IMPLEMENTASI ALGORITMA TWOFISH DAN LEAST SIGNIFICANT BIT (LSB) UNTUK PENYEMBUNYIAN FILE TEXT PADA CITRA DIGITAL

Devri Riza Setyawan[1]
devririzasetyawan@gmail.com

Willy Sudiarto Raharjo[2]
willysr@ti.ukdw.ac.id

R. Gunawan Santosa[3]
gunawan@ukdw.ac.id

Abstract

Security and confidentiality of information are considered a requirement in today's communication media. Many people have used Internet to share and distribute confidential information. On the other hand, many malicious activities attempts to steal those information by irresponsible parties. Therefore, information must be secured and converted into another form before it is sent to the destination by the use of cryptography and steganography. We developed a program to secure this information, by using Twofish and Least Significant Bit algorithm to encrypt secret messages and then insert it into an image file. By combining these two methods, we can achieve higher level of information security.

The information encoded with twofish algorithm will change the file size slightly. This changes are caused by the use of padding. In addition, both of the output image are not resistant by some manipulation operations. The output of the system in BMP format does not change the file size after insertion process, while PNG format will have a slightly difference in file size due to processing in bitmap mode.

Keywords: *twofish, block cipher, LSB*

1. Pendahuluan

Seiring berkembangnya teknologi komunikasi, cara berkomunikasi manusia sudah beralih menggunakan media Internet sebagai sarana komunikasi. Dengan Internet, komunikasi dapat dilakukan dengan lebih mudah, cepat dan dapat dilakukan dimana saja. Akan tetapi, komunikasi melalui Internet mempunyai kelemahan yaitu penyadapan atau pencurian informasi. Aspek keamanan informasi dalam komunikasi sangat penting untuk di perhatikan.Untuk hal itu dibutuhkan metode untuk mengamankan informasi tersebut.

Kriptografi dan Steganografi merupakan metode pengamanan data untuk menjaga kerahasiaan dan keaslian data serta dapat meningkatkan aspek keamanan suatu informasi. Metode ini bertujuan agar informasi yang dikirimkan melalui Internet tidak dapat disadap oleh pihak-pihak yang tidak berkepentingan pada informasi tersebut.

Oleh Karena itu perlu dibangun sebuah sistem yang memadukan algoritma kriptografi *twofish* dan steganografi *least significant bit (lsb)* yang dapat menyandikan informasi lalu menyisipkannya ke dalam sebuah file citra. Dengan menggunakan perpaduan tersebut, tingkat keamanan informasi akan menjadi lebih tinggi.

---

[1] Mahasiswa Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana

[2] Dosen Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana

[3]Dosen Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Kristen Duta Wacana

2. Teori Pendukung

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa yunani : "*cryptos*" yang artinya "*secret*" (rahasia), dan "*graphein*" yang artinya "*writing*" (tulisan), jadi kriptografi berarti "*secret writing*" (tulisan rahasia). Kriptografi adalah ilmu dan seni untuk menjaga kemanan dan kerahasiaan pesan dengan cara menyandikan ke dalam bentuk yang tidak dapat dimengerti lagi maknanya (Munir, 2006). Kriptografi memiliki beberapa tujuan yaitu Kerahasian (*secrecy*), Integritas data (*data integrity*), Otentikasi (*authentication*), dan Nirpenyangkalan (non-*repudiation*). Enkripsi adalah proses mengubah pesan biasa (*plaintext*) menjadi menjadi kode-kode yang tidak bisa dimengerti (*ciphertext*). Sedangkan dekripsi adalah mengembalikan pesan rahasia yang telah terenkripsi kebentuk asalnya (Dony, 2005).

2.2 Algoritma Kriptografi Simetris

Algoritma simetris (*symmetric algorithm*) adalah suatu algoritma dimana kunci enkripsi yang digunakan sama dengan kunci dekripsi, sehingga algoritma ini disebut juga sebagai *single-key algoritm*. Algoritma ini disebut juga algoritma kunci rahasia (*secret-key algorithm*), karena kunci yang digunakan harus terjaga kerahasiaannya dan hanya boleh diketahui oleh pengirim dan penerima pesan saja.

