KATA PENGANTAR

Puji syukur kehadirat Allah SWT, Sang Maha Pencipta yang telah memberikan limpahan rahmat dan hidayah sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "**Aplikasi Penyisipan Citra Gambar Pada Citra *Tagged Image File Format* Dengan Algoritma Enkripsi *Advanced Encryption Standard (AES)***". Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Teknik Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang. Tidak banyak yang bisa penulis sampaikan kecuali ungkapan terima kasih kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan tugas akhir ini dapat terselesaikan. Pada kesempatan kali ini, penulis mengucapkan banyak terima kasih kepada:

1. Ayah dan Ibu yang telah memberikan dukungan serta do'a dan semangat untuk terus maju hingga terselesaikan skripsi ini.
2. Dr. Ir. Sholeh Hadi Pramono, MS selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. M. Azis Muslim, S.T.,M.T.,Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Waru Djuriatno, ST., MT. selaku KKDK Teknik Informatika dan Komputer serta dosen pembimbing I yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini
5. Muhammad Aswin, Ir., MT. selaku dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini.
6. Nuriyatin dan keluarga, terima kasih atas segala perhatian, doa, kasih sayang, serta dukungan semangatnya
7. Bapak dan Ibu Dosen serta karyawan jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
8. Rekan-rekan TIK-FTUB. Pakdhe, Mas Mono, Mas Hiksa, Mas Galih, Mas Gamma, Mas Alan, Mas Firman, Mas Rizal, Riza, Wahyu, Bara, Aji,

Ajoy, Sebenk, Yusuf Faris, Ubet, Alfian, Dena, Henry yang telah bersedia berbagi pengalaman, masukan, dan bantuan yang berkaitan dengan skripsi ini.

9. Rekan-rekan 2005, Fahmanda'05, Agung Pamungkas'05, Christian Jonathan'05, Alfian'05, Bagus Arifin'05, Sauqi'05, Andi Bahtiar'04, dan rekan-rekan jurusan TEUB yang telah banyak membantu serta dukungan hingga tugas akhir ini dapat diselesaikan.
10. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Penulis menyadari sepenuhnya bahwa Tugas Akhir ini masih banyak kekurangan. Segala kritik dan saran yang membangun akan penulis terima demi kesempurnaan skripsi ini. Harapan penulis semoga skripsi ini bermanfaat bagi pembaca dan khususnya mahasiswa jurusan teknik elektro dimasa yang akan datang.

Malang, Juli 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

M. Bernie Fityanto (0510630063), Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya, Mei 2012. Aplikasi Penyisipan Citra Gambar Terenkripsi Pada Citra *Tagged Image File Format*. Dosen Pembimbing: Waru Djuriatno, ST., MT. dan M. Aswin, Ir., MT

Keamanan dalam pengiriman informasi merupakan salah satu faktor penting yang harus dijaga, apabila isi dari informasi tersebut tidak boleh diketahui oleh pihak luar. Salah satu teknik yang dapat dipakai untuk menangani hal tersebut adalah cryptografi. Cryptografi merupakan ilmu dan seni yang mempelajari cara penyembunyian informasi rahasia ke dalam suatu media kemudian dienkripsi sedemikian rupa sehingga manusia tidak dapat menyadari keberadaan pesan tersebut. Pada Tugas Akhir ini, dilakukan studi mengenai bagaimana cryptografi pada media citra. Citra yang digunakan memiliki format TIFF, maka dilakukan juga studi terhadap bagaimana representasi citra TIFF terlebih dahulu. Kemudian dilakukan juga implementasi terhadap hasil studi beserta analisis yang telah dikerjakan. Metode yang digunakan adalah LSB Modification, yaitu melakukan perubahan terhadap nilai bit yang kurang penting (pada bagian nilai terkecil) dari citra.

Perangkat lunak ini dibangun pada perangkat desktop dengan menggunakan bahasa pemrograman C++. Sistem ini mempunyai 2 macam penyisipan untuk citra terenkripsi dan penyisipan digital signature. Untuk data terkenripsi disisipkan pada alpa channel sedangkan digital signature pada 16 byte pertama red channel pada citra yang ditumpangi (citra TIFF)

Berdasarkan hasil pengujian dan analisis sistem, perangkat lunak ini berhasil menjalankan semua fungsinya dengan benar. Perangkat lunak dapat melakukan penyisipan gambar ke dalam citra TIFF, dan dapat mengekstraksinya dengan nilai yang valid dengan nilai PSNR rata-rata sebesar 28,3 dB.

Kata kunci: cryptografi, LSB, citra TIFF, alpha channel.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi digital serta internet yang cukup pesat telah memberi kemudahan dalam mengakses dan mendistribusikan berbagai informasi dalam format digital, baik berupa teks, citra, audio, maupun video. Seringkali seseorang yang hendak mengirim pesan kepada orang lain, tidak ingin isi pesan tersebut diketahui oleh orang lain. Biasanya isi pesan tersebut bersifat sangat rahasia atau pribadi, yang hanya boleh diketahui antara pihak pengirim dan pihak penerima pesan, atau kalangan terbatas saja. Oleh karena itu, biasanya pengirim tersebut mengirim pesan secara sembunyi-sembunyi agar tidak ada pihak lain yang mengetahui.

Untuk mengurangi atau mencegah terjadinya hal diatas adalah mengembangkan suatu aplikasi yang mampu menyamarkan pesan tersebut pada suatu media yang dapat diakses oleh setiap orang. Teknik ini disebut Kriptografi, yaitu teknik penyembunyian data pada suatu media kemudian menyandikannya sehingga tidak dikenali jika tidak mendeskripsikan dengan algoritma penyandiannya. Setiap orang dapat menampilkan atau membuka media tersebut, namun tidak menyadari bahwa media tersebut telah dibubuhkan pesan rahasia oleh pengirim.

Kriptografi memungkinkan penyembunyian data pada berbagai jenis media digital seperti berkas citra, suara, video, dan teks. Salah satu aplikasi yang dikembangkan untuk citra digital adalah dengan menggunakan gambar untuk disisipkan pada suatu gambar dengan berbagai tujuan, salah satunya adalah mengautentikasi berkas yang disisipinya. Dengan demikian format yang disisipi tersebut merupakan data asli yang berasal dari pengirim.

Dalam tugas akhir ini, akan dirancang suatu aplikasi untuk menyisipkan suatu gambar dengan menggunakan metode *Alpha Channel Swap*. Dengan

metode ini maka suatu gambar yang disisipi gambar sehingga dapat menyembunyikan suatu gambar yang dapat diekstraksi kembali menjadi gambar asli dan gambar yang telah disisipkan tanpa mengurangi kualitas gambar yang ditumpanginya.

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka dapat ditetapkan rumusan masalah sebagai berikut :

1. Bagaimana melakukan penyembunyian citra gambar ke dalam citra tiff dengan metode LSB.
2. Bagaimana membandingkan kualitas antara citra yang asli dengan citra yang sudah disisipi gambar.

1.3 Tujuan

Tujuan penulisan skripsi ini yaitu:

1. Merancang teknik penyembunyian citra gambar pada citra tiff agar dapat disisipkan pada LSB.
2. Merancang suatu aplikasi yang dapat menyisipkan sebuah gambar yang dienkripsi.

1.4 Batasan Masalah

Agar penulisan ini lebih terarah dan sesuai dengan tujuan yang diinginkan maka permasalahan perlu dibatasi sebagai berikut:

1. Format data yang digunakan untuk media yang ditumpangi adalah citra gambar yang mempunyai format TIFF (Tagged Image File Format).
2. Citra Gambar yang disisipkan berupa citra gambar hitam putih.
3. Ukuran citra gambar yang disisipkan sama dengan citra gambar yang ditumpangi.
4. Digital signature yang digunakan adalah MD5.
5. Panjang kunci AES yang digunakan adalah 128-bit.

6. Aplikasi ditulis dalam bahasa pemrograman C.
7. Aplikasi berjalan di atas operating system windows.

1.5 Manfaat

Manfaat yang akan diperoleh melalui pengerjaan skripsi ini adalah memberikan kemudahan pengguna ketika ingin menyisipkan gambar rahasia ke dalam sebuah gambar dengan metode alpha channel swap.

1.6 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Menjelaskan latar belakang, rumusan masalah, ruang lingkup, tujuan, metodologi dan sistematika penulisan skripsi

BAB II Dasar Teori

Menjelaskan kajian pustaka dan dasar teori yang digunakan

BAB III Metodologi

Menjelaskan metodologi yang digunakan dalam pengerjaan skripsi

BAB IV Perancangan

Menjelaskan langkah-langkah perancangan untuk merubah alpha channel suatu gambar dan menyisipkan gambar kedalamnya.

BAB V Implementasi

Menjelaskan langkah-langkah pembuatan aplikasi dengan bahasa C

BAB VI Pengujian

Menjelaskan langkah-langkah pengujian dari aplikasi yang telah dibuat dan membahas hasil pengujiannya.

BAB VIIKesimpulan dan Saran

Berisi Kesimpulan dan Saran

BAB II

TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari Bahasa Yunani: "*cryptós*" artinya "*secret*" (rahasia), sedangkan "*gráphein*" artinya "*writing*" (tulisan). Jadi, kriptografi berarti "*secret writing*" (tulisan rahasia). Ada beberapa definisi kriptografi yang telah dikemukakan di dalam berbagai literatur. Definisi yang dipakai di dalam buku-buku yang lama (sebelum tahun 1980-an) menyatakan bahwa kriptografi adalah ilmu dan seni untuk menjaga kerahasian pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya. Definisi ini mungkin cocok pada masa lalu di mana kriptografi digunakan untuk keamanan komunikasi penting seperti komunikasi di kalangan militer, diplomat, dan mata-mata.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. Autentikasi, adalah berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.

4. Non-repudiasi (penyangkalan) adalah usaha untuk mencegah terjadinya penyangkalan terhadap terciptanya suatu informasi oleh yang mengirimkan.

Menurut terminologinya, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat lain. Kata "seni" berasal dari fakta sejarah bahwa pada masa-masa awal sejarah kriptografi, setiap orang mungkin mempunyai cara yang unik untuk merahasiakan pesan. Cara-cara unik tersebut mungkin berbeda-beda pada setiap pelaku kriptografi sehingga setiap cara menulis pesan rahasia pesan mempunyai nilai estetika tersendiri.

Kriptografi berkembang menjadi sebuah seni merahasiakan pesan (kata "graphy" di dalam "cryptography" itu sendiri sudah menyiratkan sebuah seni). Pada perkembangan selanjutnya, kriptografi berkembang menjadi sebuah disiplin ilmu sendiri karena teknik-teknik kriptografi dapat diformulasikan secara matematik sehingga menjadi sebuah metode yang formal.

2.2 Enkripsi dan Dekripsi

Enkripsi adalah hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirim agar terjaga kerahasiaannya. Pesan asli disebut plaintext, yang diubah menjadi kode-kode yang tidak dimengerti. Enkripsi bias diartikan dengan cipher atau kode.

Dekripsi merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya(teks asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.

Keamanan dari kriptografi modern didapat dengan merahasiakan kunci yang dimiliki dari orang lain, tanpa harus merahasiakan algoritma itu sendiri. Kunci memiliki fungsi yang sama dengan password. Jika keseluruhan keamanan algoritma tergantung pada kunci yang dipakai, maka algoritma ini bisa dipublikasikan dan dianalisa orang lain.

2.3 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) dipublikasikan oleh *NIST (National Institute of Standard and Tecnology)* pada tahun 2001. AES merupakan simetris block cipher untuk menggantikan DES *(Data Encryption Standard).* Dasar dari pemilihan ini bukan karena AES sebagai algoritma yang paling aman, tapi karena memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai platform software dan hardware.

Input dan output dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data yang sudah terbentuk dalam satu kelompok 128 bit tersebut disebut juga sebagai blok data atau *plaintext* yang nantinya akan dienkripsi menjadi *ciphertext*. *Cipher key* dari AES terdiri dari *key* dengan panjang 128 bit, 192 bit, atau 256 bit.

Urutan bit diberi nomor urut dari 0 sampai dengan n-1 dimana n adalah nomor urutan. Urutan data 8 bit secara berurutan disebut sebagai byte dimana byte ini adalah unit dasar dari operasi yang akan dilakukan pada blok data. Dalam algoritma AES, data sepanjang 128 bit akan dibagi-bagi menjadi *array byte* dimana setiap *array byte* ini terdiri dari 8 bit data input yang saling berurutan. Array byte ini direpresentasikan dalam bentuk :

$$a_0 a_1 a_2 ... a_{15}$$

Dimana :

$$a_0 = \{ input_0, input_1, ..., input_7 \}$$

$$a_1 = \{ input_8, input_9, ..., input_{15} \}$$

$$a_{15} = \{ input_{120}, input_{121}, ..., input_{127} \}$$

$$a_n = \{ input_{8n}, input_{8n+1}, ..., input_{8n+7} \}$$

Tabel 2.2. Representasi data AES.

Urutan bit input	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...

Nomor Byte	0								1								...
Nomor Bit pada Byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

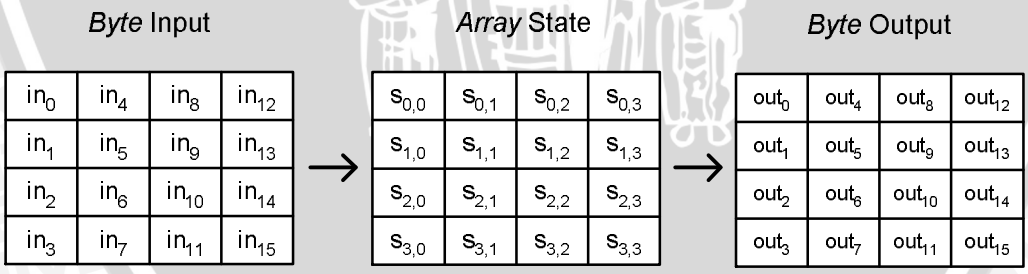
Sumber : Advanced Encryption Standard, Algoritma Rijndael.

Operasi algoritma AES dilakukan pada sebuah *State* dimana *State* sendiri adalah sebuah array byte dua dimensi. Setiap *State* pasti mempunyai jumlah baris yang tetap, yaitu 4 baris, sedangkan jumlah kolom tergantung dari besarnya blok data. Baris pada State mempunyai indeks nomor *row* (*r*) dimana $0 \leq r < 4$, sedangkan kolom mempunyai indeks nomor *column* (*c*) dimana $0 \leq c < Nb$. *Nb* sendiri adalah besarnya blok data dibagi 32.Pada saat permulaan, input bit pertama kali akan disusun menjadi suatu array byte dimana panjang dari array byte yang digunakan pada AES adalah sepanjang 8 bit data. Array byte inilah yang nantinya akan dimasukkan atau dikopi ke dalam State dengan urutan :

$$s[r,c] = in[r+4c] \text{ untuk } 0 \leq r < 4 \text{ dan } 0 \leq c < Nb$$

sedangkan dari State akan dikopi ke output dengan urutan :

$$out[r+4c] = s[r,c] \text{ untuk } 0 \leq r < 4 \text{ dan } 0 \leq c < Nb$$



Gambar 2.13. Byte Input, Array State, dan Byte Output.

2.3.1 Spesifikasi AES

Pada algoritma AES, jumlah blok input, blok output, dan State adalah 128 bit. Dengan besar data 128 bit, berarti *Nb* = 4 *word* yang mencerminkan jumlah 32 bit word (jumlah kolom) dalam State. Dengan blok input atau blok data sebesar 128 bit, *key* yang digunakan pada algoritma AES tidak harus mempunyai besar

yang sama dengan blok input. *Cipher key (K)* pada algoritma AES bisa menggunakan kunci dengan panjang 128 bit, 192 bit, atau 256 bit. Perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan diimplementasikan pada algoritma AES ini [WAW-04]. Di bawah ini adalah tabel yang memperlihatkan spesifikasi dari AES.

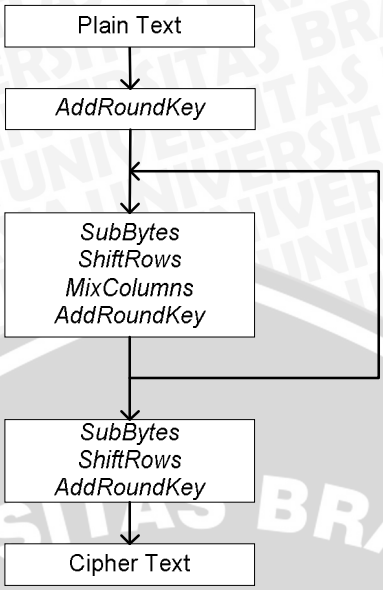
Tabel 2.3. Spesifikasi AES.

	AES-128	AES-192	AES-256
Key Length *(Nk)*	4 word	6 word	8 word
Block Size *(Nb)*	4 word	4 word	4 word
Number of Round *(Nr)*	10	12	14
Round key size	4 word	4 word	4 word
Expanded key size	44 word	52 word	60 word

Dari tabel di atas AES-128 bit menggunakan panjang kunci Nk = 4 word yang setiap wordnya terdiri dari 32 bit sehingga total kunci 128 bit, ukuran block plaintext 128 bit dan memiliki round sebanyak 10 round (putaran), sedangkan putaran untuk kunci juga terdiri dari K_i = 4 word dan total putaran kunci 128 bit dan kunci yang diperluas mempunyai ukuran 44 word atau 176 byte.

2.3.2 Enkripsi

Proses enkripsi pada algoritma AES terdiri dari 4 jenis transformasi bytes, yaitu SubBytes, ShiftRows, Mixcolumns, dan AddRoundKey. Pada awal proses enkripsi, input yang telah dikopikan ke dalam akan mengalami transformasi byte AddRoundKey. Setelah itu, *State* akan mengalami transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey secara berulang-ulang sebanyak *Nr*. Proses ini dalam algoritma AES disebut sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *State* tidak mengalami transformasi MixColumns [WAW-04].



