Implementasi Algoritma *Chaos-Based Feedback Stream Cipher* pada Enkripsi-Dekripsi Data Citra *Digital*

Theresia Anna[1], M. A. Ineke Pakereng [2], Yos Richard Beeh [3].
Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Jl. Diponegoro 52-60 Salatiga 50711
Email: Tere_ceea@yahoo.com, inekep200472@yahoo.com,
Yos.fti.uksw@gmail.com

Abstract

Chaos has suggested some new and efficient way to develop image encryption technique. It is because chaos has a chaotic random characteristic. Therefore chaos often is used as a random generator in image encryption and decryption. One of the chaos functions is logistic map. The logistic map is a polynomial mapping of degree 2, often cited as an example of how complex chaotic behavior can arise from very simple non-linear dynamical equations. The aim of this research is to implement image cryptosystems using chaos technique combined with stream cipher method, feedback function and 256-bit of external key. In stream cipher, chaos technique is used to replace key stream generator to produce random seed. The feedback function is to encrypt with previous plain image. The use of 256-bit external key can be represented as 32 characters in ASCII. The results of testing show that the size of the file don't change before and after encryption and decryption process, there's time difference between encrypt and decrypt process and also different key can influence the output of encryption process.

Keywords : Chaos, Stream cipher, Logistic Map, Image Cryptosystem.

1. Pendahuluan

Keamanan suatu informasi pada jaman global ini menjadi sebuah kebutuhan vital dalam berbagai aspek kehidupan. Suatu informasi akan memiliki nilai lebih tinggi apabila menyangkut tentang aspek-aspek keputusan bisnis, keamanan, ataupun kepentingan umum. Informasi-informasi tersebut tentunya akan banyak diminati oleh berbagai pihak yang juga memiliki kepentingan di dalamnya. Seiring dengan perkembangan jaman dan kemajuan teknologi yang sangat pesat, semakin banyak komputer yang terhubung dalam dunia maya yang biasa dikenal dengan sebutan internet. Banyak hal yang dapat dilakukan melalui internet untuk mempermudah hubungan antar lembaga tersebut. Salah satunya adalah penyampaian data dari pihak satu ke pihak lain ataupun penyampaian data yang dapat diakses oleh publik. Data yang dikirim tidak hanya dalam bentuk teks, tetapi juga dalam bentuk gambar yang sifatnya sangat rahasia bagi suatu perusahaan atau instansi terkait.

Dengan bertambahnya penggunaan teknik *digital* bagi pengiriman atau penyimpanan suatu data citra, maka keamanan dalam proses pemindahan data amat

[1] Mahasiswa Fakultas Teknologi Informasi Universitas Kristen Satya Wacana
[2] Staff Pengajar Fakultas Teknologi Informasi Universitas Kristen Satya Wacana
[3] Staff Pengajar Fakultas Teknologi Informasi Universitas Kristen Satya Wacana

sangat diperlukan. Salah satu algoritma yang dapat digunakan untuk mengenkripsi sebuah citra adalah algoritma *Chaos-Based FeedBack Stream Cipher* (CBFSC). Hubungan yang dekat antara *chaos* dan kriptografi membuat *chaos* digunakan sebagai dasar dari teknik kriptografi yang aman. Metode CBFSC menggunakan kunci sepanjang 256-bit dan kunci ini digunakan untuk menghasilkan nilai awal *logistic map*. Aplikasi yang akan dibuat menggunakan *Microsoft Visual Studio* 2005 untuk proses enkripsi dan dekripsinya.

2. Landasan Teori

Penyandian *file* gambar dengan menggunakan teknik enkripsi dan dekripsi dibahas dalam dua penelitian mahasiswa Fakultas Teknologi Informasi UKSW Salatiga. Penelitian pertama dengan judul "Penyandian *File* Gambar Dengan Menggunakan Metode Substitusi dan Transposisi". Teknik yang dilakukan adalah mengacak posisi dan susunan *pixel* pada *file* gambar dengan menggunakan metode substitusi dan transposisi. *File* gambar yang digunakan sebagai inputan adalah *file* gambar bertipe *BMP* 32 *bit*. Teknik enkripsi ini tidak cocok untuk gambar dengan warna homogen, karena gambar akan lebih mudah teridentifikasi.[1]

