *panelcarigambar*. Ketiga objek tersebut merupakan objek *panel* yang dibentuk dari kelas *PanelOrdoMatriks*, *PanelElemenMatriks* dan *PanelCariGambar*. Untuk menentukan tata letak dari *panel-panel* tersebut, digunakan *BorderLayout*. Objek *panelordomatriks* diletakkan di sebelah atas, objek *panelemenmatriks* diletakkan di tengah dan objek *panelcarigambar* diletakkan di sebelah bawah.

Kelas PanelOrdoMatriks : Kelas ini merupakan cetak biru untuk membuat *panel*, dimana pada *panel* tersebut akan diletakkan objek *Label* *labelordo* dan objek *Choice* *dok*. Pada objek *dok* ditambahkan pilihan ordo matriks, yaitu "Matriks 2 x 2" dan "Matriks 3 x 3".

```
dok.addItem("Matriks 2 x 2");
dok.addItem("Matriks 3 x 3");
```

Apabila yang dipilih adalah "Matriks 2 x 2", variabel *ord* yang bertipe integer diberi nilai 2, sedangkan apabila yang dipilih adalah "Matriks 2 x 2", variabel *ord* yang bertipe integer diberi nilai 3. Setelah itu method *setOrdo* yang terdapat pada kelas *PanelElemenMatriks* akan dijalankan dengan parameter isi dari variabel *ord*.

```
parent.panelemenmatriks.setOrdo(ord);
```

Kelas PanelElemenMatriks : Kelas ini merupakan cetak biru untuk membuat sebuah objek *panel*, dimana pada *panel* tersebut akan diletakkan objek *mat* yang dibentuk dari kelas *TextField*. Objek tersebut bertipe array berdimensi 2 yang masing-masing menyatakan baris dan kolom. Pada kelas ini terdapat method *sembunyi*.

```
private void sembunyi()
{
    for(int i=0;i<3;i++)
        for(int j=0;j<3;j++)
            mat[i][j].setVisible(false);
}
```

```

}

```

Method tersebut akan menyembunyikan objek *mat* yang telah dideklarasikan sebelumnya. Selain itu juga terdapat method *setOrdo*.

```

public void setOrdo(int ordo)
{
    sembunyi();
    for(int i=0;i<ordo;i++)
        for(int j=0;j<ordo;j++)
            mat[i][j].setVisible(true);
}

```

Method tersebut digunakan untuk menampilkan kembali objek *mat* yang disembunyikan. *Method* ini menerima nilai melalui parameter *ordo* yang dikirim dari kelas *PanelOrdoMatriks*. Jadi, jumlah objek yang akan ditampilkan adalah sejumlah *ordo*ordo*.

Kelas PanelCariGambar : Kelas ini merupakan cetak biru untuk membuat sebuah objek *panel*, dimana pada *panel* tersebut akan diletakkan objek *labelcari*, *teksfield* dan *tombolcari* yang masing-masing dibentuk dari kelas *Label*, *TextField* dan *Button*. Untuk menentukan tata letak dari *panel-panel* tersebut, digunakan *BorderLayout*. Urutan peletakkannya yaitu; *labelcari* di sebelah kiri, *teksfield* di tengah dan *tombolcari* di sebelah kanan. Pada kelas ini terdapat penanganan *event* apabila tombol *tombolcari* ditekan.

```

public void actionPerformed() {
    try {
        FileDialog file = new FileDialog(
            new Frame(), "Buka File Gambar",
            FileDialog.LOAD);
        file.setVisible(true);
        if (file.getFile() != null) {
            f = file.getDirectory() + file.getFile();
            repaint();
            teksfield.setText(f);
        }
    }
    catch(Exception e) { System.err.print(e + "\n"); }
}

```

Apabila tombol *tombolcari* ditekan, program akan menampilkan sebuah *FileDialog*. *FileDialog* tersebut akan digunakan untuk mengambil file gambar yang ingin dilakukan operasi penyediaan citra. Kemudian alamat dari file yang dituju akan ditulis pada *teksfield*. Selanjutnya alamat dari file gambar ini digunakan untuk menampilkan citra di kanvas pada

panel yang nantinya dibentuk dari kelas *PanelGambarAsli*.