Gambar 2.3.2 Diagram Alir Proses Enkripsi.

```
Algoritma Enkripsi AES
{algoritma kriptografi AES jenis kunci rahasia (simetris)}

DEKLARASI :
    Nb, Nr    : word;
    in        : array [0…15] of byte;
    out       : array [0…15] of byte;
    w         : array [0…43] of byte;
    State     : array [0…3] [0…3] of byte;
    round     : integer;

DESKRIPSI :
  1   Nb ← 4              {ukuran blok}
  2   Nr ← 10                    {jumlah round untuk key 128 bit}
  3   State ← in   (Plaintext)
  4   AddRoundKey(State, w[0, Nb-1])
  5   for round ← 1 step 1 to Nr-1
  6       SubBytes(State)
  7       ShiftRows(State)
  8       MixColumns(State)
  9       AddRoundKey(State, w[round*Nb, (round+1)*Nb-1])
 10   end for
 11   SubBytes(State)
 12   ShiftRows(State)
 13   AddRoundKey(State, w[Nr*Nb, (Nr+1)*Nb-1])
 14   out ← State (Ciphertext)
```

Pseudocode dari algoritma Enkripsi AES.

Penjelasan pseudocode dari algoritma gambar 2.15, yaitu :

1. Baris 3 : Memasukkan plaintext.
2. Baris 4 : Inisialisasi round.

3. Baris 5-10 : Proses operasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey

   sebanyak 9 round urut dari round 1 sampai round 9.

4. Baris 11-13 : Proses operasi SubBytes, ShiftRows, dan AddRoundKey sebanyak 1

   round pada round yang terakhir (round 10).

5. Baris 14 : Keluaran berupa Ciphertext.

2.3.2.1 SubBytes

SubBytes merupakan transformasi byte dimana setiap elemen pada State akan dipetakan dengan menggunakan sebuah tabel substitusi (*S-Box*). Hasil yang didapat dari pemetaan dengan menggunakan tabel *S-Box* ini sebenarnya adalah hasil dari dua proses transformasi bytes, yaitu :

1. *Invers* perkalian dalam GF(2^8) adalah fungsi yang memetakan 8 bit ke 8 bit yang merupakan *invers* dari elemen *finite field* tersebut. Suatu byte a merupakan *invers* perkalian dari byte b bila a•b = 1, kecuali {00} dipetakan ke dirinya sendiri. Setiap elemen pada *State* akan dipetakan pada tabel *invers*. Sebagai contoh, elemen “01010011” atau {53} akan dipetakan ke {CA} atau “11001010”.
2. Transformasi *affine* pada *State* yang telah dipetakan. Transformasi *affine* ini apabila dipetakan dalam bentuk matriks adalah sebagai berikut :

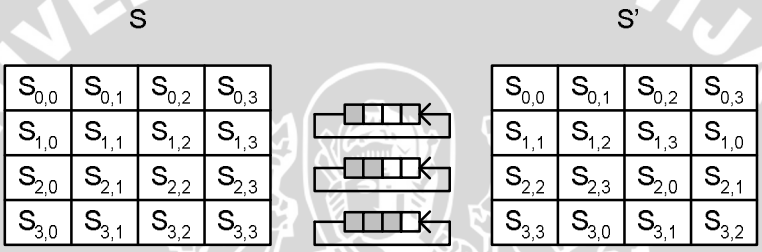
$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Matriks Affine.

$b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$ adalah urutan bit dalam elemen State atau array byte dimana b_7 adalah *most significant bit* atau bit dengan posisi paling kiri.

2.3.2.2 ShiftRows

Transformasi *Shiftrows* pada dasarnya adalah proses pergeseran bit dimana bit paling kiri akan dipindahkan menjadi bit paling kanan (rotasi bit). Transformasi ini diterapkan pada baris 2, baris 3, dan baris 4. Baris 2 akan mengalami pergeseran bit sebanyak satu kali, sedangkan baris 3 dan baris 4 masing-masing mengalami pergeseran bit sebanyak dua kali dan tiga kali.



Transformasi ShiftRows.

2.3.2.3 MixColumns

Mixolumns mengoperasikan setiap elemen yang berada dalam satu kolom pada *State*. Elemen pada kolom dikalikan dengan suatu polinomial tetap a(x) = $\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. Secara lebih jelas, transformasi mixcolumns dapat dilihat pada perkalian matriks berikut ini :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Matriks Transformasi MixColumns.

Melakukan proses penambahan pada operasi ini berarti melakukan operasi *bitwise* XOR. Maka hasil dari perkalian matriks diatas dapat dianggap seperti perkalian yang ada di bawah ini :

$$s'_{0,c} = (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Hasil perkalian dari operasi matriks MixColumns.

2.3.2.4 AddRoundKey

Pada proses *AddRoundKey*, sebuah *round key* ditambahkan pada *State* dengan operasi bitwise XOR. Setiap *round key* terdiri dari *Nb word* dimana tiap *word* tersebut akan dijumlahkan dengan *word* atau kolom yang bersesuaian dari *State* sehingga :

$$\left[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}\right] = \left[s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}\right] \oplus \left[w_{round*Nb+c}\right] \text{untuk } 0 \le c \le Nb$$

$[w_i]$ adalah *word* dari key yang bersesuaian dimana *i = round*Nb+c*. Transformasi AddRoundKey diimplementasikan pertama kali pada *round* = 0, dimana *key* yang digunakan adalah *initial key* (*key* yang dimasukkan oleh kriptografer dan belum mengalami proses *key expansion*).

2.3.2.5 Ekspansi Kunci

Algoritma AES mengambil kunci cipher, K, dan melakukan rutin ekspansi kunci (*key expansion*) untuk membentuk *key schedule*. Ekspansi kunci menghasilkan total Nb(Nr+1) *word*. Algoritma ini membutuhkan set awal *key* yang terdiri dari *Nb word*, dan setiap *round Nr* membutuhkan data kunci sebanyak *Nb word*. Hasil *key schedule* terdiri dari array 4 byte word linear yang dinotasikan dengan $[w_i]$.

SubWord adalah fungsi yang mengambil 4 byte *word* input dan mengaplikasikan S-Box ke tiap-tiap data 4 byte untuk menghasilkan *word* output. Fungsi *RotWord* mengambil *word* $[a_0, a_1, a_2, a_3]$ sebagai input, melakukan

permutasi siklik, dan mengembalikan *word* [a_1, a_2, a_3, a_0]. Rcon[i] terdiri dari nilai-nilai yang diberikan oleh [x^{i-1}, {00}, {00}, {00}], dengan x^{i-1} sebagai pangkat dari x (x dinotasikan sebagai {02} dalam *field GF*(2^8).

```
Algoritma Key Expansion AES
{algoritma kriptografi AES jenis kunci rahasia (simetris)}
DEKLARASI :
    Nk, Nb, Nr, temp : word;
    key      : array [0…15] of byte;
    w        : array [0…43] of byte;
    i        : integer;

DESKRIPSI :
  1   Nk ← 4              {panjang kunci untuk key 128 bit}
  2   Nb ← 4              {ukuran blok}
  3   Nr ← 10                    {jumlah round untuk key 128 bit}
  4   i ← 0
  5   for i ← 0 step 1 to Nk
  6      w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
  7   end for
  8   for i ← Nk step 1 to Nb * (Nr+1)]
  9      temp ← w[i-1]
 10      if i mod = 0 then
 11         temp ← SubWord(RotWord(temp)) xor Rcon[i/Nk]
 12      else if Nk > 6 and i mod Nk = 4 then
 13         temp ← SubWord(temp)
 14      end if
 15      w[i] ← w[i-Nk] xor temp
 16   end for
```

Pseudocode dari algoritma Key Expansion AES.

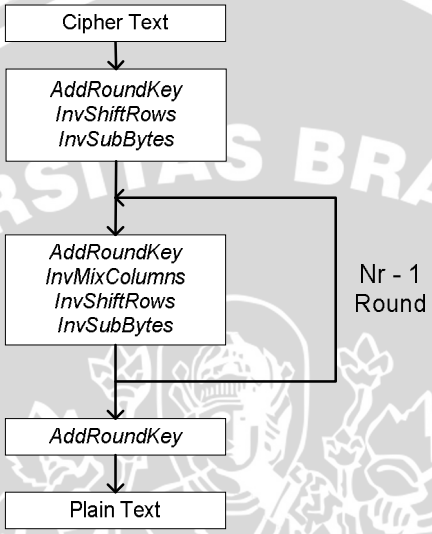
Penjelasan pseudocode dari algoritma gambar 2.20, yaitu :

1. Baris 5-7 : Proses perluasan kunci dimana *Nk* word yang pertama diisi CipherKey.

2. Baris 8-16 : Proses key schedule yang setiap penambahan wordnya *(w[i])* adalah operasi XOR dari word sebelumnya *(w[i-1])* dengan Nk word awal *(w[i-Nk])*. Jika word dalam posisi kelipatan dari *Nk* maka transformasi diberlakukan *(w[i-1])* lebih dulu pada XOR, diikuti oleh XOR dengan round konstan atau *Rcon[i]*. Transformasi ini terdiri dari permutasi siklik dalam word *(RotWord())* yang diikuti penggunaan S-Box ke semua 4 byte word *(SubWord())*. Fungsi *(SubWord())* dan *(RotWord())* mngembalikan hasil dari transformasi fungsi input, dimana transformasi dalam Cipher dan Inverse Cipher (seperti ShiftRows, SubBytes,dsb) mengubah array State yang dialamatkan oleh “State” pointer.

2.3.3 Dekripsi

Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada invers *cipher* adalah InvShiftRows, InvSubBytes, InvMixColumns, dan AddRoundKey. Algoritma dekripsi dapat dilihat pada skema berikut ini [WAW-04] :



Gambar 2.3.3. Diagram Alir Proses Dekripsi.

```
Algoritma Dekripsi AES
{algoritma kriptografi AES jenis kunci rahasia (simetris)}
DEKLARASI :
    Nb, Nr    : word;
    in        : array [0…15] of byte;
    out       : array [0…15] of byte;
    w         : array [0…43] of byte;
    State     : array [0…3] [0…3] of byte;
    round     : integer;
DESKRIPSI :
  1   Nb ← 4              {ukuran blok}
  2   Nr ← 10                    {jumlah round untuk key 128 bit}
  3   State ← in   (Ciphertext)
  4   AddRoundKey(State, w[Nr*Nb, (Nr+1)*Nb-1])
  5   for round ← Nr-1 step -1 downto 1
  6       InvShiftRows(State)
  7       InvSubBytes(State)
  8       AddRoundKey(State, w[round*Nb, (round+1)*Nb-1])
  9       InvMixColumns(State)
 10   end for
 11   InvShiftRows(State)
 12   InvSubBytes(State)
 13   AddRoundKey(State, w[0, Nb-1])
 14   out ← State (Plaintext)
```

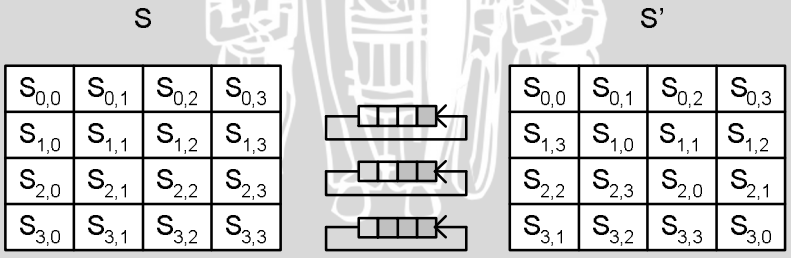
Pseudocode dari algoritma Dekripsi AES

Penjelasan pseudocode dari algoritma gambar 2.22, yaitu :

1. Baris 3 : Memasukkan ciphertext.
2. Baris 4 : Inisialisasi round.
3. Baris 5-10 : Proses operasi InvShiftRows, InvSubBytes, AddRoundKey, dan InvMixColumns, sebanyak 9 round mulai dari round 9 sampai round 1.
4. Baris 11-13 : Proses operasi InvShiftRows, InvSubBytes dan AddRoundKey sebanyak 1 round pada round yang terakhir (round 10).
5. Baris 14 : Keluaran berupa Plaintext.

2.3.3.1 InvShiftRows

InvShiftRows adalah transformasi byte yang berkebalikan dengan transformasi ShiftRows. Pada transformasi InvShiftRows, dilakukan pergeseran bit ke kanan sedangkan pada ShiftRows dilakukan pergeseran bit ke kiri. Pada baris kedua, pergeseran bit dilakukan sebanyak 3 kali, sedangkan pada baris ketiga dan baris keempat, dilakukan pergeseran bit sebanyak dua kali dan satu kali.



Transformasi *InvShiftRows.*

2.3.3.2 InvSubBytes

InvSubBytes juga merupakan transformasi bytes yang berkebalikan dengan transformasi SubBytes. Pada InvSubBytes, tiap elemen pada *State* dipetakan dengan menggunakan tabel *inverse S-Box*. Tabel ini berbeda dengan tabel *S-Box* dimana hasil yang didapat dari tabel ini adalah hasil dari dua proses yang berbeda

urutannya, yaitu transformasi *affine* terlebih dahulu, baru kemudian perkalian *invers* dalam GF(2^8).

$$\begin{bmatrix} b_7' \\ b_6' \\ b_5' \\ b_4' \\ b_3' \\ b_2' \\ b_1' \\ b_0' \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Matriks Invers Affine.

Perkalian *invers* yang dilakukan pada transformasi InvSubBytes ini sama dengan perkalian *invers* yang dilakukan pada transformasi SubBytes.

2.3.3.3 InvMixColumns

Pada *InvMixColumns*, kolom-kolom pada tiap *State* (*word*) akan dipandang sebagai polinom atas GF(2^8) dan mengalikan modulo $x^4 + 1$ dengan polinom tetap $a^{-1}(x)$ yang diperoleh dari :

$$a^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}.$$

Atau dalam matriks :

$$s'(x) = a(x) \otimes s(x)$$

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Matriks InvMixColumns.

Hasil dari perkalian diatas adalah :

$$s'_{0,c} = (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,c} = (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c})$$
$$s'_{2,c} = (\{0D\} \bullet s_{o,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0E\} \bullet s_{2,c}) \oplus (\{0B\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{0B\} \bullet s_{o,c}) \oplus (\{0D\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0E\} \bullet s_{3,c})$$

2.4 Fungsi Hash Satu Arah

Fungsi hash satu arah memiliki banyak nama: fungsi pembanding, fungsi penyusutan, intisari pesan, sidik jari, message integrity check (MIC) atau pemeriksa keutuhanpesan dan manipulation detection code (MDC) atau pendektesi penyelewengan kode.

Fungsi hash satu arah dibuat berdasarkan ide tentang fungsi pemampatan. Fungsi hash adalah sebuah fungsi atau persamaan matematika yang mengambil input dengan panjang variabel (preimage) dan merubahnya menjadi panjang yang tetap (biasanya lebih pendek), keluarannya biasa disebut nilai hash.

Fungsi hash satu arah adalah sebuah fungsi hash yang berjalan hanya satu arah. Adalah mudah untuk menghitung nilai hash dari pre-image, tetapi sangat sulit untuk membangkitkan pre-image dari nilai hash-nya.

Metode fungsi hash satu arah adalah berfungsi melindungi data dari modifikasi. Apabila ingin melindungi data dari modifikasi yang tidak terdeteksi, dapat di hitung hasil fungsi hash dari data tersebut, selanjutnya dapat menghitung hasil fungsi hash lagi dan membandingkannya dengan hasil yang pertama apabila berbeda maka terjadi perubahan selama pengiriman.

Sebagai contohnya adalah bila si pengirim (A) akan mengirim pesan kepada temannya (B). Sebelum mengirim, A melakukan hash dari pesannya untuk mendapatkan nilai hash kemudian dia mengirim pesan itu beserta nilai hashnya, Lalu B melakukan hash untuk mencari nilai hash dari pesan itu bila terjadi perbedaan maka sewaktu pengiriman telah terjadi perubahan dari pesan tersebut. Masukan dari fungsi hash satu arah adalah blok pesan dan keluaran dari blok text atau nilai hash sebelumnya.

2.5 MD5

Fungsi hash adalah fungsi yang memproduksi output dengan panjang tetap dari input yang berukuran variabel. Output dari fungsi hash disebut dengan

message digest. Fungsi hash memiliki karakteristik fungsi satu arah karena file asli tidak dapat dibuat dari *message digest*. MD5 merupakan fungsi hash yang sering digunakan untuk mengamankan suatu jaringan computer dan internet yang sengaja dirancang dengan tujuan sebagai berikut :

- Keamanan : hal ini tidak bisa dielakkan karena tidak ada suatu sistem algoritma tidak bisa dipecahkan, serangan yang sering digunakan untuk menjebol algoritma hash dengan menggunakan serangan brute force.
- Kecepatan : software yang digunakan mempunyai kecepatan yang tinggi, karena berdasarkan pada sekumpulan manipulasi operan 32 bit.
- Simple : tanpa menggunakan sruktur data yang komplek.

MD5 mengolah input yang berbentuk blok 512 bit yang dibagi dalam 16 bentuk subblok yang masing-masing berukuran 32 bit, sedangkan output terdiri dariu 4 blok yang berukuran 32 bit,dari 4 blok output digabungkan menjadi 128 bit. Contoh dari progam menggunakan MD5 adalah PGP, MD5 juga digunakan untuk pengkodean password pada sistem operasi.