2.3 Algoritma *Block Cipher*

Block cipher merupakan suatu algoritma kriptografi yang beroperasi dalam bentuk blok bit dimana mana input dan outputnya berupa satu blok, dan setiap blok terdiri dari beberapa bit (1 blok dapat terdiri dari 32 bit, 64 bit atau 128 bit). (Dony, 2005: 58). Input plaintext biasanya dibagi menjadi beberapa blok, misalnya 32 bit untuk setiap bloknya, jika input kurang dari jumlah tersebut maka akan dilakukan penambahan bit atau yang sering disebut padding, sehingga blok tersebut menjadi 32 bit. Proses enkripsi dilakukan dalam blok bit plaintext menggunakan kunci yang berukuran sama dengan ukuran blok plaintext dan menghasilkan ciphertext yang sama dengan blok plaintext.

2.4 Twofish

Twofish merupakan algoritma block cipher berbasis 128-bit yang dirancang oleh Bruce Schneier dan dan "*Counterpane Labs USA*" pada tahun 1997 kemudian dipublikasikan pada tahun 1998 (Schneier et al, 1999). Algoritma Twofish merupakan salah satu algoritma yang direkomendasikan sebagai AES. Hal ini disebabkan pemenuhan kriteria desain oleh NIST sebagai standar AES yaitu :

Merupakan blok cipher simetris 128-bit.

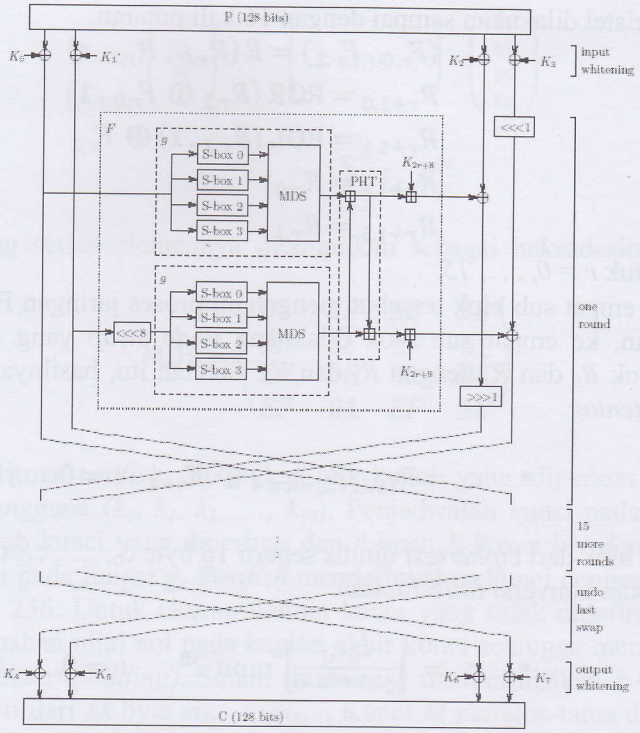
Panjang kunci yang digunakan adalah 128 bit, 192 bit, dan 256 bit.

Memiliki efisiensi pada software dan hardware dari platform yang berbeda.

Memiliki rancangan yang fleksibel, misalnya menerima panjang kunci tambahan, dapat diterapkan pada software dan hardware dari platform berbeda, cocok untuk stream chipper, fungsi hash dan MAC.

Desain yang simpel, memudahkan baik untuk analisa maupun implementasi.

Twofish menggunakan sebuah struktur jaringan Feistel 16 putaran dengan tambahan *input whitening* dan *output whitening*. Satu-satunya unsur yang bukan jaringan Feistel adalah rotasi 1 bit. *Twofish* memanfaatkan teknik manipulasi bit, kotat permutasi (*S-Box*), jaringan Feistel dan pemutaran ulang dengan pergiliran kunci sebanyak 16 kali putaran, transformasi *Pseudo-Hadamard,* penjadwalan kunci dan matrik MDS. Struktur dari algoritma *Twofish* digambarkan pada Gambar 1.



Gambar 1. Struktur Algoritma *Twofish*

Dikutip dari : Schneier, B., et al. (1999). *Twofish : A 128-Bit Block Cipher,* halaman 6.