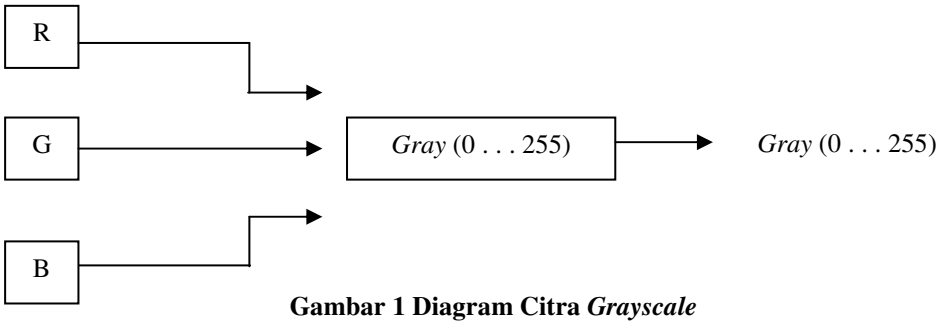
Pada penelitian dengan judul "Perancangan Perangkat Lunak Untuk Penyembunyian Data *Digital* Menggunakan 4-*Least Significant Bit Encoding* dan Visual Kriptografi" membahas tentang teknik penyembunyian data *digital* dengan menyembunyikannya pada gambar yang lain. Satu gambar sebagai *file* yang akan disembunyikan dinamakan *file share* dan satu gambar lagi yang digunakan sebagai tempat untuk menyembunyikan dinamakan *file cover*. Data *digital* yang digunakan adalah data *digital* bertipe *BMP*, *JPG*/*JPEG*, dan *PNG*. Kelemahan pada sistem ini adalah *file cover* harus lebih besar daripada *file* yang akan disembunyikan (*file share*) sehingga semakin besar *file share* maka semakin besar pula *file cover*-nya.[2]

Pada penelitian ini penulis membuat implementasi teknik enkripsi dan dekripsi pada data citra dengan menggunakan salah satu teknik *chaos* yaitu *logistic map* yang dirasa lebih kuat karena dapat menghasilkan bilangan random yang benar-benar acak. Teknik *chaos* ini akan diterapkan dengan algoritma *stream cipher* untuk menggantikan *key stream generator* ditambahkan fungsi *feedback* untuk mengenkripsi dengan menggunakan *plain image pixel* sebelumnya. Data citra yang digunakan adalah data citra *bitmap* bertipe **.bmp*.

Citra digital merupakan suatu matriks yang terdiri dari baris dan kolom, dimana setiap pasangan indeks baris dan kolom menyatakan suatu titik pada citra. Nilai matriksnya menyatakan nilai kecerahan titik tersebut. Titik-titik tersebut dinamakan sebagai elemen citra atau *pixel (picture element)* [3]. Citra yang dimiliki 16 derajat keabuan (mulai dari 0 mewakili warna hitam sampai 15 yang mewakili warna putih) direpresentasikan oleh 4 bit data. Sedangkan citra dengan 256 derajat keabuan (nilai 0 mewakili warna hitam sampai dengan 256 mewakili warna putih) direpresentasikan oleh 8 bit data.