Proses ini mempunyai masukan :

- Data gambar yang akan disisipkan

Proses ini mempunyai keluaran :

- Data -*Key* berupa *message digest*

Jika *string* menyatakan pesan (*message*), maka sembarang pesan*M* berukuran bebas dikompresi oleh fungsi hash *H* melalui persamaan :

$$h = H(M)$$

Keluaran fungsi *hash* disebut juga **nilai *hash*** (*hash-value*) atau **pesan-ringkas** (*message digest*). Pada persamaan diatas, *h* adalah nilai *hash* atau *message digest* dari fungsi *H* untuk masukan *M*. Dengan kata lain, fungsi *hash* mengkompresi sembarang pesan yang berukuran berapa saja menjadi *message digest* yang ukurannya selalu tetap (dan lebih pendek dari panjang pesan semula).

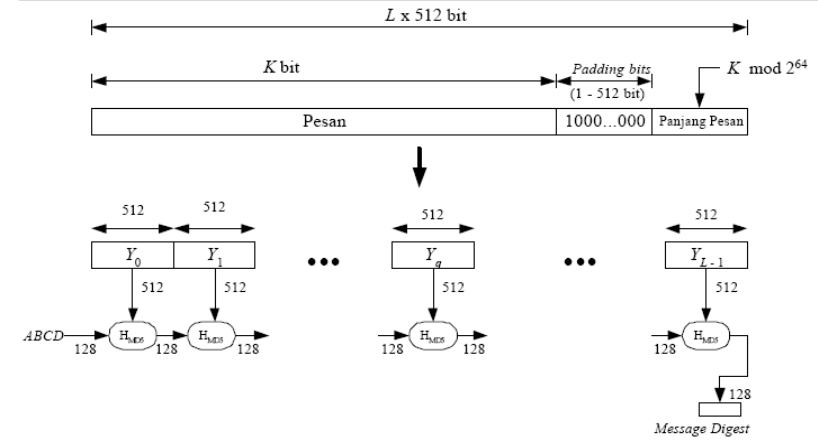
Fungsi hash digunakan untuk memverifikasi kesamaan salinan suatu data denga ndata aslinya. Sebagai contoh pada sebuah perusaahan, daripada

mengirimkan salinan arsip keseluruhan ke komputer pusat (diasumsikan perusahaan ini menggunakan basis data terpusat). Lebih baik mengirimkan *message digest*-nya. Apabila *message digest* salinan arsip sama dengan *message digest* arsip asli, maka salinan arsip tersebut sama dengan arsip didalam basis data.

Fungsi hash sering disebut juga *one-way function* karena fungsi hash bekerja dalam satu arah, yaitu pesan yang sudah di ubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula. Adapun sifat-sifat fungsi hash adalah sebagai berikut :

- Fungsi *H* dapat diterapkan pada blok data berukuran berapa saja.
- *H* menghasilkan nilai (*h*) dengan panjang tetap (*fixedlength output*).
- *H*(*x*) mudah dihitung untuk setiap nilai *x* yang diberikan.
- Untuk setiap *h* yang dihasilkan, tidak mungkin dikembalikan nilai *x* sedemikian sehingga *H*(*x*) = *h*. Itulah sebabnya fungsi *H* dikatakan fungsi *hash* satu-arah (*one-way hash function*).
- Untuk setiap *x* yang diberikan, tidak mungkin mencari *y* ¹*x* sedemikian sehingga *H*(*y*) = *H*(*x*).
- Tidak mungkin mencari pasangan *x* dan *y* sedemikian sehingga *H*(*x*) = *H*(*y*).

Cara kerja kriptografi algoritma MD5 adalah menerima input berupa pesan dengan ukuran sembarang dan menghasilkan *message diggest* yang memiliki panjang 128 bit. Berikut ilustrasi gambar dari pembuatan *message diggest* pada kriptografi algoritma MD5 :



Message Digest dengan algoritma MD5

Menilik dari gambar diatas, secara garis besar pembuatan *message digest* ditempuh melalui empat langkah, yaitu :

1. Penambahan bit bit pengganjal

Proses pertama yang dilakukan adalah menambahkan pesan dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti setelah menambahkan bit-bit pengganjal, kini panjang pesan adalah 64 bit kurang dari kelipatan 512. Hal yang perlu diingat adalah angka 512 muncul karena algoritma MD5 memproses pesan dalam blok-blok yang berukuran 512.

Apabila terdapat pesan dengan panjang 448 bit, maka pesan tersebut akan tetap ditambahkan dengan bit-bit pengganjal. Pesan akan ditambahkan dengan 512 bit menjadi 96 bit. Jadi panjang bit-bit pengganjal adalah antara 1 sampai 512. Lalu satu hal lagi yang perlu diperhatikan adalah bahwasanya bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

2. Penambahan nilai panjang pesan semula

kemudian proses berikutnya adalah pesan ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Apabila panjang pesan lebih besar dari 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. dengan kata lain, jika pada awalnya panjang pesan sama dengan K bit, maka 64 bit yang ditambahkan menyatakan K modulo 2^{64}. sehingga setelah proses kedua ini selesai dilakukan maka panjang pesan sekarang adalah 512 bit.

3. Inisialisasi penyangga MD

Pada algoritma MD5 dibutuhkan empat buah penyangga atau buffer, secara berurut keempat nama penyangga diberi nama A, B, C dan D. Masing-masing penyangga memiliki panjang 32 bit. Sehingga panjang total :

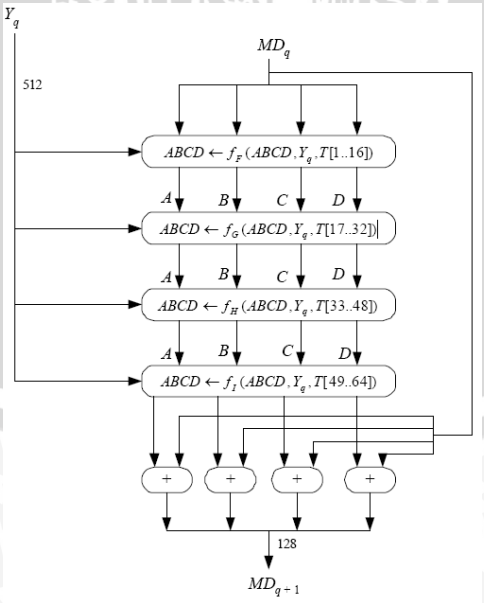
A =	32 bit
B =	32 bit
C =	32 bit
D =	32 bit
	+
total =	128 bit

Keempat penyangga ini menampung hasil antara dan hasil akhir. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi Hexadesimal) sebagai berikut :

A = 01234567
B = 89ABCDEF
C = FEDCBA98
D = 76543210

4. Pengolahan pesan dalam blok berukuran 512 bit

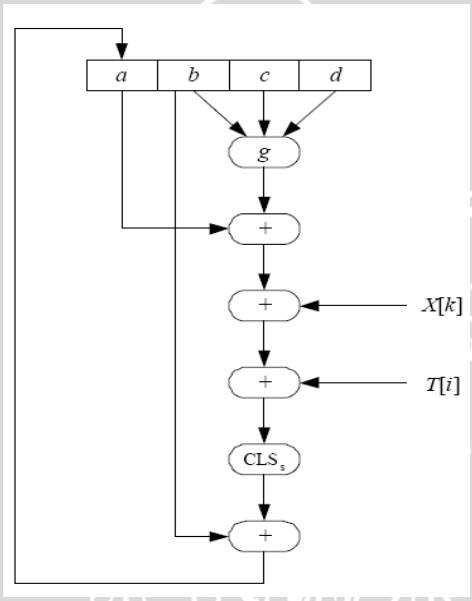
Proses berikutnya adalah pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}). Setelah itu setiap blok 512 bit diproses bersama dengan penyangga MD yang menghasilkan keluaran 128 bit, dan ini disebut H_{MD5}. Berikut ini gambaran dari proses H_{MD5} :



Gambar 6 Pengolahan blok 512 bit (Proses H_{MD5})

Dari gambar diatas dapat kita lihat bahwa proses H_{MD5} terdiri dari 4 buah putaran, dan masing-masing putaran melakukan opersi dasar MD5 sebanyak 16 kali. Dimana disetiap operasi dasar memakai sebuah elemen T. Sehingga setiap putaran memakai 16 elemen tabel T.

Pada gambar 6, Y_q menyatakan blok 512 bit ke-q dari pesan yang telah ditambahkan dengan bit-bit pengganjal pada proses pertama dan tambahan 64 bit nilai panjang pesan semula pada proses kedua. MD_q adalah nilai *message digest* 128 bit dari proses H_{MD5} ke-q. Pada awal proses , MDq berisi nilai inisialisasi penyangga MD. Kemudian fungsi f_F, f_G, f_H, dan f_I pada gambar, masing-masing berisi 16 kali operasi dasar terhadap input, setiap operasi dasar menggunakn elemen tabel T. Berikut ini ilustrasi gambar operasi dasar MD5 :



Gambar 7 Operasi dasar MD5

Operasi dasar MD5 yang diperlihatkan gambar diatas dapat dituliskan dengan persamaan berikut ini :

$$a \leftarrow b + \mathrm{CLS}_s(a + g(b, c, d) + X[k] + T[i])$$

Dimana,

a, b, c, d = empat buah peubah penyangga 32-bit (berisi nilai penyangga A, B, C, D)

g = salah satu fungsi F, G, H, I

CLS_s = *circular left shift* sebanyak s bit

$X[k]$ = kelompok 32-bit ke-k dari blok 512 bit *message* ke-q. Nilai $k = 0$ sampai 15.

$T[i]$ = elemen Tabel T ke-i (32 bit)

$+$ = operasi penjumlahan modulo 2^{32}

Fungsi f_F, f_G, f_H, dan f_I adalah fungsi untuk memanipulasi masukan *a*, *b*, *c*, dan *d* dengan ukuran 32-bit. Masing-masing fungsi dapat dilihat pada Tabel 1 dibawah ini :

Tabel 1 Fungsi-fungsi dasar MD5

Nama	Notasi	$g(b, c, d)$
f_F	$F(b, c, d)$	$(b \wedge c) \vee (\sim b \wedge d)$
f_G	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \sim d)$
f_H	$H(b, c, d)$	$b \oplus c \oplus d$
f_I	$I(b, c, d)$	$c \oplus (b \wedge \sim d)$

NB : secara berurut peartor logika AND, OR, NOT dan XOR dilambangkan dengan □, □, ~, □

Kemudian nilai T[i] dapat dilihat pada tabel dibawah ini. Tabel ini disusun oleh fungsi 2^{32} x abs(sin(i)), i dalam radian.

Tabel 2. Nilai T[i]

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 69D96122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBCB	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

Sebagaimana telah dijelaskan sebelumnya bahwa fungsi f_F, f_G, f_H, dan f_I melakukan 16 kali operasi dasar. Misalkan notasi berikut ini :

$$[abcd \quad k \quad s \quad i]$$

Menyatakan operasi

$$a \leftarrow b + ((a + g(b, c, d) + X[k] + T[i])<<<s)$$

untuk operasi diatas, <<<s melambangkan opersi *circular left shift* 32 bit, maka operasi dasar pada masing-masing putaran dapat ditabulasikan sebagai berikut :

- putaran 1 : 16 kali operasi dasar dengan g(b, c, d) – F(b, c, d), dapat dilihat pada tabel berikut ini :

Tabel 3 Rincian operasi pada fungsi F(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	0	7	1]
2	[DABC	1	12	2]
3	[CDAB	2	17	3]
4	[BCDA	3	22	4]
5	[ABCD	4	7	5]
6	[DABC	5	12	6]
7	[CDAB	6	17	7]
8	[BCDA	7	22	8]
9	[ABCD	8	7	9]
10	[DABC	9	12	10]
11	[CDAB	10	17	11]
12	[BCDA	11	22	12]
13	[ABCD	12	7	13]
14	[DABC	13	12	14]
15	[CDAB	14	17	15]
16	[BCDA	15	22	16]

- putaran 2 : 16 kali operasi dasar dengan g(b, c, d) – G(b, c, d), dapat dilihat pada tabel berikut ini :

Tabel 4 Rincian operasi pada fungsi G(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	1	5	17]
2	[DABC	6	9	18]
3	[CDAB	11	14	19]
4	[BCDA	0	20	20]
5	[ABCD	5	5	21]
6	[DABC	10	9	22]
7	[CDAB	15	14	23]
8	[BCDA	4	20	24]
9	[ABCD	9	5	25]
10	[DABC	14	9	26]
11	[CDAB	3	14	27]
12	[BCDA	8	20	28]
13	[ABCD	13	5	29]
14	[DABC	2	9	30]
15	[CDAB	7	14	31]
16	[BCDA	12	20	32]

- putaran 3 : 16 kali operasi dasar dengan g(b, c, d) – H(b, c, d), dapat dilihat pada tabel berikut ini :

Tabel 5 Rincian operasi pada fungsi H(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	5	4	33]
2	[DABC	8	11	34]
3	[CDAB	11	16	35]
4	[BCDA	14	23	36]
5	[ABCD	1	4	37]
6	[DABC	4	11	38]
7	[CDAB	7	16	39]
8	[BCDA	10	23	40]
9	[ABCD	13	4	41]
10	[DABC	0	11	42]
11	[CDAB	3	16	43]
12	[BCDA	6	23	44]
13	[ABCD	9	4	45]
14	[DABC	12	11	46]
15	[CDAB	15	16	47]
16	[BCDA	2	23	48]

- putaran 4 : 16 kali operasi dasar dengan g(b, c, d) – I(b, c, d), dapat dilihat pada tabel berikut ini :

Tabel 6 Rincian operasi pada fungsi I(b, c, d)

No.	[abcd	k	s	i]
1	[ABCD	0	6	49]
2	[DABC	7	10	50]
3	[CDAB	14	15	51]
4	[BCDA	5	21	52]
5	[ABCD	12	6	53]
6	[DABC	3	10	54]
7	[CDAB	10	15	55]
8	[BCDA	1	21	56]
9	[ABCD	8	6	57]
10	[DABC	15	10	58]
11	[CDAB	6	15	59]
12	[BCDA	13	21	60]
13	[ABCD	4	6	61]
14	[DABC	11	10	62]
15	[CDAB	2	15	63]
16	[BCDA	9	21	64]

Setelah putaran keempat, a, b, c dan d di tambahkan ke A, B, C dan D yang selanjutnya algoritma akan memproses untuk blok data berikutnya (Y_{q+1}). Output akhir dari algoritma MD5 adalah hasil penyambungan bit-bit di A, B, C dan D.

Dari uraian diatas, secara umum fungsi hash MD5 dapat ditulis dalam persamaan matematis sebagai berikut :

$$MD_0 = IV$$

$$MD_{q+1} = MD_q + f_I(Y_q + f_H(Y_q + f_G(Y_Q + f_F(Y_q + MD_q))))$$

$$MD = MD_{L-1}$$

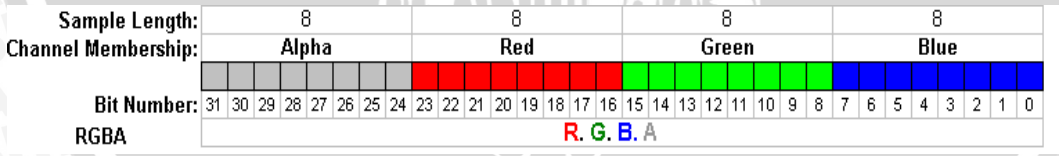
Dimana,

IV = *initial vector* dari penyangga ABCD, yang dilakukan pada proses inisialisasi penyangga.
Y_q = blok pesan berukuran 512-bit ke-q
L = jumlah blok pesan
MD = nilai akhir *message digest*

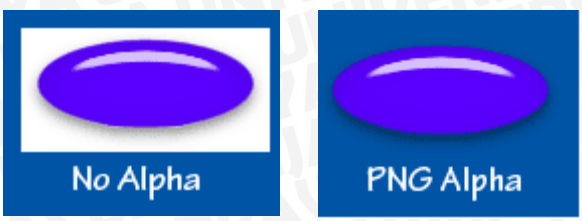
2.4 Alpha Channel

Alpha channel adalah channel tambahan yang dapat ditambahkan ke sebuah gambar. Channel ini berisi informasi transparansi tentang gambar. Tergantung dari jenis alfa, channel ini dapat berisi berbagai tingkat transparansi. Alpha channel pada dasarnya mengontrol transparansi dari semua layer warna lainnya. Dengan menambahkan alpha channel untuk gambar, maka akan dapat mengontrol transparansi saluran merah, kanal hijau dan saluran biru.

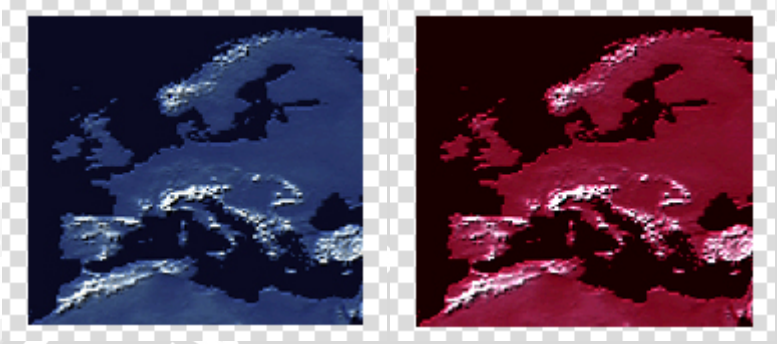
Pada dasarnya sebuah gambar yang memiliki alpha channel memiliki 4 buah layer, yaitu Red, Green, Blue, dan Alpha.



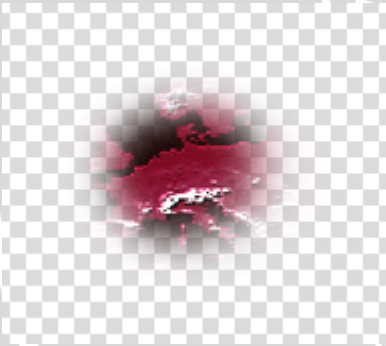
Alpha channel dapat digunakan dalam beberapa cara yang berbeda. Dalam kebanyakan kasus alpha channel digunakan untuk menghasilkan gambar yang halus di atas berbagai latar belakang warna yang dipakai oleh gambar tersebut. Tentu saja hal ini adalah keuntungan terbesar dari alpha channel.