Langkah-langkah dari algoritma twofish adalah sebagai berikut :

Plaintext sebesar 128 bit dipecah menjadi 4 blok 32 bit dengan konversi little endian.

$$P_i = \sum_{j=0}^{3} p_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 3 \qquad [1]$$

Hasil dari konversi little endian (P_0, P_1, P_2, dan P_3) akan melalui proses *input whitening* dengan meng-XOR-kan sub kunci K_0, K_1, K_2, dan K_3. (K_i adalah sub kunci ke-i yang didapatkan dari penjadwalan kunci).

$$R_{0,i} = P_i \oplus K_i \qquad i = 0, \ldots, 3 \qquad [2]$$

Hasil dari proses *input whitening*, 2 sub blok bagian kiri (R_0 dan R_1) dijadikan menjadi input dari fungsi f (salah satu blok (R_1) tersebut dirotasikan terlebih dahulu sejauh 8 bit (ROL_8). Fungsi f terdiri dari beberapa bagian yaitu fungsi g, transformasi *Pseudo-Hadamard* (PHT), dan penambahan dengan sub kunci yang diperluas.

Fungsi g terdiri dari 4 buah kotak-S dan matriks MDS. Input fungsi f (blok R_0 dan R_1) dimasukkan ke dalam fungsi g. Input tersebut diproses dengan kotak-S terlebih dahulu lalu hasilnya dilakukan pencampuran linear menggunakan matriks MDS.

Hasil dari fungsi g dimasukkan ke dalam fungsi transformasi *Pseudo-Hadamard* (PHT), kemudian hasilnya ditambahkan dengan 2 buah 32 bit sub kunci dari kunci yang diperluas (K_{2r+8} dan K_{2r+9}).

Kedua buah hasil dari fungsi f tersebut di XOR-kan dengan 2 sub blok bagian kanan (R_0 dan R_1). Salah hasil fungsi $f(R_0)$ di XOR-kan dengan satu blok bagian kanan (R_2) lalu hasilnya dirotasi ke kanan sejauh 1 bit (ROR_1). Blok R_3 pada bagian kanan dirotasi sejauh 1 bit (ROL_1), kemudian hasilnya di XOR-kan dengan blok hasil lainnya (R_1) dari fungsi $f(R_1)$.

2 buah blok 32 bit sebelah kiri dan kanan saling dipertukarkan (R_0 dan R_1 dipertukarkan dengan R_2 dan R_3).

- Jaringan Feistel dilakukan sampai dengan 16 kali putaran.

$$(F_{r,0}, F_{r,1}) = \mathrm{F}(R_{r,0}, R_{r,1}, r) \quad [3]$$

$$R_{r+1,0} = \mathrm{ROR}(R_{r,2} \oplus F_{r,0}, 1) \quad [4]$$

$$R_{r+1,1} = \mathrm{ROL}(R_{r,3}, 1) \oplus F_{r,1} \quad [5]$$

$$R_{r+1,2} = R_{r,0} \quad [6]$$

$$R_{r+1,3} = R_{r,1} \quad [7]$$

  - untuk $r = 0, \ldots, 15$.
- Setelah ke empat sub blok tersebut mengalami proses jaringan Feistel sebanyak 16 kali putaran, ke empat sub blok dilakukan *undo swap* yang mana menukarkan kembali blok R_0 dan R_1 dengan R_2 dan R_3. Setelah itu, hasilnya di lakukan proses output *whitening*.
-

$$c_i = R_{16,(i+2) \bmod 4} \oplus K_{i+4} \quad i = 0, \ldots, 3 \quad [8]$$

- Empat sub blok dari *ciphertext* ditulis seperti 16 byte $c_0, \ldots, c_{15}$ dengan menggunakan konversi *little-Endian.*
-

$$c_i = \left\lfloor \frac{c_{[i/4]}}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8 \quad i = 0, \ldots, 15 \quad [9]$$

Fungsi *f* adalah suatu permutasi yang bergantung pada kunci (*key-dependent*) dengan nilai 64 bit. Fungsi ini menerima 3 argumen yaitu, 2 buah input 32 bit R_0 dan R_1 dan nomor putaran yang juga digunakan untuk menentukan sub kunci mana yang akan dipakai pada putaran tersebut. R_0 dimasukkan ke dalam fungsi *g* dan akan menghasilkan T_0, dan R_1 akan digeser 8 bit ke kanan terlebih dahulu kemudian dimasukkan ke dalam fungsi *g* dan akan menghasilkan T_1. T_0 dan T_1 selanjutkan dikombinasikan ke dalam sebuah fungsi transformasi *Pseudo-Hadamard* (PHT) lalu ditambahkan dengan 2 buah 32 bit sub kunci dari kunci yang diperluas.