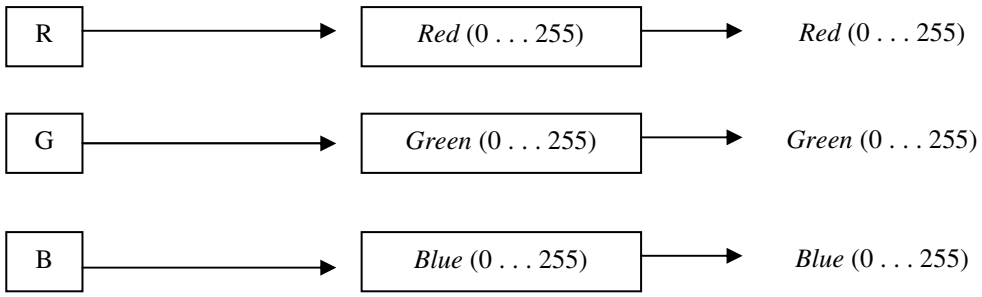
Citra hitam-putih disebut juga citra satu kanal, karena warnanya hanya ditentukan oleh satu fungsi intensitas saja. Citra berwarna (*color images*) dikenal dengan nama citra spektral, karena warna pada citra disusun oleh tiga komponen warna yang disebut komponen RGB, yaitu: merah (*red*), hijau (*green*) dan biru (*blue*). Intensitas suatu titik pada citra berwarna merupakan kombinasi dari tiga intensitas: derajat keabuan merah (*fmerah* (*x, y*)), hijau (*fhijau* (*x,y*)), dan biru (*fbiru* (*x, y*)). Dalam citra berwarna, jumlah warna bisa beragam mulai dari 16, 256, 65536 atau 16 juta warna yang masing-masing direpresentasikan oleh 4, 8, 16 atau 24 bit data untuk setiap *pixel*-nya. Pada citra *grayscale* (8 bit) nilai warna primer (merah, hijau, biru) mempunyai nilai yang sama yaitu antara 0 - 255. Citra *gray* merupakan citra dua dimensi, yang direpresentasikan ke dalam sebuah matriks. Posisi baris dan kolom pada matriks menunjukkan posisi piksel pada citra, sedangkan warna piksel adalah nilai yang tersimpan dalam citra.

Gambar 1 adalah skema citra *grayscale* dengan nilai R, G, B mempunyai nilai yang sama, artinya nilai komponen tiga warna dasar (R,G,B) pada (x,y) mempunyai nilai yang sama.



Gambar 1 Diagram Citra *Grayscale*

Gambar 2 adalah skema citra berwarna di mana masing-masing warna R,G dan B mempunyai nilai 0...255.



Gambar 2 Diagram Citra *Color*

Secara matematik teori *chaos* menggambarkan tingkah laku dari sebuah sistem dinamik yang menunjukkan fenomena *chaos*. Salah satu karakteristik sistem *chaos* yaitu peka terhadap nilai awal (sering disebut dengan *buterfly effect*). Hasil dari kesensitifan ini menunjukkan sebuah pertumbuhan yang eksponensial dari kekacauan di dalam kondisi inisial, tingkah laku dari sistem *chaotic* ini muncul secara acak atau *random*. Ini tetap terjadi walaupun pada sistem ini bersifat deterministik, yang berarti bahwa kejadian dinamik yang terjadi ditentukan oleh kondisi inisial mereka dan tidak mempunyai parameter acak. Kelakuan seperti ini dikenal dengan nama *deterministic chaos* atau *chaos*. Sistem *chaos* berguna untuk pembangkitan bilangan acak dan bilangan acak dengan *chaos* tidak mempunyai periode. Hubungan yang sangat dekat antara *chaos* dan kriptografi membuat *chaos* sebagai dasar dari algoritma kriptografi yang aman untuk komunikasi dan kriptografi. Teknik enkripsi berdasarkan *chaos* dianggap baik dalam praktek penggunaannya, karena didukung oleh 3 (tiga) teknik yaitu kecepatan, tingkat keamanan yang tinggi dan kompleksitasnya [4]. Dalam *stream cipher* teknik *chaos* digunakan untuk menggantikan *keystream generator* yaitu untuk membangkitkan aliran kunci bilangan acak yang selanjutnya kunci tersebut akan digunakan untuk proses enkripsi dan dekripsi. Salah satu fungsi yang paling sederhana dari fungsi *chaos* adalah *logistic map*. *Logistic map* adalah pemetaan polinomial tingkat dua, sering disebut sebagai contoh dasar tentang seberapa kompleksnya *chaotic behaviour* dapat meningkat dari persamaan non-linear yang sangat sederhana. Hal ini dikemukakan dalam surat kabar tahun 1976 oleh seorang ahli biologi yang bernama *Robert May*. Persamaan logistik ini pertama kali dibuat oleh *Pierre François Verhulst* [5]. Secara matematis fungsi dari persamaan *logistic map* dapat direpresentasikan dalam rumus persamaan ke satu [3].

$$X_{n+1} = r\, X_n \left(1 - X_n\right) \qquad (1)$$

dimana:

X_n : adalah nilai awal iterasi memiliki nilai antara 0 dan 1 $\left(0 \le X \le 1\right)$, untuk tahun ke – n.

r : adalah laju pertumbuhan fungsi $\left(0 \le X \le 4\right)$.