Kelas CariRGB : Kelas *CariRGB* merupakan kelas yang digunakan untuk mendapatkan komponen warna dari sebuah citra. Proses pengambilan komponen warna dari citra tersebut dilakukan pada *method* *rgb*. Langkah yang pertama adalah membuka file sesuai alamat yang terdapat pada *teksfield* pada *panel* yang dibentuk oleh kelas *PanelCariGambar*. Kemudian dibuat suatu *BufferedImage* yang bernama gambar.

```

try
{
    String namafile = parent.teksfield.getText();
    FileInputStream fis = new
        FileInputStream(namafile);
    gambar = ImageIO.read(fis);
    fis.close();
}

```

Apabila ternyata *teksfield* pada *panel* yang dibentuk dari kelas *PanelCariGambar* masih kosong, program akan menampilkan *MessageDialog* yang memberitahukan bahwa nama file belum diisi.

```

catch (Exception e)
{
    JFrame frame = new JFrame();
    final String pesenError =
        "Nama file belum diisi\n" +
        "Silahkan masukkan nama file ...!!!";
    JOptionPane.showMessageDialog(
        (frame, pesenError, "Error",
        JOptionPane.ERROR_MESSAGE);
    gambar = null;
}

```

Setelah itu objek gambar diambil komponen warnanya. Komponen warna yang diambil adalah sebuah nilai integer yang merepresentasikan nilai warna merah, hijau dan biru untuk setiap piksel pada gambar. Nilai yang didapat, dimasukkan ke dalam variabel array *pksl* yang bertipe integer.

```

for(int y=0; y<gambar.getHeight(); y++)
    for(int x=0; x<gambar.getWidth(); x++)
    {
        int piksel = gambar.getRGB(x,y);
        Color warna = new Color(piksel);
        pksl[n] = warna.getRed(); n++;
        pksl[n] = warna.getGreen(); n++;
        pksl[n] = warna.getBlue(); n++;
    }

```

Kelas Enkripsi : Kelas ini merupakan kelas inti dari proses penyandian terhadap citra. Di kelas, ini untuk melakukan penyandian citra, matriks kunci yang dimasukkan pengguna dikalikan dengan matriks dari komponen warna citra. Sedangkan untuk mengembalikan citra kekeadaan semula, dilakukan dengan 2 tahap. Yang pertama, lakukan operasi perkalian inverse determinan matriks kunci dengan setiap elemen dari adjoint matriks kunci. Yang kedua, hasil dari perkalian tersebut dikalikan dengan matriks gambar yang ingin dikembalikan kekeadaan semula. Proses perkalian matriks tersebut dilakukan pada *method* kaliMatriks.

```
for (int i=0; i<ordo; i++)
    for (int j=0; j<kolomB; j++)
    {
        for (k=0; k<ordo; k++)
            hasil[i][j] = hasil[i][j]+matA[i][k]*matB[k][j];
            hasil[i][j] = hasil[i][j] % 256;
            while (hasil[i][j] < 0){ hasil[i][j] += 256;}
    }
```

Nilai dari setiap elemen matriks hasil harus dalam *range* 0 sampai 255. oleh karena itu setiap elemen dari matriks hasil perkalian dilakukan operasi modulo 256. setelah itu nilai dari setiap elemen matriks disimpan dalam variabel array kali.

Pada kelas ini terdapat *method* setGambar. *Method* ini berfungsi untuk membentuk suatu gambar dari matriks hasil operasi perkalian pada *method* kaliMatriks.

```
for(int y=0; y<ye; y++)
    for(int x=0; x<ex; x++)
    {
        int r = kali[n]; n++;
        int g = kali[n]; n++;
        int b = kali[n]; n++;
        Color warnanya = new Color(r, g, b);
        int rgbnya = warnanya.getRGB();
        pic.setRGB(x, y, rgbnya);
    }
```

Kelas PanelKanvas : Kelas ini merupakan cetak biru untuk membuat sebuah objek *panel*, dimana pada *panel* tersebut akan diletakkan objek panel asli dan panel hasil yang masing-masing dibentuk dari kelas *PanelGambarAsli* dan *PanelGambarHasil*. Untuk menentukan tata letak *panel-panel* tersebut,

digunakan *GridLayout* dengan jumlah baris 2 dan jumlah kolom 1.

Kelas PanelGambarAsli : Kelas ini merupakan cetak biru untuk membuat sebuah objek *panel*, dimana pada *panel* tersebut akan diletakkan objek asli yang dibentuk dari kelas *GambarAsli*. Untuk pengaturan tata letak, kelas ini menggunakan *GridLayout*. Objek *panel* yang dibentuk dari kelas ini akan diberi judul “Gambar Asli”.