$$T_0 = \mathrm{g}(R_0) \quad [10]$$

$$T_1 = \mathrm{g}(ROL(R_1, 8)) \quad [11]$$

$$F_0 = (T_0 + T_1 + K_{2r+8}) \bmod 2^{32} \quad [12]$$

$$F_1 = (T_0 + T_1 + K_{2r+9}) \bmod 2^{32} \quad [13]$$

F_0 dan F_1 merupakan output dari fungsi *f,* yang masing-masing panjangnya 32 bit. Output fungsi *f* akan dipertukarkan dan dimasukkan kembali ke putaran selanjutanya sampai 16 kali putaran (Jaringan Feistel).

Fungsi *g* merupakan jantung dari keseluruan algoritma *Twofish*. 32 bit yang menjadi input dari fungsi *f* dipecah menjadi 4 bagian, masing-masing bagian tersebut memiliki panjang 8 bit. Setiap 8 bit kemudian di proses dengan kotak-S yang bergantung pada vektor S yang di dapat dari kunci. Setiap kotak-S bersifat bijektif, yaitu menerima input sebesar 8 bit dan mengeluarkan output sebesar 8 bit juga. 4 buah 8 bit hasil dari 4 kotak-S kemudian kemudian diinterpretasikan sebagai vektor yang panjangnya 4 di atas *GF* (2^8), dan dikalikan dengan matriks *Most Distance Separable* (MDS) 4x4 (menggunakan bidang *GF* (2^8) untuk perhitungannya). Vektor yang diinterpretasikan menjadi blok 32-bit, yang juga merupakan hasil dari fungsi *g* :

$$x_i = \lfloor X / 2^{8i} \rfloor \bmod 2^8 \quad i = 0, \ldots, 3 \quad [14]$$

$$y_i = s_i[x_i] \quad i = 0, \ldots, 3 \quad [15]$$

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & \text{MDS} & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix} \quad [16]$$

$$Z = \sum_{i=0}^{3} z_i \cdot 2^{8i} \quad [17]$$

Matriks MDS yang setiap elemennya ditampilkan sebagai heksadesimal adalah sebagai berikut:

$$\text{MDS} = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & 3F & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix} \quad [18]$$

Pada algoritma *twofish* diperlukan 40 kunci yang diperluas dari kunci yang diinputkan oleh pengguna (k_0, k_1, k_2, …, k_{39}). Penjadwalan kunci pada algoritma *twofish* menghasilkan 40 sub kunci yang diperluas dan 4 buah *S-Boxes key-dependent* yang mana nantinya digunakan pada fungsi *g*. *Twofish* mendefinisikan kunci dengan panjang $N = 128$, $N = 192$, dan $N = 256$. Untuk ukuran-ukuran kunci yang tidak didefinisikan, kunci akan mengalami penambahan nilai nol pada bagian akhir kunci sehingga memenuhi persyaratan kunci yang dibutuhkan (*padding*). Selain itu *twofish* mendefinisikan $k = N / 64$ dan juga kunci M yang terdiri dari $8k$ byte $m_0, \ldots, m_{8k-1}$. Kunci M pertama-tama di ubah kedalam $2k$ blok bit yang mana berukuran 32 bit :

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 2k-1 \quad [19]$$

Lalu hasil dari perhitungan diatas dimasukkan ke dalam 2 vektor dari panjang k :

$$M_e = (M_0, M_2, \ldots, M_{2k-2}) \quad [20]$$

$$M_o = (M_1, M_3, \ldots, M_{2k-1}) \quad [21]$$

Untuk vektor s mempunyai perhitungan :

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & \text{RS} & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix} \quad [22]$$

$$S_i = \sum_{j=0}^{3} s_{i,j} \cdot 2^{8j} \quad [23]$$

untuk $i = 0, \ldots, k-1$, dan

$$S = (S_{k-1}, S_{k-2}, \ldots, S_0) \quad [24]$$

Vektor S disusun secara terbalik (*reverse*).
Dalam pemetaan ini, matriks RS di tunjukkan sebagai berikut :

$$\text{MDS} = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix} \quad [25]$$