Henon Map

Henon Map adalah sebuah sistem dinamik yang berlainan yang merupakan salah satu contoh fungsi dari *logistic map*. *Henon Map* mengambil titik (x,y) dari sebuah bidang dan memetakannya pada titik yang baru. Secara matematis rumus dari *henon map* dijelaskan pada rumus ke dua [5].

$$x_n + 1 = Y_n + 1 - ax_n^2$$
$$y_n + 1 = bx_n \quad (2)$$

dimana,
x, y adalah nilai awal (*initial point*) dari titik (x,y)
a dan *b* adalah parameter menurut *canonical Henon map* nilai dari *a* = 1.4 dan *b* = 0.3 [6].

Arnold's Cat Map

Arnold's Cat Map adalah salah satu fungsi dari *logistic map* yang ditemukan oleh *Vladimir Arnold* pada tahun 1960an. *Arnold's Cat Map* digambarkan dengan menggunakan *image* seekor kucing [6]. Fungsi dari *Arnold's map* dapat dijelaskan pada rumus persamaan nomor tiga [5].

$$f(x,y) = \begin{cases} (2x, y/2), & 0 \le x \le 1/2, 0 \le y \le 1 \\ (2x-1, (y+1)/2), & 1/2 \le x \le 1, 0 \le y \le 1 \end{cases} \quad (3)$$

dimana,
x, y adalah nilai awal (*initial point*) dari titik (x,y)

Algoritma CBFSC ini diambil dari jurnal *An Efficient Chaos Based-Feedback Stream Cipher* oleh Hossam El-din H, Hamdy M. Kalash, dan Osama S. Farag Allah tahun 2005 [7].

Algoritma CBFSC ini menggunakan kunci sepanjang 256-*bit* atau 32 karakter (ASCII). Proses utama dalam CBFSC terdiri dari:

- Proses *Mapping Session* M1, adalah pemetaan panjang kunci dan semua *integer* [0...255] ke dalam *logistic map* dengan interval [0...1] dan M2, mengembalikan interval [0…1] ke interval [0…255].
- Proses pemberian nilai awal (*Initial Condition*) X_0 untuk *logistic map* dan C_0 untuk inisialisasi kode.

$$R = \sum_{i=1}^{32} (M1[K_i]) \quad (4)$$

$$X_0 = R - \lfloor R \rfloor \quad (5)$$

$$C_0 = \left[\sum_{i=1}^{32} (K_i) \right] \text{Mod } 256 \quad (6)$$

dimana *R*, K_i, M1 berturut-turut adalah bilangan *real*, kunci dan pemetaan panjang kunci dan *integer* [0...255] ke dalam interval [0...1]

- Proses nilai *input* untuk *logistic map* X_i dan Iterasi $\#_i$ (data nilai ke – i)

$$X_i = M1[X_{i-1} + C_{i-1} + K_i] \quad (7)$$

$$\#_i = K_{i+1} + C_{i-1} \quad (8)$$

- Proses Enkripsi dan Dekripsi

Proses Enkripsi

$$C_i = \left(P_i + M2 \left[\sum_{i=1}^{\#i} r\, X_i (1 - X_i) \right] \right) \text{mod } 256 \quad (9)$$

dimana,

r, didapatkan dari perhitungan *logistic map*. *r* = 3.9

Ci, *cipher image* (*file* gambar yang akan dienkripsi)

Pi, *plain image* (*file* gambar original)

M2, adalah pemetaan peta *logistic map*[0...1] kembali ke interval [0...255]

Proses Dekripsi

$$P_i = \left(C_i - M2 \left[\sum_{i=1}^{\#i} r\, X_i (1 - X_i) \right] \right) \quad (10)$$

dimana,

r, didapatkan dari perhitungan *logistic map*. *r* = 3.9

Ci, *cipher image* (*file* gambar yang akan dienkripsi)