Kelas GambarAsli : Kelas ini akan membentuk sebuah kanvas yang pada kanvas tersebut nantinya akan ditampilkan citra yang akan dilakukan operasi penyandian. *Method* yang bertugas untuk melakukan penggambaran terhadap kanvas adalah *method* paint.

```
public void paint(Graphics g)
{
    String file = parent.parent.parent.panelproses.
        panelmatriks.panelcarigambar.f;
    if (file != null)
    {
        Toolkit tool = getToolkit();
        Image gbr = tool.getImage(file);
        g.drawImage(gbr, 120, 40, 100, 100, this);
    }
}
```

Kelas PanelGambarHasil : Kelas ini merupakan cetak biru untuk membuat sebuah objek *panel*, dimana pada *panel* tersebut akan diletakkan objek hasil yang dibentuk dari kelas *GambarHasil*. Untuk pengaturan tata letak, kelas ini menggunakan *GridLayout*. Objek *panel* yang dibentuk dari kelas ini akan diberi judul “Gambar Hasil”.

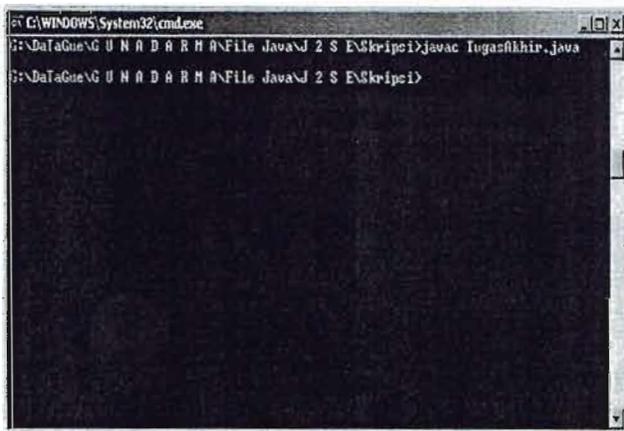
Kelas GambarHasil : kelas ini akan membentuk sebuah kanvas yang pada kanvas tersebut nantinya akan ditampilkan citra hasil operasi penyandian. *Method* yang bertugas untuk melakukan penggambaran terhadap kanvas adalah *method* paint.

```
public void paint(Graphics g)
{
    String file = parent.parent.
        parent.panelproses.enkrip.file;
    if (file != null)
    {
        Toolkit tool = getToolkit();
        Image gbr = tool.getImage(file);
        g.drawImage(gbr, 120, 40, 100,
            100, this);
    }
}
```

Kelas PanelKeluar : Kelas ini merupakan cetak biru untuk membentuk sebuah *panel*, dimana pada *panel* tersebut akan diletakkan sebuah objek *Button* tombolkeluar. Pada kelas ini terdapat sebuah method yang bertugas melakukan penanganan event. Method tersebut akan bekerja saat tombol tombolkeluar ditekan. Saat tombol ditekan, aplikasi akan berhenti dan kemudian keluar dari program.

```
public boolean action(Event evt, Object arg)
{
    if (evt.target instanceof Button)
        if (((String)arg).equals("Keluar"))
            System.exit(0);
    return true;
}
```

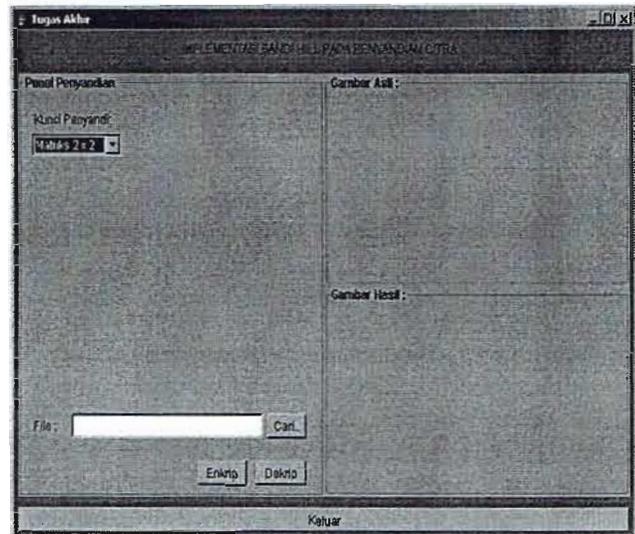
Pengujian Dan Pengeksekusian. Sebelum di eksekusi, program dikompilasi terlebih dahulu dengan menulis “javac TugasAkhir.java” pada DOS *prompt* dimana file java berada dengan mengatur *path* terlebih dahulu. Pengkompilasian ini bertujuan untuk menterjemahkan bahasa program menjadi bahasa yang dapat dimengerti oleh komputer. Dengan pengkompilasian ini akan diketahui kesalahan apa saja yang terdapat pada program yang dibuat. Apabila tidak terdapat kesalahan, akan muncul tampilan seperti gambar 2.