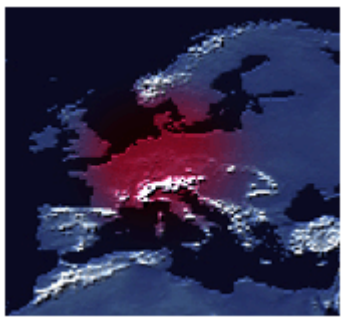
Kita dapat menciptakan berbagai efek special dengan meng-copy sebuah gambar dan mengaturnya dalam warna yang berbeda, kemudian menghapus variasi pixel seperti yang kita inginkan kemudian menggabungkannya.



Sebagai contoh kita mempunyai dua buah gambar. Kemudian dengan menggunakan partial eraser kita menghapus gambar 2 dengan meninggalkan bagian tengah dan bagian lainnya menjadi transparan.



Kemudian kita menggabungkan kedua gambar tersebut



Alpha transparansi tidak merubah warna dalam layer RGB. Ketika kita tampaknya menghapus pixel dalam alpha tranparansi, sebenarnya warna pixel yang terdapat di dalamya tidak berubah dan masih ada. Warna itu hanya

transparan karena nilai-nilai alpha channel telah diset pada nilai tinggi (range : 0 (fully opaque), 255 (completely transparent).

2.5 TIFF (Tagged Image File Format)

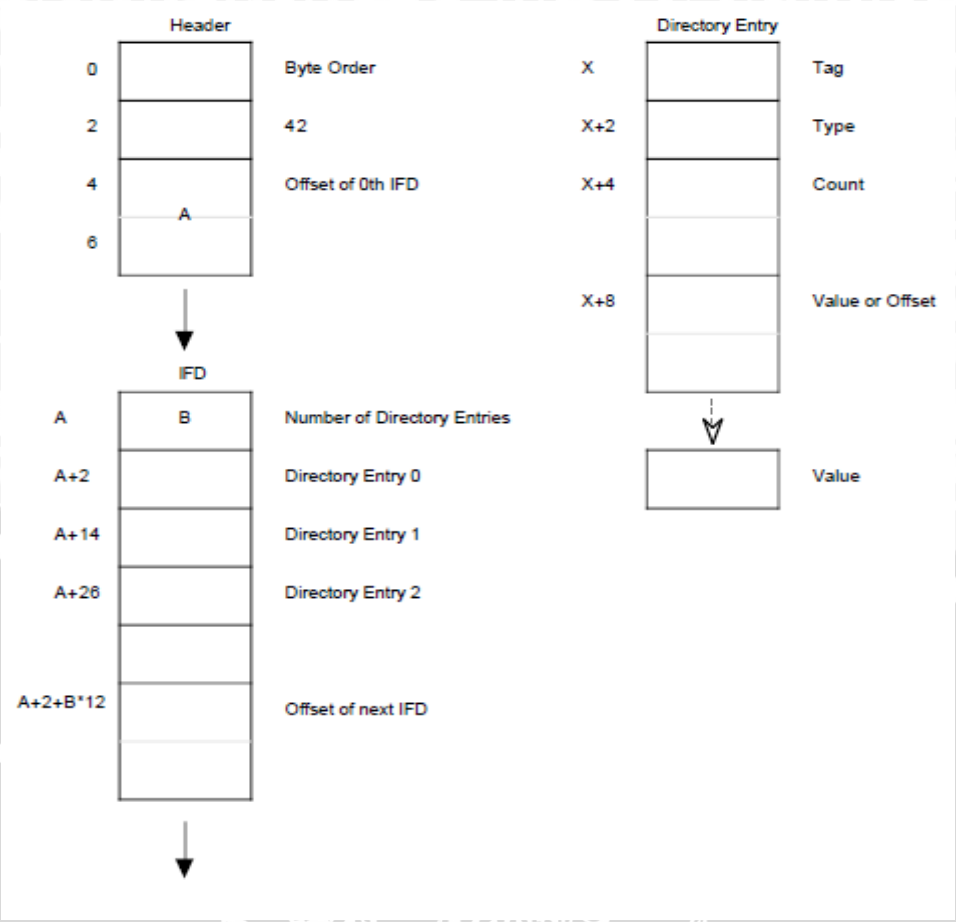
TIFF adalah format paling universal dan paling didukung di semua platform , Mac, Windows, Unix. Format TIFF juga paling flexible dan pada normalnya menyimpan 8 bits atau 16 bits per warna dan biasanya menggunakan ekstensi .tiff atau .tif. TIFF mempunyai kompresi lossless, kompresi tersebut tidak mempengaruhi kualitas gambar karena kompresinya tidak menghilangkan degadrasi warna. TIFF juga menjadi standart format dalam dunia fotografi. Namun sayangnya format ini tidak didukung oleh kebanyakan web browsers.

TIFF biasanya adalah keluaran terbaik yang dihasilkan oleh kamera digital. Kamera digital sering menawarkan keluaran berupa 3 level JPEG ditambah TIFF. Dikarenakan JPEG mempunyai beberapa loss quality sedangkan TIFF mempunyai kualitas yang baik. Tetapi ukuran file yang dihasilkan adalah saah satu kekurangan format TIFF karena ukurannya yang besar.

Format TIFF dapat mempunyai 3 atau 4 buah channels yang berupa RGB dan alpha. Sebenarnya channel alpha ini semacam attribute bagi sebuah gambar karena pada bagian channel tersebut tidak terdapat sebuah gambar yang jelas yang dapat dideskripsikan secara jelas melalui satu channel itu saja tetapi channel tersebut dapat dideteksi dengan menggabungkan channel lainnya. Sesuai standar industry tahun 2005, tiap channel terdiri dari 8 bit pixel warna.

2.5.1 Format File TIFF

TIFF adalah sebuah image file format. Sebuah TIFF file dimulai dengan sebuah image file header (IFH) yang menunjuk ke sebuah image file directory (IFD). Sebuah IFD berisi informasi tentang gambar dan juga menunjuk pada data gambar sebenarnya.



2.5.2 Image File Header (IFH)

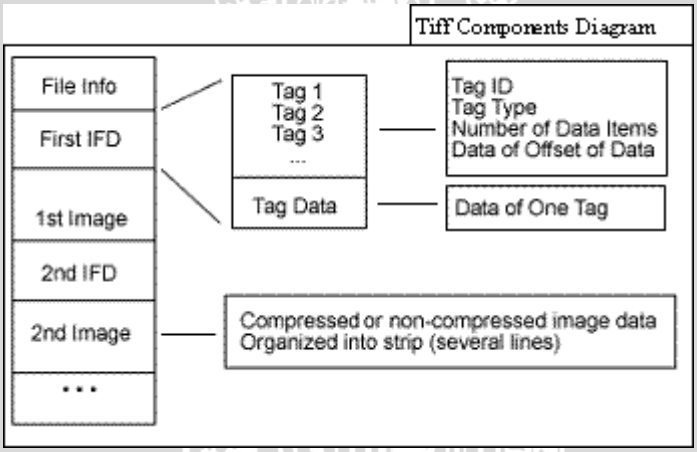
Sebuah IFH mempunyai ukuran tetap yaitu 8 bytes yang berisi informasi sebagai berikut:

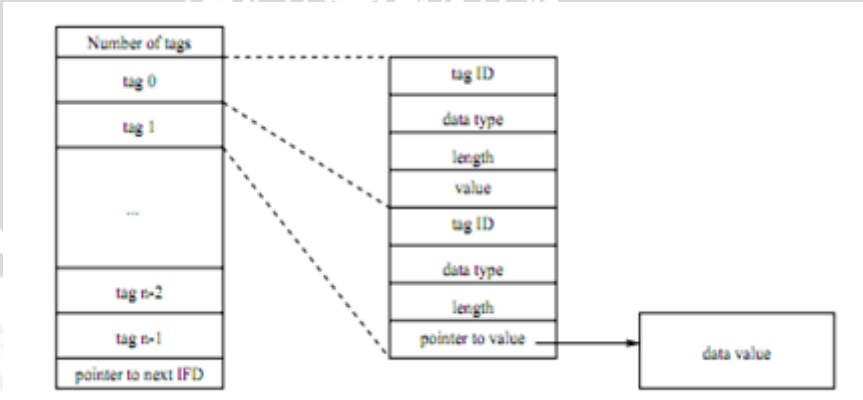
- Bytes 0-1: Word (2 bytes) pertama mengindikasikan urutan tipe yang digunakan dalam format file. 0x4D4D untuk Big-Endian (MSB to LSB) dan 0x4949 untuk Little-Endian (LSB to MSB).
- Bytes 2-3: Word kedua mengindikasikan versi TIFF file yang selalu 42
- Bytes 4-7: menunjukkan offset (dalam bytes) ke IFD pertama. Direktori mungkin berada di sembarang lokasi dalam file. Pembaca harus mengikuti pointer yang ditunjukkan oleh nilai offset tersebut.

2.5.3 Image File Directory (IFD)

Sebuah Image File Directory (IFD) terdiri dari hitungan 2-byte dari jumlah direktori
entri (yaitu, jumlah fileds), diikuti dengan urutan 12-byte bidang entri, diikuti oleh 4-byte offset IFD berikutnya (atau 0 jika tidak ada). Ukuran IFD bermacam-macam. Setiap IFD terisi oleh bidang entri (tags). Ukuran tiap tags tetap yaitu 12 bytes.

- Word pertama mengindikasikan jumlah tags di IFD dan diikuti tags-nya.
- 2 word terakhir menunjuk pada IFD selanjutnya. Pointer di IFD terakhir adalah NULL.





Setiap tags terdiri dari 4 element:

- Tag ID : identitas tag

- Data type : tipe data yang berisi dalam tag ini
- Length : nomor data yang ada dalam tag ini
- Value / value pointer : berisi nilai data jika nilai data lebih kecil atau sama dengan 4 byte, jika tidak maka mengarah ke data block.

Jenis-jenis tipe data :

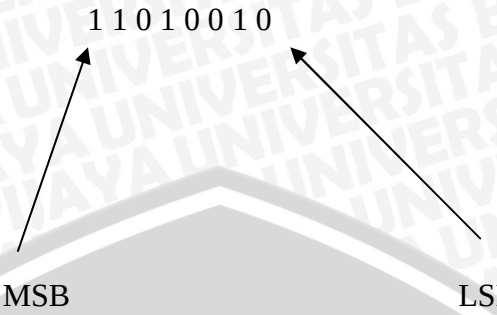
Type	Code	Description
BYTE	1	8-bit, unsigned byte
ASCII	2	8-bit
SHORT	3	16-bit, unsigned number
LONG	4	32-bit, unsigned number
RATIONAL	5	Two 32-bit, unsigned number
SBYTE	6	8-bit, signed integer
UNDEFINED	7	An 8-bit byte that may contain anything, depending on the definition of the field
SSHORT	8	A 16-bit (2-byte) signed (twos-complement) integer
SLONG	9	A 32-bit (4-byte) signed (twos-complement) integer
SRATIONAL	10	Two SLONG's: the first represents the numerator of a fraction, the second the denominator
FLOAT	11	Single precision (4-byte) IEEE format
DOUBLE	12	Double precision (8-byte) IEEE format

Offset (hex)	Description	Value (numeric values are expressed in hexadecimal notation)			
Header:					
0000	Byte Order	4D4D			
0002	42	002A			
0004	1st IFD offset	00000014			
IFD:					
0014	Number of Directory Entries	000C			
0016	NewSubfileType	00FE	0004	00000001	00000000
0022	ImageWidth	0100	0004	00000001	000007D0
002E	ImageLength	0101	0004	00000001	00000BB8
003A	Compression	0103	0003	00000001	8005 0000
0046	PhotometricInterpretation	0106	0003	00000001	0001 0000
0052	StripOffsets	0111	0004	000000BC	000000B6
005E	RowsPerStrip	0116	0004	00000001	00000010
006A	StripByteCounts	0117	0003	000000BC	000003A6
0076	XResolution	011A	0005	00000001	00000696
0082	YResolution	011B	0005	00000001	0000069E
008E	Software	0131	0002	0000000E	000006A6
009A	DateTime	0132	0002	00000014	000006B6
00A6	Next IFD offset	00000000			
Values longer than 4 bytes:					
00B6	StripOffsets	Offset0, Offset1, ... Offset187			
03A6	StripByteCounts	Count0, Count1, ... Count187			
0696	XResolution	0000012C 00000001			
069E	YResolution	0000012C 00000001			
06A6	Software	"PageMaker 4.0"			
06B6	DateTime	"1988:02:18 13:59:59"			
Image Data:					
00000700		Compressed data for strip 10			
xxxxxxxx		Compressed data for strip 179			
xxxxxxxx		Compressed data for strip 53			
xxxxxxxx		Compressed data for strip 160			
.					
.					
End of example					

2.6 Least-Significant Bit Coding

Teknik paling dasar dalam steganografi pada citra/gambar disebut Least-Significant Bit Coding. Bit atau binary digit adalah unit dasar penyimpanan data di dalam komputer, nilai bit suatu data adalah 0 atau 1. Semua data yang ada pada komputer disimpan ke dalam satuan bit ini, termasuk gambar, suara, ataupun video.

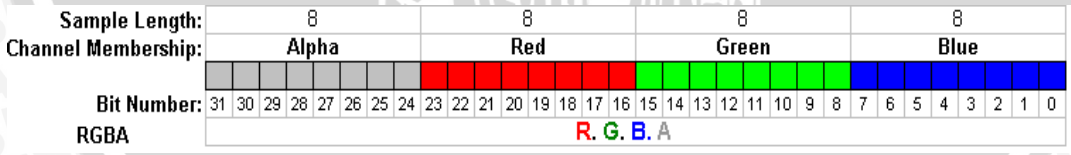
Contoh :



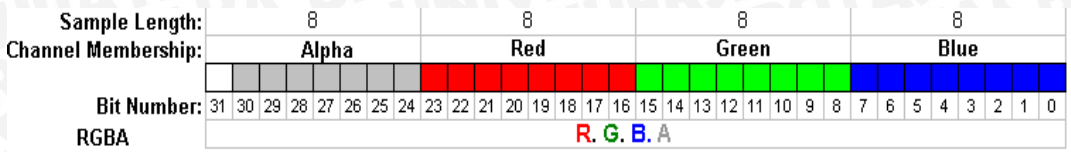
Bit yang cocok untuk diganti adalah bit LSB, sebab perubahan tersebut hanya mengubah nilai byte satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan byte tersebut menyatakan warna merah, maka perubahan satu bit LSB tidak mengubah warna merah tersebut secara berarti. Lagi pula, mata manusia tidak dapat membedakan perubahan yang kecil.

Format pewarnaan di dalam media gambar, seperti grayscale, RGB, dan CMYK, juga menggunakan satuan bit ini dalam penyimpanannya. Sebagai contoh pewarnaan monochrome bitmap (menggunakan 1 bit untuk tiap pixelnya), RGB – 24 bit (8 bit untuk Red, 8 bit untuk Green, dan 8 bit untuk Blue), Grayscale-8 bit (menentukan tingkat kehitaman suatu pixel berdasarkan nilai bitnya).

Suatu gambar 32 bit mempunyai struktur tiap bit pixel seperti gambar di bawah ini.



Gambar yang disisipkan merupakan gambar yang mempunyai warna hitam putih dengan menggeser bit alpha untuk diisi dengan gambar yang akan disisipkan.



2.7 Pengukuran Kualitas Citra

Dalam melakukan encode citra, biasanya terdapat perubahan-perubahan seperti efek blurring, sharpening, atau timbulnya noise. Oleh karena itu, biasanya dilakukan pengukuran kualitas citra terlebih dahulu untuk mengetahui bagaimana hasil encode citra tersebut; apakah kualitasnya mirip dengan citra yang asli atau tidak. Terdapat dua cara pengukuran, yaitu subjektif dan objektif. Subjektif berarti kualitas citra ditentukan oleh para pengamat citra tersebut. Pendapat dari pengamat akan beraneka ragam, karena sangat tergantung dari persepsi dan standar masing-masing[WAN-05].

Cara objektif merupakan pengukuran secara matematika terhadap citra yang diukur, dan dapat dikerjakan otomatis oleh komputer. Ada atau tidaknya citra referensi, mengklasifikasikan cara ini menjadi tiga jenis, yaitu apabila citra referensi tersedia secara penuh (full reference), hanya sebagian citra referensi yang tersedia (reduced reference), atau tidak tersedianya citra referensi (no reference). Pengukuran citra secara objektif biasanya dilakukan pada full reference, sehingga terdapat citra asli sebagai perbandingan[WAN-05].

Metode pengukuran kualitas citra secara objektif dilaksanakan dengan mengukur kualitas pada citra. Salah satu contoh metode pengukuran kualitas antara dua citra adalah PSNR.