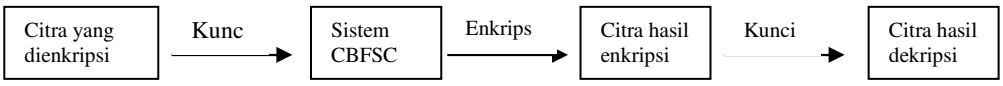
Pi, *plain image* (*file* gambar original)

M2, adalah pemetaan peta *logistic map*[0...1] kembali ke interval [0...255]

3. Metode Penelitian dan Perancangan Aplikasi

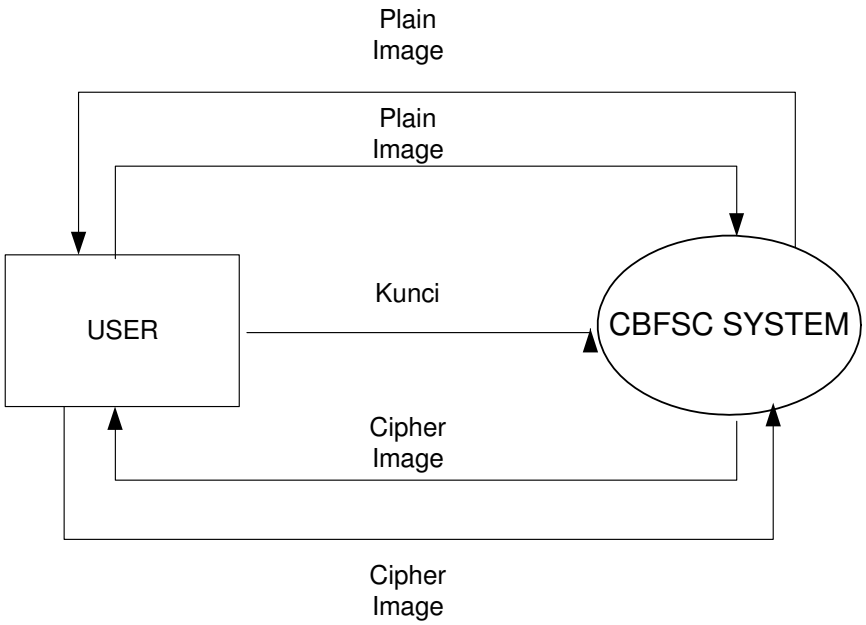
Penelitian ini merupakan implementasi algoritma CBFSC pada citra *digital* 2 (dua) dimensi dari jurnal yang dibuat oleh Hossam El-din H. Ahmed, dkk pada tahun 2005 [7]. Pada penelitian terdahulu implementasinya dibuat dengan menggunakan bahasa pemrograman Borland C++. Penelitian dimulai dengan mempelajari algoritma CBFSC dan kode program sebagai hal yang mendasar dari perancangan aplikasi, serta studi kepustakaan dan perancangan desain antarmuka sebagai penghubung antara pengguna dan aplikasi. Implementasi aplikasi dengan menggunakan bahasa pemrograman *Microsoft Visual Studio* 2005 dan proses pengujian enkripsi dan dekripsi data citra *digital* 2 (dua) dimensi bertipe *bitmap* (**.bmp*). Pengujian dengan menggunakan spesifikasi komputer : *Intel Pentium* IV 2.0 GHz, *RAM* : 512 Mb (atau lebih), *harddisk* 40 GB, *Monitor Mouse* dan *keyboard*.

Aplikasi yang akan dibuat akan memanfaatkan bahasa pemrograman *Microsoft Visual Studio* 2005 dan studi kepustakaan terhadap berbagai referensi yang berkaitan dengan penelitian yang dilakukan. Penelitian dimulai dengan merancang perangkat lunak enkripsi dan dekripsi data citra *digital*, kemudian membuat program aplikasinya dan melakukan pengujian kerja sistem. Proses keseluruhan dari sistem yang dibuat dapat dilihat pada Gambar 3.