Gambar 2. Pengkompilasian program

Selanjutnya adalah mengeksekusi program. Caranya ialah dengan mengetik “java TugasAkhir”. Perlu diperhatikan bahwa pada saat pengeksekusian program, ekstensi (.java) tidak dipergunakan. Output dari program dapat dilihat pada gambar 3. Untuk menggunakan aplikasi ini, langkah yang pertama dapat dilakukan adalah memilih ordo dari matriks kunci penyandi. Pengguna dapat memilih

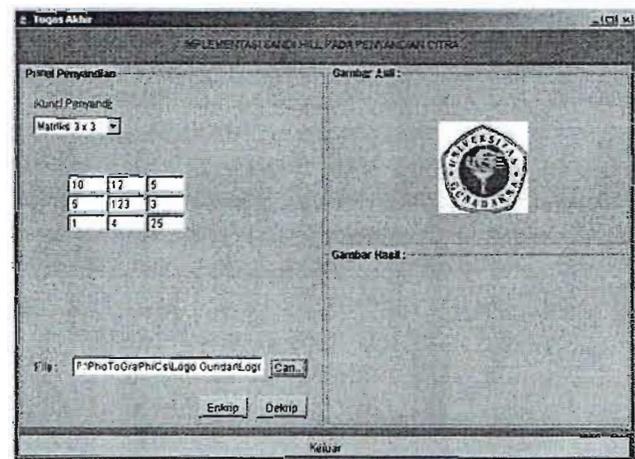
“Matriks 2 x 2” atau “Matriks 3 x 3”. Sebenarnya matriks tidak terbatas untuk dua ordo matriks tersebut, namun pada pembuatan Tugas Akhir ini,



Gambar 3. Tampilan awal aplikasi

penulis hanya membatasi penggunaan matriks kunci penyandi menjadi dua pilihan. Selanjutnya pengguna dapat memasukkan elemen-elemen dari matriks kunci. Matriks kunci yang diperkenankan adalah matriks kunci yang memiliki determinan dan determinan matriks tersebut memiliki *inverse* dalam modulo 256.

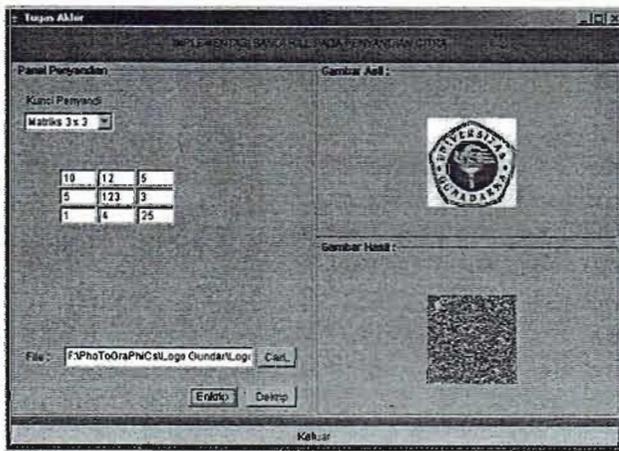
Setelah itu, pengguna dapat memasukkan nama file gambar yang ingin disandikan. Gambar 4 Merupakan contoh penggunaan aplikasi sampai dengan memasukkan nama file gambar.



Gambar 4. Tampilan penginputan data

Untuk melakukan penyandian citra, pengguna dapat menekan tombol enkrip. Setelah penekanan

tombol enkrip, program akan menampilkan kotak dialog untuk penyimpanan file gambar hasil enkripsi. Pada aplikasi ini, format file yang diperkenankan untuk menyimpan citra yang dihasilkan pada operasi penyandian citra adalah format png. Apabila proses penyandian selesai, citra hasil penyandian akan ditampilkan pada panel gambar hasil. Gambar 5 merupakan tampilan jika proses penyandian citra telah selesai.