2.8 Peak Signal to Noise Radio (PSNR)

Perhitungan PSNR akan menilai kemiripan gambar berdasarkan besarnya perbedaan yang dianggap sebagai kerusakan pada salah satu gambar. Nilai pada PSNR merupakan hasil pembagian dari kekuatan sinyal maksimal yang diterima, dengan sinyal noise. Cara menghitung PSNR ditunjukkan pada persamaan[ARY-08]:

$$PSNR = 10log\frac{m^2}{MSE}dB \quad (2\text{-}1)$$

dimana *m* adalah nilai maksimum yang mungkin dimiliki oleh suatu *pixel*. Sebagai contoh, untuk data citra 8 bit, nilai maksimumnya adalah 255. Satuan

nilai PSNR adalah desibel (dB). Sedangkan MSE, atau Mean Square Error adalah suatu nilai noise. Tingkat kemiripan yang tinggi akan didapat apabila nilai error (MSE) yang dimasukkan kecil, sehingga nilai PSNR menjadi besar. Nilai MSE didapatkan dengan persamaan berikut ini[ARY-08]:

$$MSE = \frac{1}{XY}\sum_{x}\sum_{y}\left[I(x,y) - I'(x,y)\right]^2 \quad (2\text{-}2)$$

dimana $I(x,y)$ adalah nilai *grey-level* citra asli di posisi (x,y) pada frame *t*, *I'* adalah nilai derajat keabuan citra yang telah diberi *steganograf* di posisi (x,y), *X* dan *Y* adalah ukuran panjang dan lebar. MSE tidak mempunyai besaran nilai. Suatu sistem dikatakan mempunyai kinerja yang baik apabila nilai MSE mendekati nol atau dapat dikatakan tidak ada nya kesalahan pada sistem tersebut. Perlu diperhatikan bahwa nilai PSNR hanya terdefinisi dengan baik pada pengukuran *luminance* (intensitas cahaya, misalnya pada citra *grayscale*). Tidak ada kesepakatan pengukuran nilai PSNR untuk data citra / video berwarna. Salah satu pendekatan yang dapat dilakukan adalah dengan menghitung PSNR untuk masing-masing kanal dan menghitung nilai rata-ratanya. Nilai MSE yang rendah akan lebih baik, sedangkan nilai PSNR yang tinggi akan lebih baik. Nilai PSNR yang baik pada perbandingan dua berkas citra adalah diatas 60 dB dB[ARY-08].

2.9 Bahasa Pemrograman C

Bahasa pemrograman C merupakan salah satu bahasa pemrograman komputer. Dibuat pada tahun 1972 oleh Dennis Ritchie untuk Sistem Operasi Unix di Bell Telephone Laboratories. Meskipun C dibuat untuk memprogram sistem dan jaringan komputer namun bahasa ini juga sering digunakan dalam mengembangkan software aplikasi. C juga banyak dipakai oleh berbagai jenis platform sistem operasi dan arsitektur komputer, bahkan terdapat beberapa compiler yang sangat populer telah tersedia. C secara luar biasa memengaruhi bahasa populer lainnya, terutama C++ yang merupakan extensi dari C.

Salah satu keunggulannya adalah alokasi memori. Tidak seperti bahasa komputer yang lain, C memungkinkan programmer untuk menulis secara langsung ke memori. Kunci konstruksi di C seperti struct, pointer dan array dirancang untuk struktur, dan memanipulasi memori dengan cara efisien yaitu berdasrakan machine-independent. Secara khusus, C memberikan kontrol atas tata letak memori struktur data. Selain itu alokasi memori dinamis berada di bawah kendali programmer. Ini biasanya hal yang baik, karena berhubungan dengan alokasi memori ketika membangun sebuah program tingkat tinggi adalah proses yang sangat rentan terhadap kesalahan. Namun, ketika berhadapan dengan kode tingkat rendah seperti bagian dari OS yang mengontrol perangkat, C menyediakan seragam, antarmuka bersih.

2.10 Microsoft Visual Studio

Microsoft Visual Studio adalah sebuah peralatan pengembangan (IDE) dari Microsoft. Visual Studio dapat digunakan untuk mengembangkan aplikasi baik dengan console atau graphical user interface (GUI) bersamaan untuk aplikasi berbasis windows, website, aplikasi web, dan web service.

Visual Studio menyertakan code editor IntelliSense pendukung serta refactoring kode. Built-in tools termasuk desainer untuk membangun aplikasi GUI, web designer dan perancang skema database. Dan juga dapt menerima plug-in yang menigkatkan fungsionalitas pada hampir setiap tingkat termasuk untuk menambahkan dukungan untuk source control system (seperti subversion).

Visual studio banyak mendukung bahasa pemrograman yang berbeda, yang memungkinkan kode editor dan debugger dalam berbagai tingkat. Visual studio memberikan layanan bahasa c/c++, c#, VB.NET. Dukungan bahasa lain seperti M, Phyton, dan Ruby yang dapat diinstal secara terpisah. Juga mendukung XML/XSLT, HTML / XHTML, JavaScript dan CSS.

BAB III

Metodologi Penelitian

Bab ini menjelaskan mengenai langkah-langkah yang dilakukan untuk membuat aplikasi penyisipan gambar. Langkah-langkah yang diperlukan antara lain studi literatur, perancangan sistem, penentuan batasan algoritma yang digunakan, implementasi, pengujian dan analisis, pengambilan kesimpulan dan saran, serta penulisan laporan.

3.1 Studi Literatur

Berupa kajian pustaka yang mendukung pembuatan aplikasi yang berupa :

1. Kajian pustaka mengenai stuktur data citra digital, dalam hal ini citra gambar berformat *tiff* yang digunakan sebagai media penyimpan data.
2. Kajian pustaka mengenai teori dasar algoritma AES.
3. Kajian pustaka mengenai fungsi Hash MD5
4. Kajian pustaka mengenai metode steganografi yang digunakan untuk teknik menyembunyikan gambar dalam citra digital.
5. Kajian pustaka mengenai struktur dan bahasa pemrograman yang digunakan yaitu bahasa c terutama c++.

3.2 Perancangan

Secara garis besar perancangan terdiri dari perancangan sistem dan cara kerja sistem,. Perancangan sistem yaitu merancang blok sistem diagram untuk menggambarkan sistem secara global dan selanjutnya dilakukan analisis sesuai kebutuhan. Analisis kebutuhan digunakan untuk menentukan kebutuhan-kebutuhan fungsi, algoritma yang digunakan, dan batasan perancangan dan implementasi. Perancangan ini akan didahului dengan cara kerja sistem kemudian macam penyisipan data. Berikut ini adalah alur utama dari sistem :

Gambar 3.2 Diagram Cara Kerja Sistem

Cara kerja sistem:

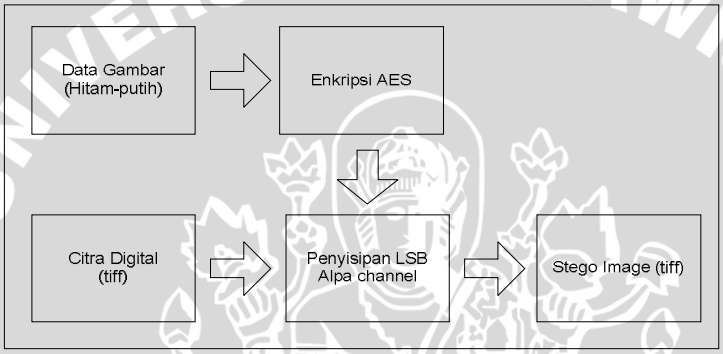
1. Program mempunyai masukan data berupa gambar tiff untuk citra yang ditumpangi dan gambar hitam putih sebagai citra gambar yang akan disisipkan.
2. Terdapat dua macam penyisipan yaitu data gambar hitam putih akan dienkripsi kumudian hasil enkripsi akan disisipkan ke alpa channel sedangkan hasil MD5 gambar hitam putih akan disisipkan ke red channel

3. Keluaran program adalah gambar .tiff yang di dalamnya terdapat gambar yang disisipkan yaitu berupa gambar hitam putih yang terenkripsi.
4. Untuk pendeteksian gambar maka akan terdapat informasi berupa data gambar enkripsi, gambar setelah dekripsi dan data hash 16-byte dalam hexadesimal dan desimal.

3.2.1 Macam Penyisipan :

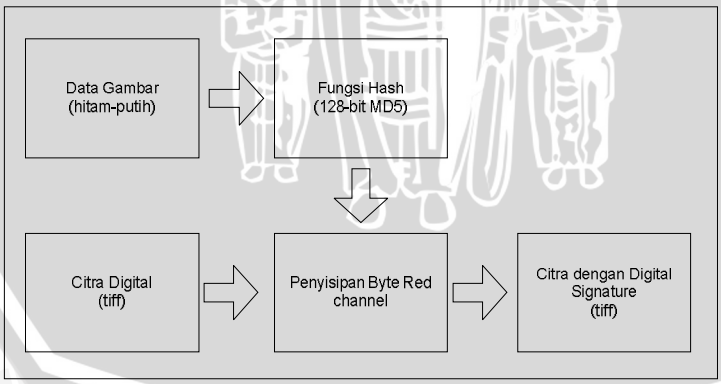
2. Penyisipan data terenkripsi

   Penyisipan data terenkripsi merupakan penyisipan yang dilakukan untuk menyisipkan hasil enkripsi data ke dalam LSB channel alpha.



3. Penyisipan Digital Signature

   Proses ini menyisipkan data pada channel red citra penampung, hanya sepanjang 16 byte pertama dari channel red.



3.3 Implementasi

Proses selanjutnya adalah implementasi yakni, proses transformasi hasil perancangan perangkat lunak yang telah dibuat ke dalam kode *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan yaitu menerapkan bahasa pemrograman C++ dengan menggunakan Microsoft Visual Studio 2010.

Implementasi meliputi :

8. Implementasi metode pengenkripsian
9. Implementasi Enkripsi Data
10. Implementasi Digital Signature

3.4 Pengujian dan Analisis

Pengujian aplikasi yaitu untuk mengetahui kesesuaian analisa kebutuhan yang dibuat dengan implementasi aplikasi. Analisis sistem dilakukan dengan membandingkan sistem aplikasi yang dengan teori yang ada sehingga didapatkan suatu kesimpulan.

3.4.1 Pengujian Program Penyisipan dan Ekstraksi Gambar

Pengujian dilakukan dengan menjalankan program penyisipan citra gambar kemudian dilakukan ekstraksi untuk memperoleh citra gambar asli sebelum disisipkan.

3.4.2 Pengujian Kualitas Gambar dengan penilaian Subjektif dan Objektif

Pengujian ini dilakukan untuk menguji kualitas dari citra hasil penyisipan, yaitu dengan membandingkannya dengan citra yang asli. Untuk pengujian aplikasi ini dibagi dua macam cara yaitu:

3. Untuk pengujian subjektif dilakukan dengan memberikan citra hasil penyisipan dan citra yang asli kepada responden sebagai sampel untuk dianalisa terlihat adanya perbedaan atau tidak antara citra tersebut.
4. Untuk pengujian objektif dilakukan dengan menghitung nilai Peak Signal to Noise Ratio (PSNR) antara citra hasil penyisipan dan citra asli.

3.4.3 Pengujian Manipulasi Citra Penampung

Pengujian ini dilakukan untuk mengetahui sejauh mana data di dalam gambar berubah atau tidak ketika dilakukan manipulasi terhadap gambar penampung. Manipulasi yang dilakukan antara lain adalah :

1. Melakukan *cropping* pada gambar
2. Melakukan *resize* resolusi gambar

3.5 Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan dan didasarkan pada kesesuaian antara teori dan praktek. Kesimpulan diambil untuk menjawab rumusan masalah yang telah ditetapkan sebelumnya. Tahap yang paling akhir adalah pengambilan saran terhadap penelitian yang selanjutnya, sehingga bisa menyempurnakan kekurangan yang ada dan mengembangkan pada tingkat pokok kajian yang lebih lanjut.

BAB IV
PERANCANGAN

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan yang dilakukan meliputi perancangan sistem, cara kerja sistem, dan penyisipan data.

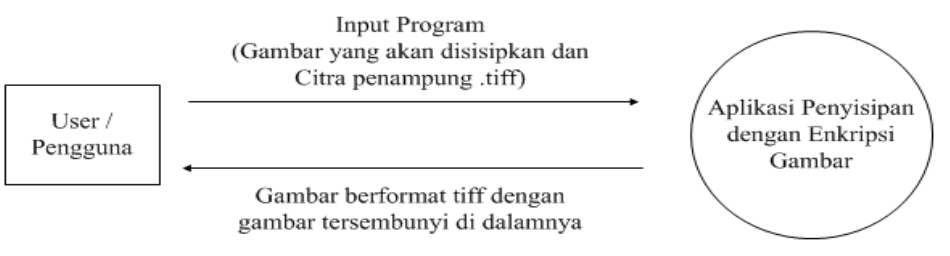
4.1 Perancangan Sistem

Perancangan sistem merupakan tahap awal dari perancangan perangkat lunak. Perancangan ini dilakukan untuk mengetahui aplikasi sistem yang akan dibuat secara umum. Perancangan sistem ini akan didahului dengan pendefinisian pelaku atau user yang akan menggunakan progam ini dan juga alat bantu yang digunakan, yaitu:

1. User : Pelaku yang akan mengambil citra dan juga menjalankan program dengan tujuan untuk menyisipkan gambar rahasia.
2. Program : Perangkat lunak yang digunakan menyisipkan gambar rahasia ke dalam citra penampung (citra penampung yang digunakan adalah tiff dan citra yang disisipkan dalam jpeg).

4.1.1 Diagram Konteks

Diagram konteks adalah sebuah diagram sederhana yang menggambarkan hubungan dengan entitas luar, masukan dan keluaran dari sistem. Diagram konteks aplikasi penyisipan gambar dengan enkripsi ditunjukkan pada gambar 4.1



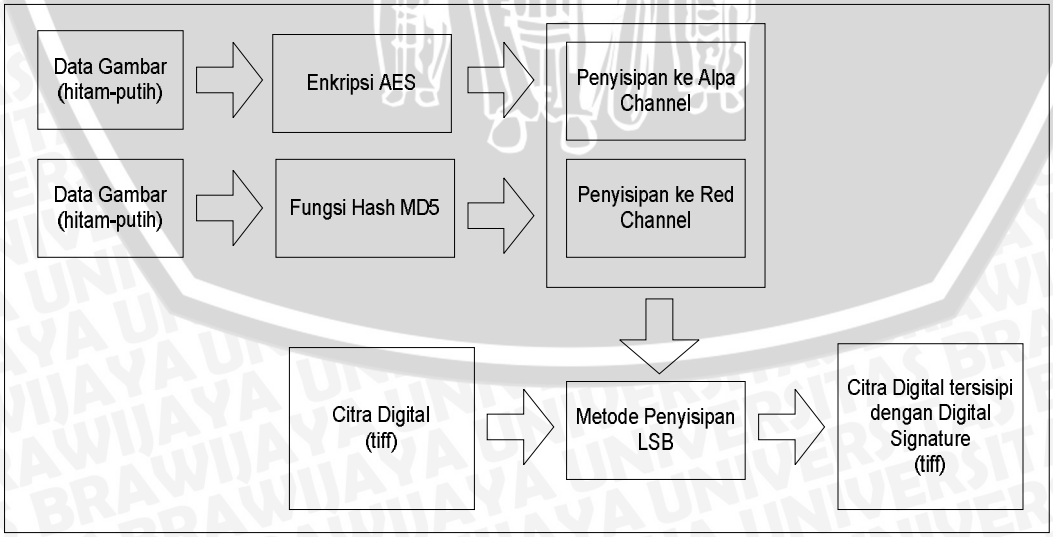
Gambar 4.1 Diagram Konteks

4.2 Cara Kerja Sistem

Cara kerja aplikasi penyisipan gambar adalah sebagai berikut:

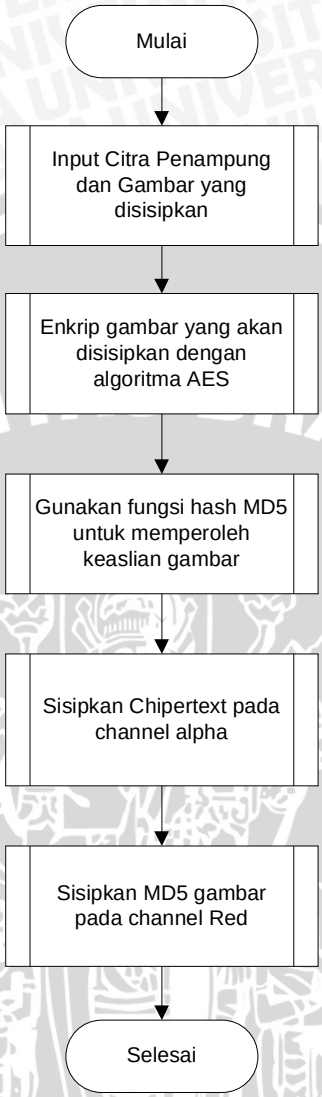
1. Program mempunyai masukan data berupa gambar tiff untuk citra yang ditumpangi dan gambar hitam putih sebagai gambar yang akan disisipkan.
2. Terdapat dua macam penyisipan yaitu data gambar hitam putih akan dienkripsi kumudian hasil enkripsi akan disisipkan ke alpa channel sedangkan hasil MD5 gambar hitam putih akan disisipkan ke red channel
3. Keluaran program adalah citra gambar berformat tiff yang di dalamnya terdapat citra gambar yang disisipkan yaitu berupa gambar hitam putih yang terenkripsi.
4. Untuk pendeteksian citra gambar maka akan terdapat informasi berupa data gambar enkripsi, gambar setelah dekripsi dan data hash 16-byte dalam hexadesimal dan desimal.

Secara umum program crypto-steganografi ini mempunyai fungsi untuk menyembunyikan informasi berupa data digital didalam data digital lainnya dalam hal ini media yang digunakan adalah citra digital yaitu gambar yang mempunyai format tiff.



Gambar 4.2 Metode Sistem Kriptografi dengan Digital Signature

Proses-proses dari aplikasi penyisipan dan aplikasi pengenkripsi dengan metode AES akan di gambarkan dengan diagram alir.



Gambar 4.3 Diagram Alir Sistem secara Keseluruhan

Proses dari diagram alir dapat dijelaskan sebagai berikut :

1. Program menyediakan input untuk aplikasi yaitu berupa citra penampung (dalam format tiff) dan gambar yang akan disisipkan (dalam format jpeg).
2. User memasukkan kedua file input tersebut kemudian program akan menampilkan file input.
3. Setelah memasukkan kedua gambar maka program akan merubah gambar yang disisipkan menjadi gambar hitam putih dengan perlakuan oleh pengguna / user.