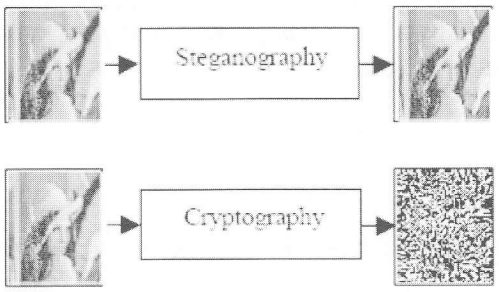
Vektor *Me, Mo,* dan *S* merupakan factor utama untuk membentuk penjadwalan kunci. Dimana Nilai output *Me* dan *Mo* akan digunakan untuk langkah *Expanded Key* dan *S* akan digunakan dalam fungsi *g*. (Schneier et al, 1999)

2.5 Steganografi

Kata steganografi berasal dari bahasa Yunani, yaitu dari kata "*Steganif*" (tersembunyi) dan "*Graptos*" (tulisan). Steganografi di dunia modern biasanya mengacu pada informasi atau suatu arsip yang telah disembunyikan ke dalam suatu arsip citra digital, audio atau video.

Steganografi adalah suatu teknik untuk menyembunyikan informasi yang bersifat pribadi dengan sesuatu yang hasilnya akan tampak seperti informasi normal lainnya. Media yang digunakan pada umumnya merupakan suatu media yang berbeda dengan media pembawa informasi rahasia, dimana disinilah fungsi dari teknik steganografi yaitu teknik penyamanaran menggunakan media lain yang berbeda sehingga informasi rahasia dalam media awal tidak terlihat secara jelas (Waheed, 2000).

Steganografi berbeda dengan kriptografi yang dengan mudah dideteksi dengan panca indera manusia. Perbedaan yang mendasar antara kedua teknik tersebut adalah informasi yang di lindungi dengan teknik steganografi tidak nampak pada indera penglihatan manusia. Gambar 2 yang menunjukkan bagaimana perbedaan antara steganografi dan kriptografi.



Gambar 2. Perbedaan Steganografi dengan Kriptografi
Dikutip dari : Aditya, Y, et al.(2010). *Studi Pustaka untuk Steganografi dengan Beberapa Metode,* hlm G-33

2.6 LSB

Salah satu metode steganografi yang banyak digunakan adalah metode modifikasi LSB (*Least Significant Bit*). Metode LSB tergolong metode yang menggunakan teknik substitusi dan berbasis media (*media-based steganography*).

Metode LSB menyembunyikan data atau pesan rahasia dalam *pixel-pixel* yang tidak signifikan dari sebuah citra digital. Pengubahan nilai *pixel-pixel* yang tidak signifikan pada dasarnya memberikan pengaruh pada berkas citra digital, tetapi karena perubahan yang terjadi sangat kecil, sehingga indra penglihatan manusia tidak dapat menyadari perubahan yang terjadi pada citra digital tersebut. Sebagai contohnya terdapat bit pada citra digital dengan ukuran 3 *pixel* sebagai berikut :

0011111	11101001	11001000
0011111	11001000	11101001
1100000	00100111	11101001

Pesan yang akan disisipkan adalah sebuah karakter 'A', karakter tersebut memiliki nilai biner 01000001, maka *pixel-pixel* pada citra digital tersebut akan dirubah menjadi seperti berikut:

0011110	11101001	11001000
0011110	11001000	11101000
1100000	00100111	11101001

Perubahan yang tidak signifikan seperti contoh diatas tidak akan tertangkap oleh indera penglihatan manusia.

3. Perancangan Program

Pada saat program menjalankan proses enkripsi, pengguna diharapkan menginputkan pesan rahasia berupa file teks (txt), file citra (PNG atau BMP) yang mempunyai ukuran minimal 200 KB, lokasi penyimpanan file output stego (file citra yang disisipi pesan terenkripsi) dan kunci yang akan digunakan untuk proses enkripsi dengan ketentuan kunci dengan panjang 8 – 32 karakter, dan harus mempunyai kombinasi huruf besar, huruf kecil dan angka. File teks akan dienkripsi terlebih dahulu dengan algoritma *twofish* lalu hasilnya akan disisipkan pada file citra yang sudah ditentukan dan outputnya akan berubah file citra mirip dengan file citra inputan.