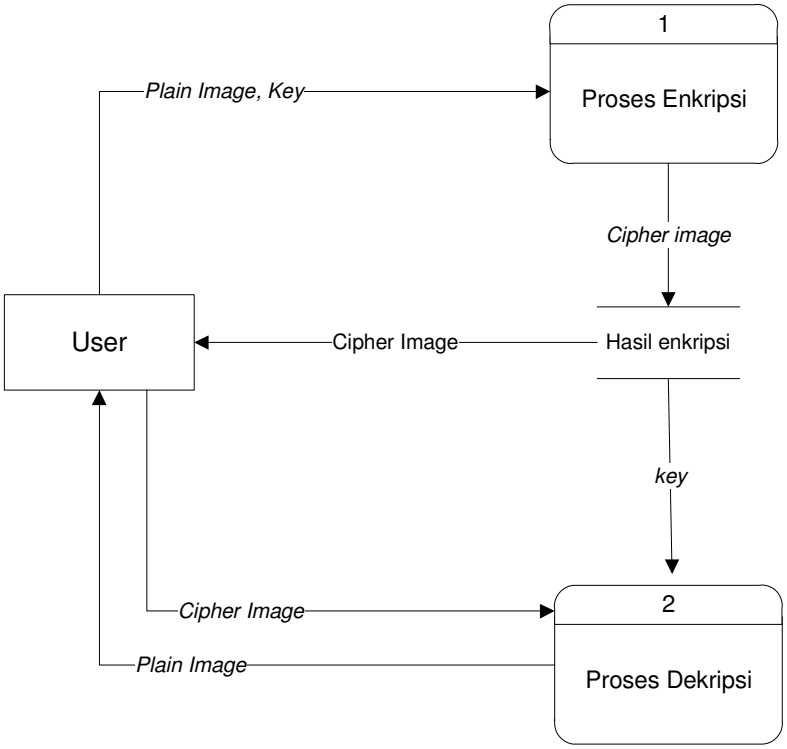
Gambar 3. Proses Keseluruhan Sistem CBFSC

Diagram Konteks dari sistem yang dibangun dapat dilihat pada Gambar 4.



Gambar 4. DFD Konteks Sistem CBFSC

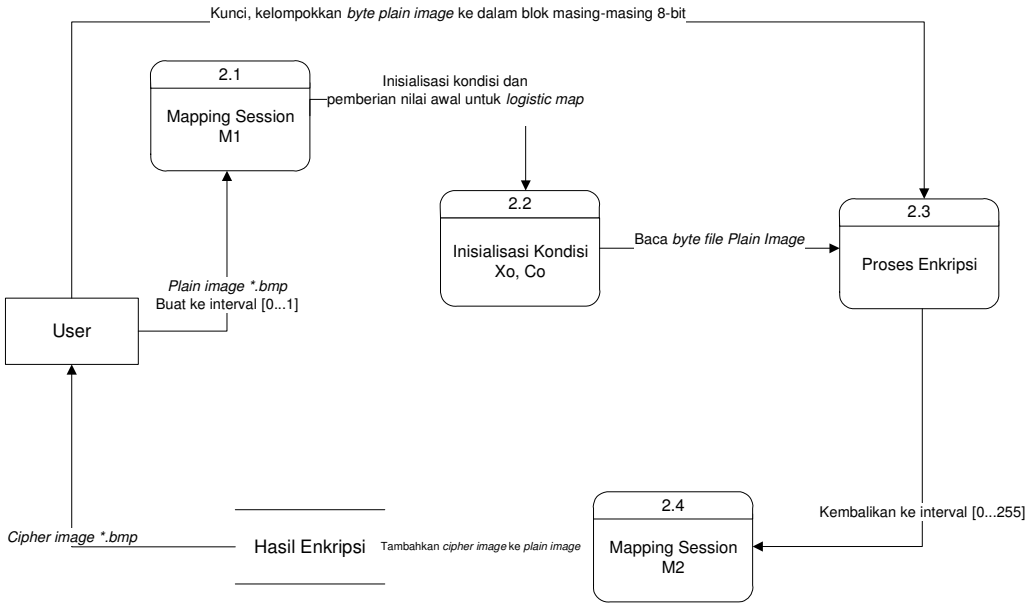
Gambar 4 menggambarkan diagram konteks dari kriptosistem yang menggunakan algoritma *chaos-based feedback stream cipher* (CBFSC) pada *file* gambar bertipe **.bmp*, *user* memasukkan kunci, *file* gambar **.bmp* asli (*plain image*) atau *file gambar *.bmp* hasil enkripsi (*cipher image*) ke dalam sistem. Sistem akan memberikan keluaran berupa file hasil enkripsi (*cipher image*) atau *file* gambar asli (*plain image*). Sistem kriptografi CBFSC diuraikan lebih lanjut dalam DFD *level* 1 seperti terlihat pada Gambar 5.