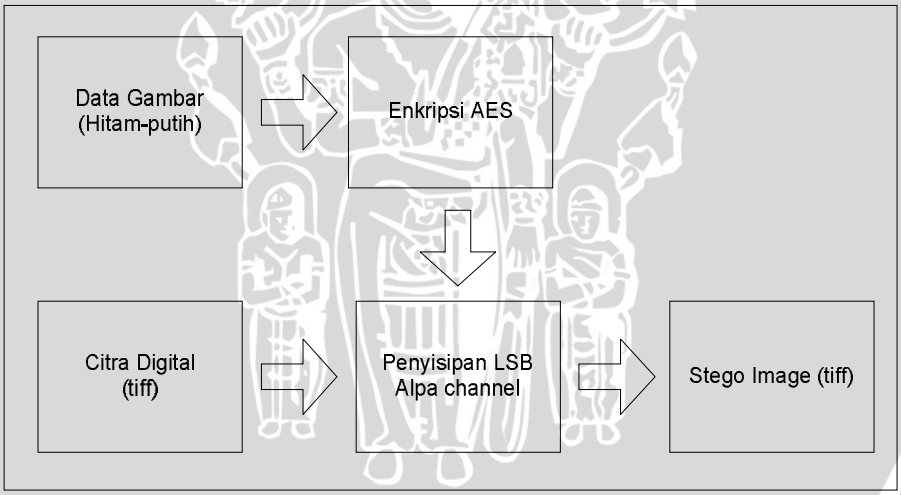
4. Setelah menjadi gambar hitam putih maka gambar akan dienkripsi oleh program. Pengguna dapat melihat hasil enkripsi pada proses list yang ditampilkan oleh program.
5. Bila user menekan tombol sisip maka program akan meyisipkan hasil enkripsi (chipertext) ke dalam channel alpa citra penampung dan hash gambar (MD5) ke channel red citra penampung.

4.3 Penyisipan Data

Dalam sistem ini data enkripsi dan digital signature dipisahkan letak penyisipannya, sehingga sistem ini menggunakan 2 macam penyisipan data, yaitu penyisipan pada data terenkripsi dan penyisipan *digital signature*.

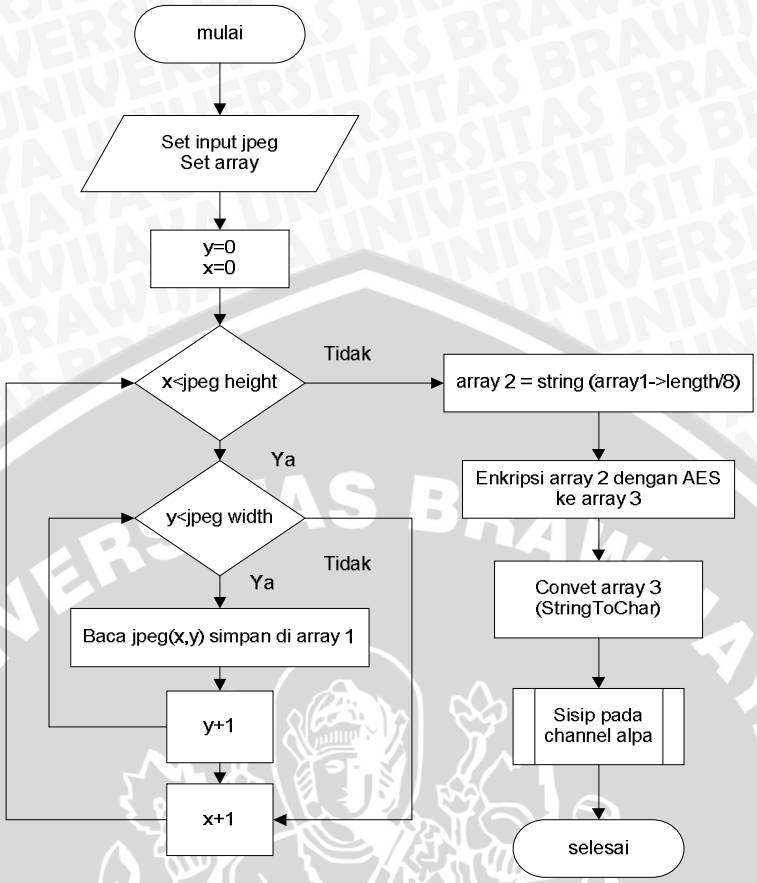
4.3.1 Penyisipan Pada Data Terenkripsi

Penyisipan data terenkripsi merupakan penyisipan yang dilakukan untuk menyisipkan hasil enkripsi data ke dalam channel alpha. Algoritma enkripsi yang digunakan adalah AES. Input dari proses penyisipan ini yaitu gambar hitam putih, yang secara umum mempunyai gambaran sebagai berikut:



Gambar 4.4 Metode Pengenkripsi Gambar

Metode pengenkripsi gambar memproses data gambar hitam putih menjadi chipertext. Algoritma AES (Rijndael) digunakan untuk mengenkripsi data yang akan disisipkan. Kemudian hasil enkripsi yang berupa chipertext disisipkan dalam channel alpha. Jika nilai chipertext adalah 1 maka dalam channel alpha akan diisi 255, sedangkan jika chipertext 0 maka channel alpha bernilai 254.



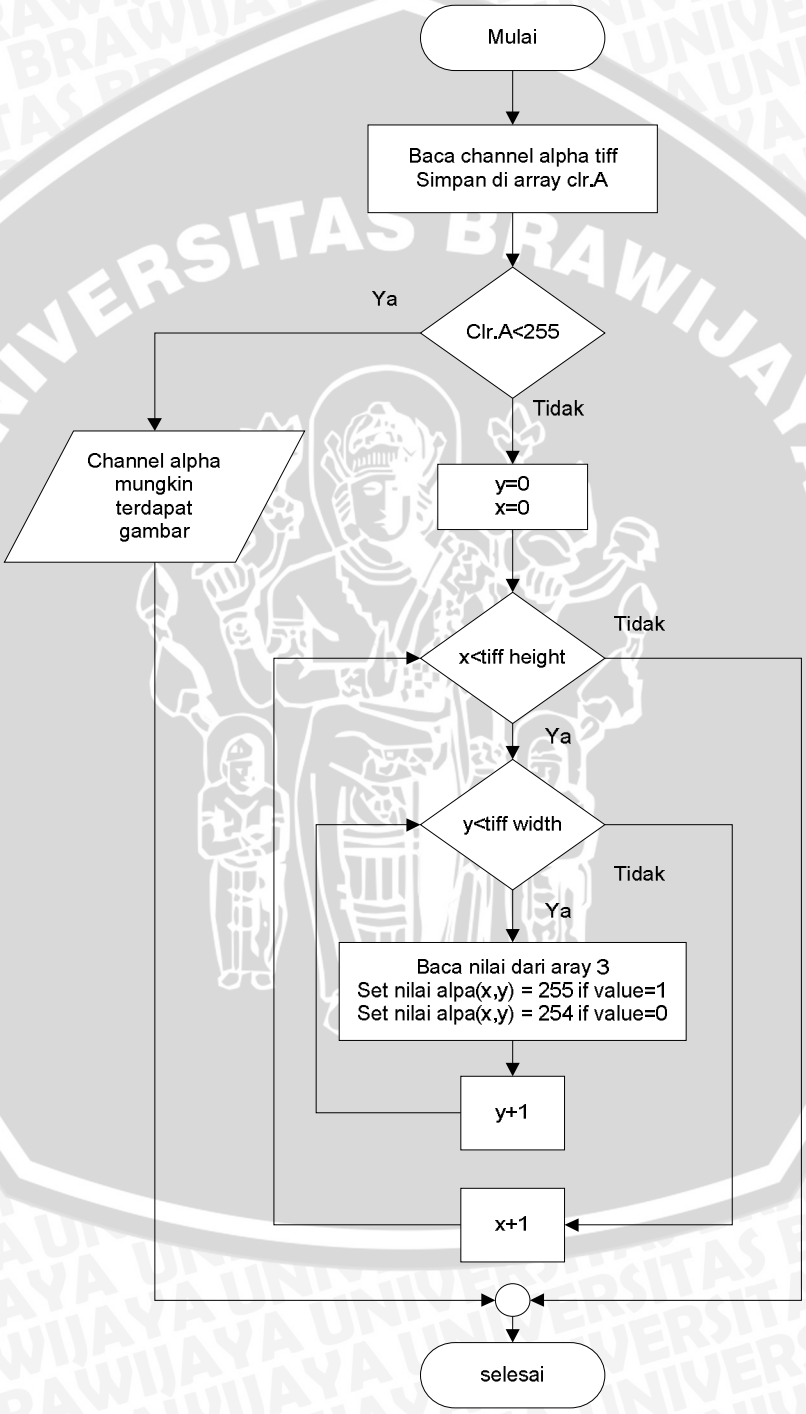
Flowchart Penyisipan Data Terenkripsi

Alur flowchart dapat dijelaskan sebagai berikut:

1. Program akan membaca gambar hitam putih (jpeg) kemudian memasukkan nilai dari piksel gambar per-*bit* ke dalam array 1.
2. Array 1 yang berisi informasi antara 0 atau 1 kemudian dikelompokkan menjadi grup yang tiap grup adalah 8-*bit* yang dapat digambarkan dengan format *string* dan dimasukkan ke array 2. Jadi semisal data di array 1 adalah 1,1,1,1,1,1,0,0,…dst maka nilai di array 2 adalah 11111100,…dst. Jika data tidak habis dibagi 8 maka data sisa hasil pembagian tersebut dijadikan satu walau tidak 8 bit.
3. Array 2 adalah array yang berisi string tadi kemudian nilai tiap array 2 dienkripsi dengan algoritma AES dengan kunci 128-*bit* dan nilainya dimasukkan ke array 3.
4. Kemudian array 3 yang berupa chipertext dipisahkan kembali per-bit nya dengan meng-convert StringToChar nilai array, sehingga didapat kembali nilainya.

4. Bit-bit yang ada dalam array inilah yang kemudian disisipkan ke dalam alpa channel.

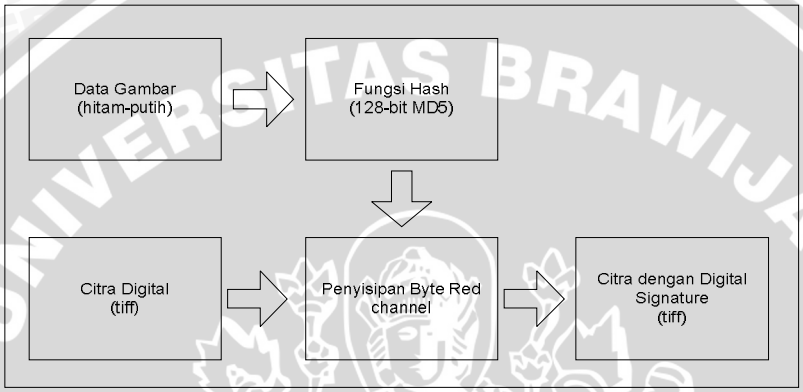
Kemudian untuk menyisipkan pada channel alpa menggunakan proses looping seperti flowchart di bawah ini. Proses looping menggunakan height dan width yang didapat dari gambar tiff.



Flowchart Sisip pada Channel Alpa

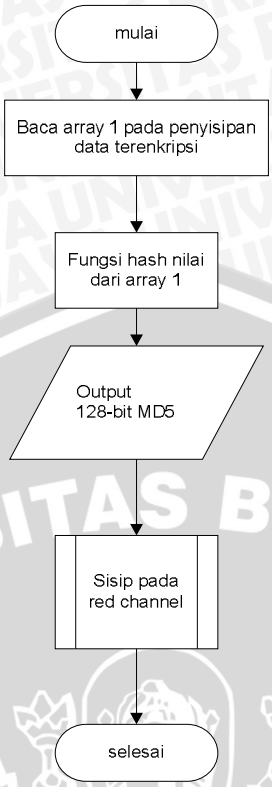
4.3.2 Penyisipan Digital Signature

Digital signature yang digunakan dalam skripsi ini adalah fungsi hash MD5. Fungsi hash dalam sistem ini digunakan untuk menentukan keaslian gambar dikirim pada citra penampung yang mempunyai masukan data plaintext sebelum enkripsi AES. Data yang dihasilkan berupa bilangan yang terdiri 16 byte atau 128 bit. Sehingga untuk menyisipkan data ini cukup disisipkan pada LSB channel red citra penampung, hanya sepanjang 128 bit pertama dari channel red.

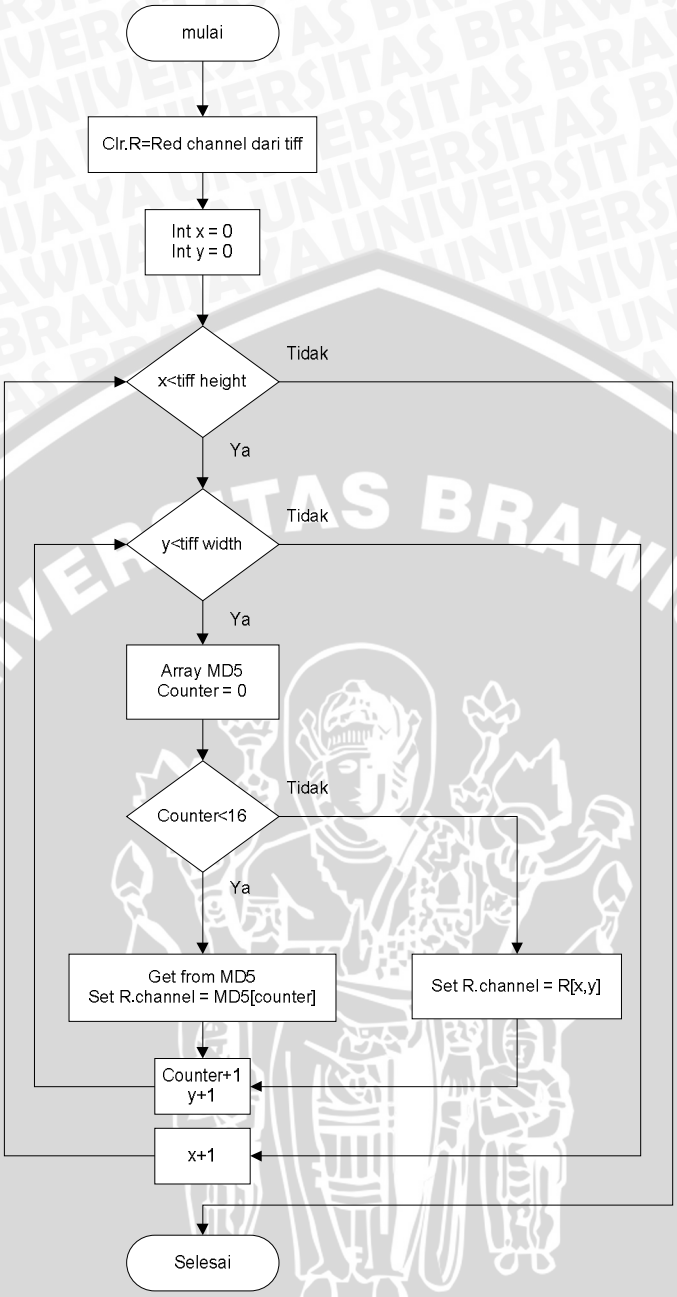


Gambar 4.6 Metode Digital Signature dengan Fungsi Hash

Pada proses digital signature ini, input dari proses adalah data gambar hitam putih yang kemudian dihashkan menjadi format MD5 sepanjang 128-bit. Input gambar mengambil data dari array 1 yang sebelumnya ada pada penyisipan pada data terenkripsi. Dalam proses ini yang diambil adalah data asli piksel yaitu pada array 1. Alur program dari penyisipan digital signature adalah sebagai berikut :



Flowchart Penyisipan Digital Signature



Flowchart Sisip pada Red Channel

Flowchart sisip pada red channel mempunyai penjelasan sebagai berikut :

1. Program menginisialisai tipe data Clr.R dan diisi dengan nilai piksel yang ada di red channel gambar tiff.
2. Program mengeset nilai yang ada di channel R dengan menggunakan looping sepanjang panjang x lebar gambar tiff.
3. Karena panjang data yang disisipkan hanya 128-bit (16-byte) saja maka untuk mempermudah menggunakan counter. Jika counter masih di bawah

16 maka data di red channel akan diganti sesuai dengan nilai yang bergantung pada nilai pada array MD5. Jika tidak maka nilai red channel sesuai dengan loopingnya yaitu pada nilai R[x,y].

BAB V

IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah didapatkan dari perancangan perangkat lunak yang telah dibuat. Implementasi merupakan proses transformasi hasil perancangan perangkat lunak ke dalam kode *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan. Pembahasan terdiri dari penjelasan tentang Lingkungan Implementasi, Algoritma Implementasi, dan Implementasi Antarmuka Aplikasi.

5.1.Lingkungan Implementasi

Sistem dibuat dengan menggunakan aplikasi pemrograman Microsoft Visual Studio 2010. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

5.1.1 Spesifikasi Perangkat Keras

- CPU : AMD Turion™ 64 RM-70 (2.0 GHz)
- Memory : 2 GB RAM
- Hard Disk : 160 GB
- Graphics Adapter : Nvidia GeForce 9100M G

5.1.2 Spesifikasi Perangkat Lunak

- Sistem operasi : Windows 7
- Aplikasi yang digunakan : Microsoft Visual Studio 2010

5.2 Algoritma Sistem

Penyajian yang digunakan berupa urutan baris algoritma seperti kode pemrograman dan parameter yang digunakan dalam menggunakan class yang telah disediakan. Semua proses yang dijelaskan dalam bentuk potongan listing program dan parameternya. Tahap Implementasi ini berdasarkan perancangan sistem pada tahap Perancangan.