Pada saat proses dekripsi, pengguna hanya menginputkan file stego dan kunci yang akan digunakan untuk dekripsi yang tentunya sama dengan kunci yang digunakan pada proses enkripsi sebelumnya. Pesan terenkripsi akan diekstrak terlebih dahulu dari file citra stego lalu hasilnya akan dilakukan proses dekripsi dengan kunci yang sudah ditentukan dan outputnya akan berupa file teks yang sama dengan file teks yang disisipkan sebelumnya.

4. Hasil dan Pembahasan

Pada penelitian ini, pengujian dilakukan terhadap 2 macam file citra, 3 macam file teks dan 3 macam kunci. File citra yang digunakan antara lain "Penguin Biru.bmp" (400 x 400 pixels 24 bit, 480,056 bytes) dan "Penguin Tas.png" (400 x 400 pixels 24 bit, 239,351 bytes). File teks yang digunakan antara lain "Pesan Pendek.txt" (2,279 bytes), "Pesan Sedang.txt" (7,992 bytes) dan "Pesan Panjang.txt" (21,293 bytes). Kunci yang digunakan antara lain kunci 128 bit (DaveRSetyawan123), kunci 192 bit (DevriRizaSetyawan1234567) dan kunci 256 bit (ImplementasiTwofishLSB2208456712).

a. Pengujian terhadap Pengaruh Penggunaan Kunci pada Proses Enkripsi terhadap Perubahan Ukuran File yang disisipkan

Tabel 1.
Hasil pengujian enkripsi terhadap file teks dengan menggunakan berbagai kunci.

No	File Citra	File Teks	Kunci (bit)	Sebelum Enkripsi	Sesudah Enkripsi	Sesudah Dekripsi	Sisip Ekstraksi
1	Penguin Biru .bmp	Pesan Pendek.txt	128	2,279	2,888	2,279	V
2	Penguin Biru.bmp	Pesan Sedang.txt	128	7,922	7,936	7,922	V
3	Penguin Biru.bmp	Pesan Panjang.txt	128	21,293	21,296	21,293	V
4	Penguin Tas.png	Pesan Pendek.txt	128	2,279	2,888	2,279	V
5	Penguin Tas.png	Pesan Sedang.txt	128	7,922	7,936	7,922	V
6	Penguin Tas.png	Pesan Panjang.txt	128	21,293	21,296	21,293	V
7	Penguin Biru .bmp	Pesan Pendek.txt	192	2,279	2,888	2,279	V
8	Penguin Biru.bmp	Pesan Sedang.txt	192	7,922	7,936	7,922	V
9	Penguin Biru.bmp	Pesan Panjang.txt	192	21,293	21,296	21,293	V
10	Penguin Tas.png	Pesan Pendek.txt	192	2,279	2,888	2,279	V
11	Penguin Tas.png	Pesan Sedang.txt	192	7,922	7,936	7,922	V
12	Penguin Tas.png	Pesan Panjang.txt	192	21,293	21,296	21,293	V
13	Penguin Biru .bmp	Pesan Pendek.txt	256	2,279	2,888	2,279	V
14	Penguin Biru.bmp	Pesan Sedang.txt	256	7,922	7,936	7,922	V
15	Penguin Biru.bmp	Pesan Panjang.txt	256	21,293	21,296	21,293	V
16	Penguin Tas.png	Pesan Pendek.txt	256	2,279	2,888	2,279	V
17	Penguin Tas.png	Pesan Sedang.txt	256	7,922	7,936	7,922	V
18	Penguin Tas.png	Pesan Panjang.txt	256	21,293	21,296	21,293	V

Catatan :

- V = Berhasil, X = Gagal
- Ukuran file dalam bytes

Dari hasil pengujian pada table 1, dapat disimpulkan bahwa proses enkripsi dengan algoritma *twofish* hampir tidak mengubah ukuran file teks. Dari data didapatkan informasi bahwa perubahan yang terjadi sangatlah kecil sekali sehingga tidak terlalu mempengaruhi kapasitas yang akan dibutuhkan pada file citra yang menjadi wadah steganografi. Hal ini disebabkan karena algoritma *twofish* berjalan pada 128 blok bit, input *plainteks* sebesar 128

bit dan output *cipherteks*-nya sebesar 128 bit juga. Panjang kunci yang digunakan tidak mempengaruhi perubahan besar file output yang dihasilkan dari proses enkripsi dan dekripsi algoritma ini. Panjang kunci yang digunakan hanya berpengaruh dalam proses penjadwalan kunci, kotak S (*S-boxes*) dan PHT (*Pseudo-Hadamard Transforms*) pada algoritma *twofish*. Perubahan kecil yang terjadi pada file teks yang menjadi input proses enkripsi disebabkan oleh penambahan karakter atau yang sering disebut dengan istilah *padding* yang dilakukan supaya memenuhi persyaratan dari algoritma *twofish*.