Gambar 5. Diagram DFD *level* 1 (satu) Sistem CBFSC

Pada Gambar 5 terlihat ada dua proses yang terjadi dalam sistem, yaitu : proses enkripsi dan proses dekripsi. Pada proses enkripsi, *user* memasukkan *plain image* dan kunci sepanjang 32 karakter (ASCII). Pada proses dekripsi, *output* dari *logistic map* diambil dari *pixel cipher image* yang akan menghasilkan *plain image*.

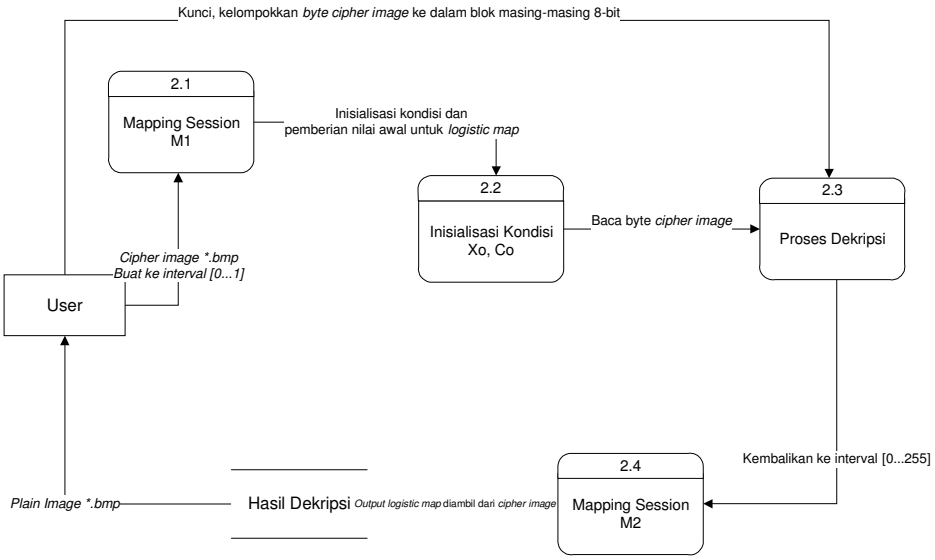
Untuk menjelaskan proses enkripsi CBFSC, maka akan dijabarkan DFD *level* 2 proses Enkripsi Algoritma CBFSC yang terlihat pada Gambar 6.



Gambar 6. DFD *Level* 2 (dua) Proses Enkripsi Algoritma CBFSC

Gambar 6 menjelaskan tentang DFD *level* 2 (dua) dari proses enkripsi algoritma CBFSC. Ada empat proses yang terjadi di dalam sistem, yaitu: Proses *mapping session* M1, terjadi proses pemetaan dari interval [0...255] ke dalam peta *logistic* dalam interval [0...1]. Proses pemberian nilai awal (*initial condition*) C_0 dan X_0, C_0 untuk inisialisasi kode dan X_0 adalah nilai awal untuk *logistic map*. Pada proses enkripsi, *user* akan meng*input*kan *file* gambar (*plain image*) yang bertipe *bitmap* (**.bmp*). Pada proses *mapping session* M2 terjadi proses pemetaan dari peta *logistic* dalam interval [0...1] ke dalam interval [0...255]

Proses dekripsi algoritma CBFSC dapat dilihat dalam DFD *Level* 2 (dua) pada Gambar 7.