5.2.1 Implementasi Metode Pengenkripsian Data

Sebelum data dienkripsi, data terlebih dahulu diproses untuk bisa dimasukkan ke dalam channel alpa dalam format terenkripsi. Format enkripsi yang dimaksud adalah data berupa piksel acak hitam putih yang nilainya diperoleh dari hasil enkripsi Rijndael.

Pertama-tama data piksel diperoleh dari gambar hitam-putih kemudian dimasukkan ke dalam array pix1.

```
pix1 = gcnew array<String^>(thres_bm->Width * thres_bm->Height);
        counter = 0;
        for (int x = 0; x <= thres_bm->Width - 1; x++)
        {
         for (int y = 0; y <= thres_bm->Height - 1; y++)
         {
                Color clr = thres_bm->GetPixel(x, y);
                if (clr.R == 0 | clr.R <= thresPilihan){
                        Threshold[x, y] = 0;
                        pix1[counter] = "0";
                }
                if (clr.R == 255 | clr.R > thresPilihan){
                        Threshold[x, y] = 255;
                        pix1[counter] = "1";
                }
                clr = Color::FromArgb(clr.A, Threshold[x, y], Threshold[x, y], Threshold[x, y]);
                        thres_bm->SetPixel(x, y, clr);
      counter++;
    }
   }
pix2 = gcnew array<String^>(pix1->Length/8);
pix3 = gcnew array<String^>(pix2->Length);
counter = 0;
        for (int g = 0; g < pix2->Length; g++)
   {for (int n = 0;n<8;n++)
     {pix2[g] = pix2[g] + pix1[counter];
                counter++;
          }
    pix3[g] = Encryption::BinaryToDec(pix2[g]->ToString());
         pix4[g] = Convert::ToByte(pix3[g]);
   }
pix5 = gcnew array<Byte>(pix2->Length);
pix5 = Encryption::EncryptFile(pix4,"pass");
pix6 = gcnew array<String^>(pix2->Length);
```

Kode Implementasi memperoleh nilai piksel gambar yang akan dienkripsi

Data pada pix1 berupa antara 0 atau 1. Data 2 kemudian dikelompokkan tiap 8 bit agar dapat dimasukkan ke proses enkripsi yaitu perbyte.

Kemudian pix5 diisi dengan nilai byte yang telah dienkripsi AES, seperti pada listing berikut :

Kemudian data dibalik lagi agar menjadi bit acak dengan cara mengubah byte hasil enkripsi menjadi bit lagi.

```
for (int b = 0;b < pix6->Length;b++){
                    pix6[b] = Encryption::DecToBinary(pix5[b].ToString());
                    if(pix6[b]->Length < 8){
                              pix6[b] = gcnew String('0',8 – pix6[b]->Length) + pix6[b];
                    }
    }
pix8 = gcnew array<Byte>(pix5->Length * 8);
pix9 = gcnew array<Byte>(pix5->Length * 8);
counter = 0;
for (int b = 0;b < pix6->Length;b++){
          array<Char> ^pix7 = gcnew array<Char>(8);
          pix7 = pix6[b]->ToCharArray();
          for(int v=0;v<pix7->Length;v++){
                    pix8[counter] = Convert::ToByte( pix7[v].ToString());
                    counter++;
          }
}
for(int z = 0;z < pix8->Length;z++){
                    if(pix8[z] == 0) pix9[z] = 254;
                    else pix9[z] = 255;
          }
```

Kode Implementasi Proses Pengenkripsian Data dengan menggunakan Array

```
counter = 0;
for (int x = 0; x < thres_bm->Width; x++)
{
          for (int y = 0; y < thres_bm->Height; y++)
          {Color Cx = thres_bm->GetPixel(x, y);
           Cx = Color::FromArgb(pix9[counter], Cx.R, Cx.G, Cx.B);
     thres_bm->SetPixel(x, y, Cx);
           counter++;
          }
}

picEnkripsi->Image = thres_bm;
```

Kode Implementasi Penyisipan bit yang sudah terenkripsi ke dalam channel Alpa

5.2.2 Implementasi Algoritma Enkripsi Data

Class Rijndael

Inisialisasi class Rijndael pada net framework 3.5 sudah ada. Jadi metode AES sudah dibuatkan oleh pihak developer dengan menggunakan manual source kodenya. Kita tinggal memanggil metodenya dengan inisialisasi awal untuk deklarasi, key dan inisial vektornya.

```
Rijndael ^rij = Rijndael::Create();
```

PasswordDeriveBytes ^pwdBytes = gcnew PasswordDeriveBytes(pass, gcnew array<Byte>{});

Algoritma Enkripsi yang digunakan adalah sebatas menggunakan class yang telah disediakan oleh library dari bahasa pemrogramannya dalam hal ini visual studio. Algoritma Rijndael yang digunakan merupakan algoritma System Security Cryptography yang terdapat pada Microsoft visual studio dengan inisialisasi sebagai berikut :

```
using namespace System::Security::Cryptography;

namespace AlphaChannel
{
public ref class Encryption
        {
public: static array<Byte> ^EncryptFile(array<Byte> ^inputData, String ^pass)
        {
                PasswordDeriveBytes ^pwdBytes = gcnew PasswordDeriveBytes(pass, gcnew
array<Byte> {});
                MemoryStream ^stream = gcnew MemoryStream();

                Rijndael ^rij = Rijndael::Create();
                rij->Key = pwdBytes->GetBytes(16);
                rij->IV = pwdBytes->GetBytes(16);
                CryptoStream ^cStream = gcnew CryptoStream(stream, rij->CreateEncryptor(),
CryptoStreamMode::Write);

                cStream->Write(inputData, 0, inputData->Length);
                cStream->Close();
                array<Byte> ^encryptedData = stream->ToArray();
                array<Byte> ^xz = gcnew array<Byte>(encryptedData->Length);
                for (int g = 0; g < xz->Length; g++)
                {
                        xz[g] = encryptedData[g];
                }
                return xz;
        }

public: static array<Byte> ^DecryptFile(array<Byte> ^outputData, String ^pass)
        {
                PasswordDeriveBytes ^pwdBytes = gcnew PasswordDeriveBytes(pass, gcnew
array<Byte> { });
                MemoryStream ^stream = gcnew MemoryStream(outputData);
                Rijndael ^rij = Rijndael::Create();
                rij->Key = pwdBytes->GetBytes(16);
                rij->IV = pwdBytes->GetBytes(16);
                CryptoStream ^cStream = gcnew CryptoStream(stream, rij->CreateDecryptor(),
CryptoStreamMode::Write);
                cStream->Write(outputData, 0, outputData->Length);
                array<Byte> ^decryptedData = stream->ToArray();
                array<Byte> ^xz = gcnew array<Byte>(decryptedData->Length - 16);
                for (int g = 0; g < xz->Length; g++)
                {
                        xz[g] = decryptedData[g];
                }
                return xz;
        }
```

Kode Implementasi Fungsi / *method* Algoritma AES pada header encryption.h

5.2.3 Implementasi Algoritma Digital Signature

Untuk memberikan gambar nilai dari md5 nya dengan menggunakan ^GiveMD5.

```
public: static array<Byte> ^MD5SUM(array<Byte> ^FileOrText)
        {
                MD5CryptoServiceProvider ^md5sum = gcnew MD5CryptoServiceProvider();
                return md5sum->ComputeHash(FileOrText);
        }
public: static void HashMD5(Image ^I)
                        {
                                Bitmap ^B = gcnew Bitmap(I);
                                int x;
                                int y;
                                int loop = 0;
                                array<Byte> ^by = gcnew array<Byte>(B->Width * B->Height *3);
                                for (x = 0; x < B->Width; x++)
                                {
                                        for (y = 0; y < B->Height; y++)
                                        {
                                                Color C = B->GetPixel(x, y);
                                                by[loop] = C.R;
                                                by[loop + 1] = C.G;
                                                by[loop + 2] = C.B;
                                                loop = loop + 1;
                                        }
                                }
                                loop = 0;
                                globalGoesHere::md5Byte = MD5SUM(by);
                        }

public: static Bitmap ^GiveMD5(Bitmap^ B)
{
        int loop = 0;
        for (int x = 0; x < B->Width; x++)
        {
                for (int y = 0; y < B->Height; y++)
                {
                        Color Cx = B->GetPixel(x, y);
                        if (loop < 16){
                        Cx = Color::FromArgb(Cx.A, md5Byte[loop], Cx.G, Cx.B);

                                loop = loop + 1;
                        }
                        else
                        {
                                Cx = Color::FromArgb(Cx.A, Cx.R, Cx.G, Cx.B);
                        }
                        B->SetPixel(x, y, Cx);
                }
        }
        return B;
}
```

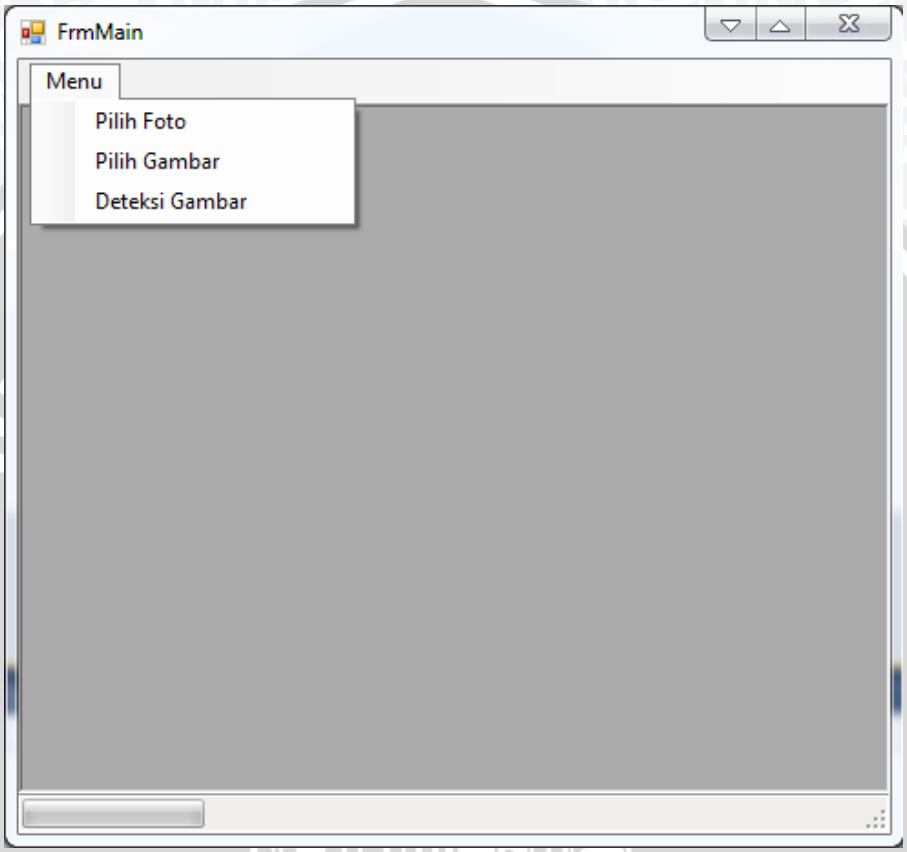
Kode Implementasi Fungsi Hash ke dalam gambar

5.4.1 Implementasi Antarmuka Program Penyisipan Gambar

Program penyisipan gambar mempunyai tampilan sebagai berikut.

5.4.1.1 Form Menu utama

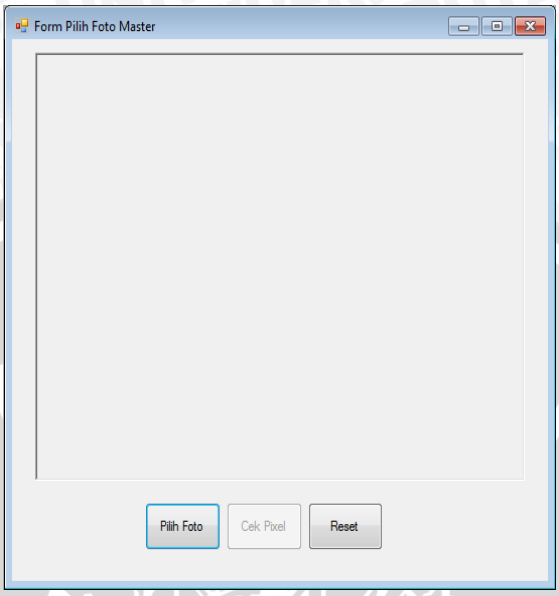
Tampilan paling awal ini digunakan untuk memilih apakah untuk mendeteksi gambar yang ada pada gambar tiff. Jika ingin menyisipkan gambar maka harus memilih ketiga menu yang ada yaitu pilih foto, pilih gambar yang disisipkan, dan sisipkan gambar.



Gambar 5.1 Tampilan menu utama program penyisipan pesan

5.4.1.2 Form Pilih Foto

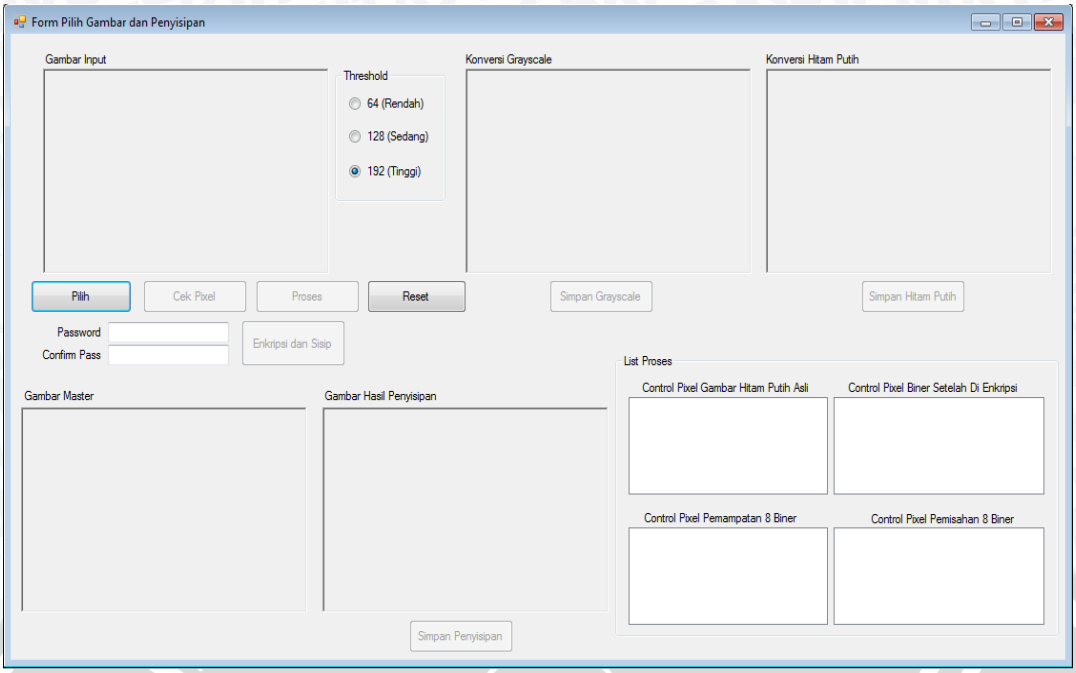
Form pilih foto digunakan untuk memilih gambar yang akan ditumpangi. Gambar yang dipilih harus berformat .tiff.



Gambar 5.2 Tampilan Pilih Foto

5.4.1.3 Form Pilih Gambar yang disisipkan

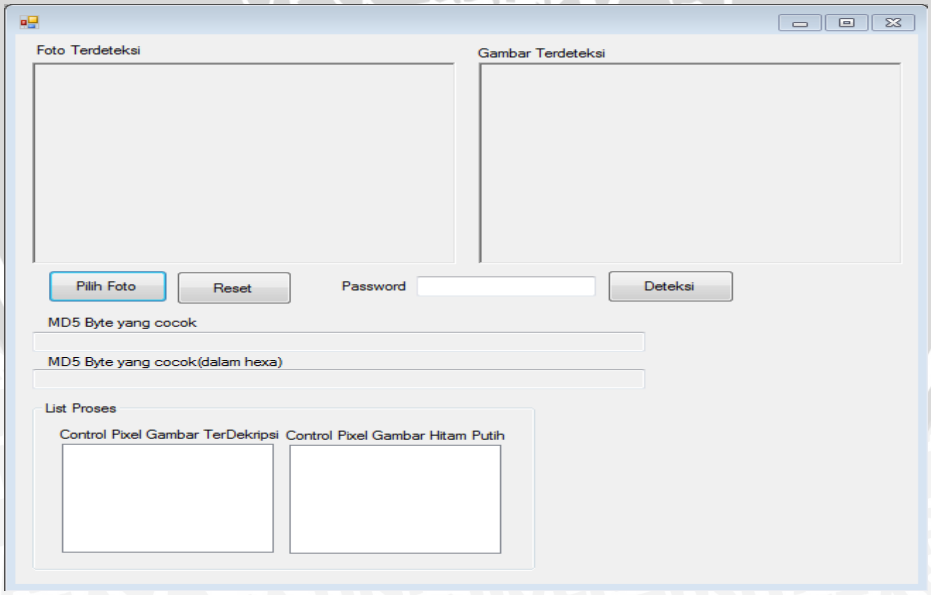
Form pilih gambar hitam putih mempunyai opsi treshold yaitu jika user memasukkan gambar yang berwarna maka program akan otomatis membuatnya hitam putih dengan menggunakan treshold.



Gambar 5.3 Pilih Gambar hitam putih

5.4.1.4 Form Deteksi Gambar

Form ini berfungsi mendeteksi gambar yang sudah dibuat oleh program ini. Untuk memeriksa keaslian data yang ada maka MD5 byte akan ditampilkan. Jika tidak cocok maka akan segera akan terlacak.