b. Kasus Perubahan yang Terjadi pada File Citra yang disisipi

Tabel 2.
Hasil pengujian penyisipan pesan terenkripsi pada file citra.

No	File Citra	File Teks	Kunci	Sebelum Disisipi	Sesudah Disisipi
1	Penguin Biru.bmp	Pesan Pendek.txt	128 bit	480,056	480,056
2	Penguin Biru.bmp	Pesan Pendek.txt	192 bit	480,056	480,056
3	Penguin Biru.bmp	Pesan Pendek.txt	256 bit	480,056	480,056
4	Penguin Biru.bmp	Pesan Sedang.txt	128 bit	480,056	480,056
5	Penguin Biru.bmp	Pesan Sedang.txt	192 bit	480,056	480,056
6	Penguin Biru.bmp	Pesan Sedang.txt	256 bit	480,056	480,056
7	Penguin Biru.bmp	Pesan Panjang.txt	128 bit	480,056	480,056
8	Penguin Biru.bmp	Pesan Panjang.txt	192 bit	480,056	480,056
9	Penguin Biru.bmp	Pesan Panjang.txt	256 bit	480,056	480,056
10	Penguin Tas.png	Pesan Pendek.txt	128 bit	239,351	244,018
11	Penguin Tas.png	Pesan Pendek.txt	192 bit	239,351	244,031
12	Penguin Tas.png	Pesan Pendek.txt	256 bit	239,351	244,029
13	Penguin Tas.png	Pesan Sedang.txt	128 bit	239,351	252,979
14	Penguin Tas.png	Pesan Sedang.txt	192 bit	239,351	252,957
15	Penguin Tas.png	Pesan Sedang.txt	256 bit	239,351	252,915
16	Penguin Tas.png	Pesan Panjang.txt	128 bit	239,351	267,926
17	Penguin Tas.png	Pesan Panjang.txt	192 bit	239,351	267,827
18	Penguin Tas.png	Pesan Panjang.txt	256 bit	239,351	267,764

Catatan :

- Ukuran file dalam bytes.

Dari tabel 2 hasil pengujian penyisipan pada kedua file yang berbeda, dapat dilihat bahwa pada file citra BMP tidak mengalami perubahan ukuran file sesudah disisipi pesan terenkripsi sedangkan pada file citra PNG terjadi perubahan yang cukup significant sesuai dengan ukuran file yang disisipkan. Hal tersebut dapat terjadi karena sistem membaca input file citra (BMP maupun PNG) dalam bentuk Bitmap. Jadi citra BMP tidak mengalami perubahan ketika dibaca oleh sistem, sedangkan pada citra PNG mengalami perubahan ketika dibaca dalam bentuk bitmap. Selain itu, dapat dilihat bahwa citra BMP mempunyai daya tampung lebih besar daripada citra PNG. Hal itu dapat dilihat pada citra BMP dan PNG yang sama-sama mempunyai resolusi 400 x 400 *pixel* dan memiliki gradiasi gambar yang hampir sama, file citra BMP mempunyai ukuran file lebih besar daripada file citra PNG.

c. Kasus Ketahanan Output Sistem (File Stego) terhadap Beberapa Operasi Manipulasi

Pada kasus ini, pengujian terhadap ketahanan file stego (file citra yang disisipi file teks yang terenkripsi) dengan melakukan penambahan efek ketajaman (*Sharpen*), penambahan kontras, dan proses transformasi berupa rotasi 90°, 180°, dan 270°.

Tabel 3.
Hasil pengujian ketahanan citra terhadap beberapa operasi manipulasi.