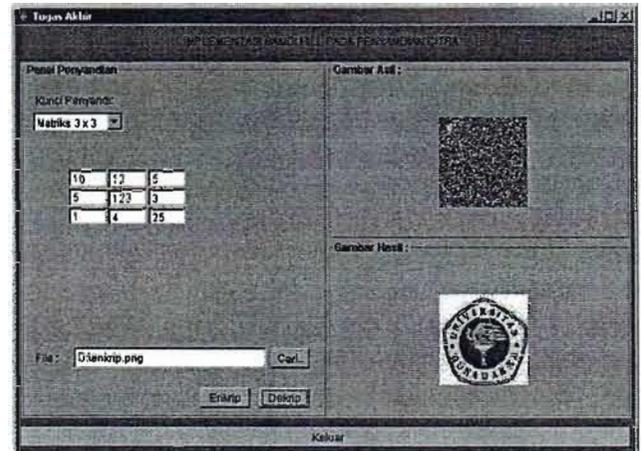


Gambar 5. Tampilan hasil enkripsi

Untuk mengembalikan citra hasil penyandian, langkah yang harus dilakukan hampir sama dengan penyandian citra. Perbedaannya hanya terletak pada penekanan tombol. Jika pada penyandian citra yang ditekan adalah tombol enkrip, pada pengembalian citra menjadi citra aslinya, yang ditekan adalah tombol dekrip. Matriks kunci yang digunakan untuk melakukan proses dekripsi adalah matriks yang sama yang digunakan untuk menyandikan citra tersebut. Gambar 6 memperlihatkan hasil pengembalian citra menjadi gambar aslinya. Format file hasil, mungkin tidak sama dengan format file pada saat citra belum dilakukan operasi penyandian. Citra hasil, memiliki format file png.

PENUTUP

Simpulan. Setelah menggunakan Aplikasi Penyandian Citra ini, penulis mendapatkan beberapa catatan. Catatan tersebut diantaranya; sandi Hill merupakan metode penyandian sederhana yang cocok diterapkan pada citra dengan variasi warna tinggi (seperti foto), dan tidak cocok diterapkan pada citra yang warnanya tidak terlalu bervariasi. Keuntungan utama pemakaian sandi Hill untuk



Gambar 6. Tampilan hasil dekripsi

penyandian citra adalah metodenya yang sederhana sehingga prosesnya cepat. Kerugian yang terjadi adalah tidak tunggalnya matriks kunci A (dan A^{-1}) yang menyebabkan tingkat keamanannya berkurang. Hanya matriks bujur sangkar yang nilai determinannya relatif prima dengan 256 saja yang dapat dipakai untuk proses penyandian. Dekripsi tidak bisa dilakukan jika nilai determinannya tidak relatif prima dengan 256.

Saran. Aplikasi ini belum sepenuhnya dapat memberikan solusi untuk sistem keamanan sebuah data citra, sehingga perlu dikembangkan lebih lanjut. Selain itu pada aplikasi ini belum terdapat fasilitas bantuan untuk pengguna, sehingga penulis menyarankan untuk menambahkan fasilitas tersebut apabila ada pihak yang mungkin tertarik untuk mengembangkan aplikasi ini. Serta format file hasil enkripsi hendaknya dapat lebih bervariasi, tidak terbatas pada format file png.

Sebagai pertimbangan bagi pihak yang ingin mengembangkan aplikasi ini, penyandian dapat dilakukan dengan mengelompokkan nilai rgb menjadi beberapa matriks terpisah yang masing-masing ditangani oleh matriks kunci yang berbeda.

DAFTAR PUSTAKA

Anonim. *Algoritma & Pemrograman*. Gunadarma. Jakarta. 1997.
 Deitel, Harvey.M., and Paul. J. *Java™ How To Program*, Prentice Hall. New Jersey. 2003.
 Kadir, Abdul. *Dasar Pemrograman Java 2*. Andi. Yogyakarta. 2004.
 Leonardo, Ian. *Belajar Sendiri Pemrograman Java 2D*. PT Elex Media Komputindo. Jakarta. 2003.

Naughton, Patrick. **The Java Handbook**. McGraw-Hill, Inc. New York. 1996.

Sanjaya, Ridwan. **Membuat Aplikasi Windows Multi Platform dengan Java GUI**, PT Elex Media Komputindo, Jakarta. 2003.

Wicaksono, Ady. **Dasar – Dasar Pemrograman Java 2**. Elex Media Komputindo. Jakarta. 2002.