Gambar 5.5 Deteksi Gambar

BAB VI
PENGUJIAN DAN ANALISIS

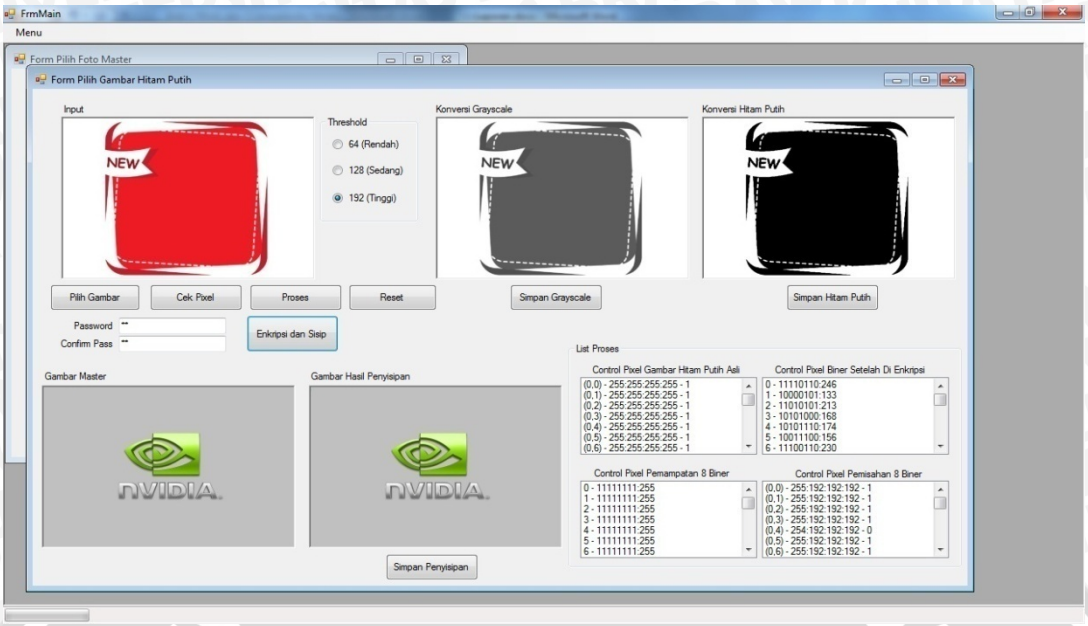
Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan dalam bab ini adalah sebagai berikut:

1. Pengujian program penyisipan
2. Pengujian kualitas citra penyisipan
    a. Pengukuran secara subjektif dilakukan dengan mengamati gambar hasil penyisipan
    b. Pengukuran secara objektif dilakukan dengan menghitung nilai *PSNR*.
3. Pengujian manipulasi citra penampung

6.1 Pengujian Program Penyisipan

Pengujian dilakukan dengan menyisipkan citra gambar sesuai dengan alur program. Proses penyisipan dimulai dengan memilih citra tiff sebagai citra penampung.

Citra gambar tiff dipilih terlebih dahulu sebagai citra penampung untuk gambar hitam putih, kemudian memilih citra gambar hitam putih untuk dienkripsi dengan password. Sebelum dienkripsi, citra gambar yang disisipkan akan dirubah ke gambar hitam putih dengan menggunakan grayscaling kemudian tresholding yang dapat kita atur sendiri yaitu antara 64, 128 dan 192. Data piksel hitam putih inilah yang akan dienkripsi dengan algoritma AES. Hasil enkripsi berupa ciphertext kemudian dijadikan bilangan biner seperti terlihat pada listbox. Deretan biner tersebut kemudian disisipkan pada channel alpa, sedangkan hash dari gambar disisipkan pada channel red citra tiff.

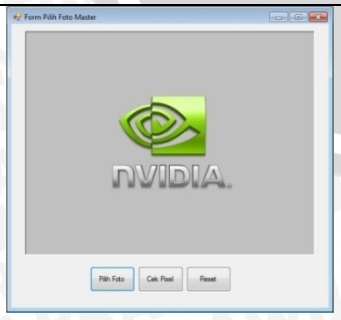
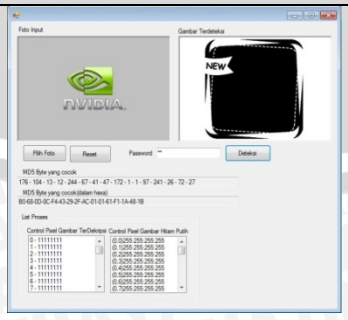
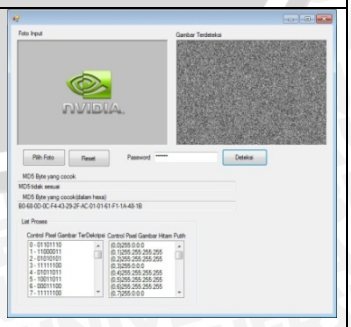


Gambar 6.1.1 Hasil Penyisipan pada citra gambar

List proses pada control piksel adalah tampilan array citra awal yang berformat RGB, array pengelompokkan data bit hitam putih sebelum enkripsi dan sesudah enkripsi.

6.1.1 Deteksi Citra Gambar yang disisipkan

Untuk pendeteksian citra yang disisipkan menggunakan program dekripsi yang telah dibuat dengan metode kebalikan dari enkripsinya. Di sini faktor keamanan terletak pada kunci AES. Jika passwordnya benar maka citra dekripsinya akan sesuai dengan sebelum enkripsi, apabila passwordnya salah maka citra tidak dapat dideteksi.

Citra Gambar yang Dideteksi	Password Benar	Password Salah
		

Tabel 6.1.1 Hasil Pengujian Deteksi

6.2 Pengujian Kualitas Citra Penyisipan

Pengujian ini dilakukan untuk menguji kualitas dari citra hasil penyisipan, yaitu dengan membandingkan citra setelah disisipi dengan citra yang asli. Untuk pengujian kualitas dari citra yang ditumpangi, dilakukan dua penilaian yaitu subjektif dan objektif.

1. Untuk pengujian subjektif dilakukan dengan memberikan citra hasil penyisipan dan citra yang asli kepada responden sebagai sampel untuk dianalisa terlihat adanya perbedaan atau tidak antara citra tersebut.
2. Untuk pengujian objektif dilakukan dengan menghitung nilai Peak Signal to Noise Ratio (PSNR) antara citra hasil penyisipan dan citra asli.

Pengujian akan berhasil apabila dari masing-masing cara, didapatkan hasil seperti berikut:

1. Pengukuran secara subjektif antara citra hasil penyisipan dan citra yang asli dianggap mirip.
2. Pengukuran secara objektif yaitu dengan mengukur nilai *Peak Signal to Noise Ratio* (PSNR) yang didapatkan berada diatas 20 dB.

6.2.1 Hasil Pengujian

6.2.1.1 Prosedur Pengujian Subjektif

Pengujian ini dilakukan pada suatu gambar tiff kemudian disisipi dengan gambar dengan tipe fotografis dan non-fotografis untuk mengetahui bagaimana pengaruh gambar yang disisipkan akan mempengaruhi gambar citra penampung.

6.2.1.2 Pengujian Subjektif

Prosedur pengujian subjektif dilakukan dengan mengamati gambar hasil penyisipan dengan citra asli apakah perbedaan piksel terlihat oleh responden atau tidak.

Hasil pengujian perangkat lunak untuk pengujian subjektif dapat dilihat dalam tabel 6.1

No.	Gambar Sebelum Disisipi	Gambar Setelah disisipkan	Perbedaan Citra Asli dan Termodifikasi		
			Responden I	Responden II	Responden III
1.	Nojiri_village	Tulips	Tak terlihat	Tak terlihat	Tak terlihat
2.	Nojiri_village	Img24	Tak terlihat	Tak terlihat	Tak terlihat
3.	Nojiri_village	sports-car-vector	Tak terlihat	Tak terlihat	Tak terlihat
4.	Nojiri_village	vector_art_flyer	Tak terlihat	Tak terlihat	Tak terlihat
5.	Nojiri_village	banner-vector	Tak terlihat	Tak terlihat	Tak terlihat

Tabel 6.1 Hasil Pengujian terhadap Responden

6.2.1.3 Pengujian Objektif

Prosedur pengujian yaitu dengan membandingkan citra sebelum disisipi dengan citra setelah disisipi gambar hitam putih. Hasil pengujian kinerja perangkat lunak untuk pengujian objektif dapat dilihat dalam tabel 6.2

No.	File Citra Sebelum Disisipi	File Citra Setelah Disisipi	Nilai PSNR (dB)
1.	Nojiri_village	tulips	26,992
2.	Nojiri_village	Img24	28,047
3.	Nojiri_village	sports-car-vector	27,299
4.	Nojiri_village	vector_art_flyer	27,125
5.	Nojiri_village	banner-vector	27,658

Tabel 6.2 Hasil Pengujian terhadap nilai PSNR

6.3 Pengujian Manipulasi Citra Penampung

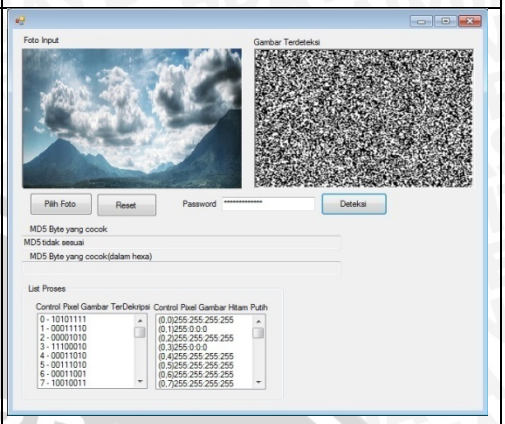
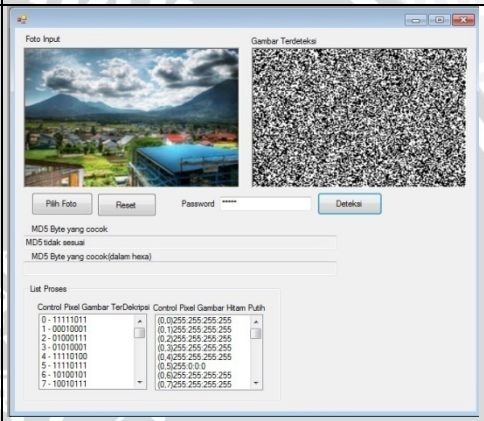
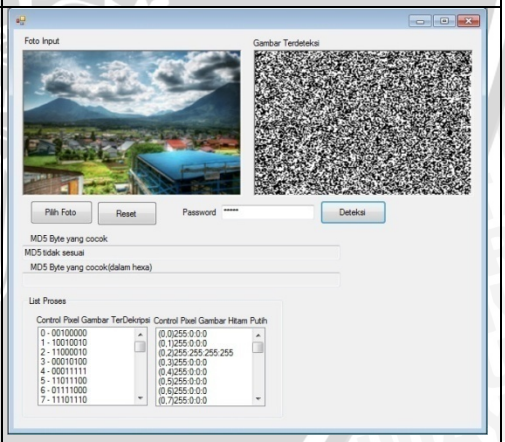
Pengujian ini dilakukan untuk mengetahui sejauh mana data di dalam gambar berubah atau tidak ketika dilakukan manipulasi terhadap gambar penampung. Manipulasi yang dilakukan antara lain adalah :

1. Melakukan *cropping* pada gambar.
2. Melakukan *resize* resolusi gambar (Resolusi gambar awal 800x600).

6.3.1 Hasil Pengujian Manipulasi Citra Penampung

Prosedur Pengujian

1. Gambar hasil aplikasi diuji dengan aplikasi pengolah gambar Photosop CS3 kemudian gambar dipotong sehingga menjadi setengah dari ukuran semula
2. Gambar awal hasil aplikasi diuji lagi dengan memperkecil/memperbesar resolusi gambar menjadi setengahnya/2x ukuran semula

Pengujian	Keterangan	Hasil
Pengujian cropping menjadi setengah (gambar dipotong menjadi 2 bagian simetris)	Gagal dideteksi	
Pengujian resize resolusi menjadi 0,5 ukuran semula (menjadi 400x600)	Gagal dideteksi	
Pengujian resize resolusi menjadi 2 ukuran semula (menjadi 1600x1200)	Gagal dideteksi	

Tabel 6.3 Hasil pengujian manipulasi citra penampung

6.4 Analisis Faktor Kegagalan

Sistem dapat berjalan dengan baik apabila data gambar tiff (citra penampung) tidak dirubah. Program mengalami kegagalan dengan kemungkinan sebagai berikut :

1. Jika data piksel red channel dirubah maka fungsi hash yang terdapat didalamnya akan berubah sehingga program menghasilkan *output* yang tidak sesuai dengan yang diharapkan.
2. Jika citra penampung dirubah (resize dan cropping) maka channel alpa akan berubah. Hal ini mengakibatkan program akan mendapatkan data yang salah untuk dekripsi dari channel alpa sehingga program mendeteksi gambar sebagai gambar yang acak.

BAB VII

KESIMPULAN DAN SARAN

Berdasarkan hasil-hasil yang dicapai selama perancangan, pembuatan dan pengujian dari penelitian ini, maka dapat diambil beberapa kesimpulan dan saran.

7.1 Kesimpulan

Dari hasil perancangan dan pengujian yang telah dilakukan dapat disimpulkan beberapa hal berikut:

1. Penyisipan citra hitam putih ke dalam citra *tiff* berhasil dilakukan
2. Gambar yang disisipkan dibatasi hanya sebesar resolusi gambar itu sendiri karena dikhawatirkan apabila terlalu besar gambar yang disisipkan akan merubah struktur dari citra yang ditumpangi.
3. Nilai rata-rata PSNR yang dihasilkan program berkisar antara 28,3 dB. Jadi dapat dikatakan bahwa hasil tersebut masih baik karena nilai PSNR untuk perbandingan gambar berada di batas minimum 20 dB.
4. Teknik ini tidak menghasilkan perbedaan yang berarti antara citra asli dan setelah penyisipan.
5. Teknik ini merubah ukuran citra penampung ketika menyisipkan gambar hitam putih.
6. Citra hasil penyisipan tidak tahan terhadap manipulasi yang dilakukan pada citra penampung.

7.2 Saran

Hasil penelitian ini masih terdapat kekurangan yang masih bisa diperbaiki pada penelitian selanjutnya di tahun yang akan datang. Adapun beberapa kekurangan yang perlu diperbaiki pada penelitian yang akan datang adalah:

1. Diharapkan stega-cryptografi melalui aplikasi ini dapat diterapkan pada berbagai jenis citra lainnya antara lain yang mempunyai channel alpa seperti : PNG dan GIF.
2. Diharapkan gambar yang disisipkan bisa berupa citra berwarna dan juga teks.

DAFTAR PUSTAKA

Ahmad, Usman. 2005. *Pengolahan Citra Digital*. Graha Ilmu. Yogyakarta. http://www.cert.or.id/~budi/courses/ec5010/projects/wihartantyo-report.doc, 02 Februari 2011.

Cummins, Jonathan., Diskin , Patrick ., Lau, Samuel., and Parlett, Robert. 2004. *Steganography And Digital Watermarking, School of Computer Science*. The University of Birmingham.

Jafilun, 2006, "Digital Watermaking pada Domain Spasial Menggunakan Teknik Least Significant Bit". *http://yudiagusta.files.wordpress.com/2009/11/47-53-snsi06-08-digital-watermarking-pada-domain-spasial-menggunakan-teknik-least-significant-bit.pdf*, diakses tanggal 07 Juli 2010

Supangkat H Suhono, Kuspriyanto,Juanda. 2000. *Watermarking sebagai Teknik Penyembunyian Label Hak Cipta Pada Data Digital*. Departemen Teknik Elektro Insitut Teknologi Bandung.

http://en.wikipedia.org/wiki/RGB_color_model. Tanggal akses: 1 Desember 2010.

http://en.wikipedia.org/wiki/Alpha_compositing. Tanggal akses: 1 Desember 2010.

http://id.wikipedia.org/wiki/Steganografi. Tanggal akses: 26 Desember 2010.

http://www.infosyssec.com/infosyssec/Steganography/techniques.htm. Tanggal akses: 26 Desember 2010

http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf. Tanggal akses 1 Desember 2010

http://www.ietf.org/rfc/rfc1321.txt. Tanggal akses: 1 Desember 2010

Sams. 1998. "Programming - Teach Yourself Visual C++ In 21 Days". *www.angelfire.com/art2/ebooks/teachyourselfvisualcplusplusin21days.pdf*. Diakses tanggal 7 Juli 2010

Jafilun, 2006, "Digital Watermaking pada Domain Spasial Menggunakan Teknik Least Significant Bit". *http://yudiagusta.files.wordpress.com/2009/11/47-53-snsi06-08-digital-watermarking-pada-domain-spasial-menggunakan-teknik-least-significant-bit.pdf*. Diakses tanggal 07 Juli 2010 pukul 11.00

Jardiknas, JENI, 2007,"Pengenalan Pemrograman 1", *http://space.meruvian.org/jeni*, diakses tanggal 03 Maret 2010 pukul 18.00.

Sosra, 2009, "Pengenalan Steganografi",*http://sosrapolice.blogspot.com/2009/06/pengenalan-steganography.html*, diakses tanggal 01-Juli-2011 pukul 19.10.

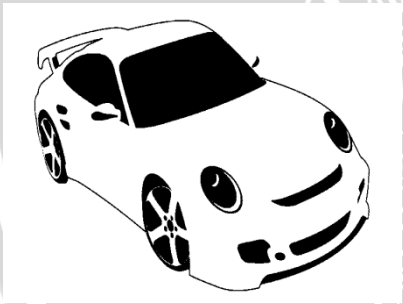
LAMPIRAN

Lampiran 1 Citra Asli

- Nojiri_village(800x600).tiff

Lampiran 2 Citra yang disisipkan

1. Car.jpg



2. Banner.jpg

3. Img24.jpg

4. Tulips.jpg

5. Art.jpg

Lampiran 3 Citra Pengujian Manipulasi Citra Penampung

1. Citra_crop.tiff

2. Resize05.tiff

3. Resize2x.tif