No	File Citra	Operasi Manipulasi	Berhasil Diekstrak	Berhasil Didekripsi
1	Penguin Biru.bmp	Penambahan efek *Sharpen*	X	X
2	Penguin Tas.png	Penambahan efek *Sharpen*	X	X
3	Penguin Biru.bmp	Penambahan kontras	X	X
4	Penguin Tas.png	Penambahan kontras	X	X
5	Penguin Biru.bmp	Rotasi 90^o	X	X
6	Penguin Tas.png	Rotasi 90^o	X	X
7	Penguin Biru.bmp	Rotasi 180^o	X	X
8	Penguin Tas.png	Rotasi 180^o	X	X
9	Penguin Biru.bmp	Rotasi 270^o	X	X
10	Penguin Tas.png	Rotasi 270^o	X	X

Catatan :
V = Berhasil, X = Gagal

Dari hasil pengujian dengan sistem yang telah dibangun, dapat dilihat bahwa file stego (citra yang sudah disisipi pesan terenkripsi) tidak tahan terhadap beberapa operasi manipulasi. File stego yang telah dilakukan operasi manipulasi tidak dapat menghasilkan output ekstrasi yang sama dengan input yang disisipkan bahkan identitas yang disematkan pada proses penyisipan pun nilainya berubah sehingga tidak dapat mengekstrasi pesan yang disisipkan pada file stego. Ketidak tahanan file stego terhadap beberapa operasi manipulasi adalah karena adanya perubahan nilai bit pada tiap *pixel* citra karena operasi manipulasi yang dilakukan pada citra tersebut, sehingga data yang telah disisipkan pun telah berubah termasuk identitas yang disisipkan untuk menandai file stego.

5. Kesimpulan dan Saran

Pada penelitian, ada beberapa hal yang dapat disimpulkan yaitu sebagai berikut :

Twofish adalah sebuah algoritma kriptografi yang baik digunakan dalam penyandian pesan rahasia karena selain algoritma ini merupakan algoritma yang susah dipecahkan oleh para kriptanalisis, hasil dari penyandian pada algoritma ini hampir tidak mengubah ukuran file yang dilakukan penyandian tersebut. Ukuran file hanya berubah sedikit, hal itu terjadi dikarenakan *padding* yang dilakukan pada media yang disandikan supaya memenuhi syarat dari algoritma *twofish* yang mana berjalan pada 128 blok bit (*plainteks* 128 bit *cipherteks* 128 bit).

File citra BMP tidak mengalami perubahan ukuran file setelah disisipi pesan terenkripsi, sedangkan file citra PNG mengalami perubahan ukuran file setelah disisipi pesan. Hal tersebut dapat terjadi karena input citra diproses dengan dalam bentuk bitmap.

Pengujian dari sistem yang dibangun menghasilkan file stego yang tidak tahan terhadap beberapa operasi manipulasi seperti penambahan ketajaman (*Sharpen*), kontras dan transformasi rotasi.

Untuk pengembangan lebih lanjut, bisa dikembangkan dengan menambahkan fungsi untuk membandingkan apakah media penampung sebelum disisipi sama dengan media penampung setelah disisipi apakah sama atau berbeda dan membandingkan file pesan rahasia asli dan pesan rahasia hasil ekstrasi dan dekripsi apakah mempunyai nilai hash yang sama. Penerapan algoritma *twofish* dan LSB dapat dilakukan lebih lanjut terhadap media yang disisipkan dan media yang menjadi penampung. Media yang disisipkan bisa dilakukan penelitian lebih lanjut antara lain citra, audio, atau video. Sedangkan untuk media penampung bisa dilakukan penelitian lebih lanjut antara lain citra lain (JPG, JPEG, TIFF, GIF, dll), audio, atau video.

DAFTAR PUSTAKA

Aditya, Y., Pratama, A., & Nurlifa, A. (2010). *Studi Pustaka untuk Steganografi dengan Beberapa Metode*. Yogyakarta: Universitas Islam Indonesia.

Dony, Ariyus. (2005). *Kriptografi Keamanan Data dan Komunikasi*. Yogyakarta: Penerbit Andi Offset.

Munir, Rinaldi. (2006). *Kriptografi*. Bandung: Penerbit Informatika.

Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., & Ferguson, N. (1999). *The twofish encryption algorithm: a 128-bit block cipher*. New York: J. Wiley.

Waheed, Q. (2000). *Steganography and Steganalysis*. PhD thesis