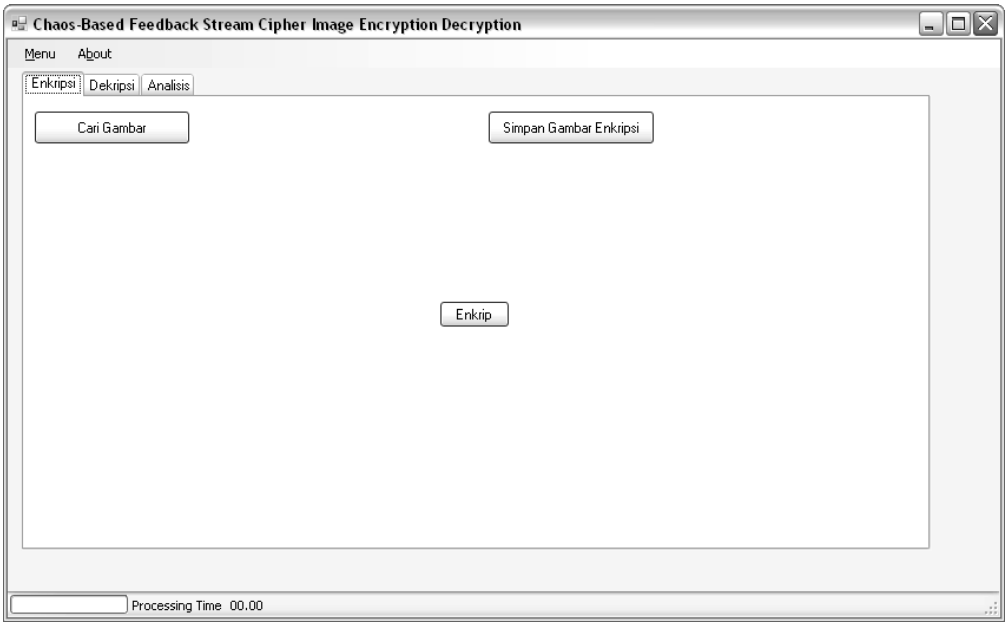
Gambar 7. DFD *Level* 2 (dua) Proses Dekripsi Algoritma CBFSC

Gambar 7 menjelaskan tentang DFD *level* 2 (dua) dari proses dekripsi algoritma CBFSC. Ada empat proses yang terjadi di dalam sistem, yaitu: Proses *mapping session* M1, terjadi proses pemetaan dari interval [0...255] ke dalam peta *logistic* dalam interval [0...1]. Proses pemberian nilai awal (*initial condition*) C_0 dan X_0, C_0 untuk inisialisasi kode dan X_0 adalah nilai awal untuk *logistic map*. Pada proses dekripsi, *user* akan meng*input*kan *file* gambar (*plain image*) yang bertipe *bitmap* (**.bmp*) dan pada proses *mapping session* M2, terjadi proses pemetaan dari peta *logistic* dalam interval [0...1] ke dalam interval [0...255]

4. Implementasi dan Pengujian Perangkat Lunak

Perangkat lunak yang dibuat adalah perangkat lunak untuk melakukan enkripsi dan dekripsi data citra *digital* 2 (dua) dimensi dan diberi nama *'ChaosApplication1'*. Pada tampilan awal *user* perlu memasukkan kunci dengan memilih '*password*' pada submenu '*menu*' sepanjang 32 karakter (ASCII). *Pada Tabcontrol* enkripsi *user* dapat memasukkan *file* yang akan dienkripsi dengan menekan tombol 'Cari Gambar', kemudian dapat menekan tombol 'Enkrip*'* yang ada pada aplikasi untuk melakukan proses enkripsi, setelah itu *file* akan disimpan dengan menekan tombol 'simpan gambar'. *File* yang akan tersimpan yaitu *file bitmap* (**.bmp*). Kemudian untuk melakukan proses dekripsi, *user* dapat memilih *tabcontrol* dekripsi dengan memasukkan *file* yang akan didekripsi yang bertipe *bitmap* (**.bmp*), lalu menekan tombol '*Decrypt*' yang ada pada sistem. Proses dekripsi akan berlangsung dan untuk menyimpan *file* hasil dekripsi (*plain image*) dengan menekan tombol '*Save*' dan untuk mengakhiri aplikasi dengan menekan tombol '*Exit*' pada submenu '*Menu*'. *User* juga dapat melakukan analisis perbedaan *byte* antara file original (*plain image*) dengan file hasil dekripsi (*cipher image*) pada *tabcontrol* 'analisis'.

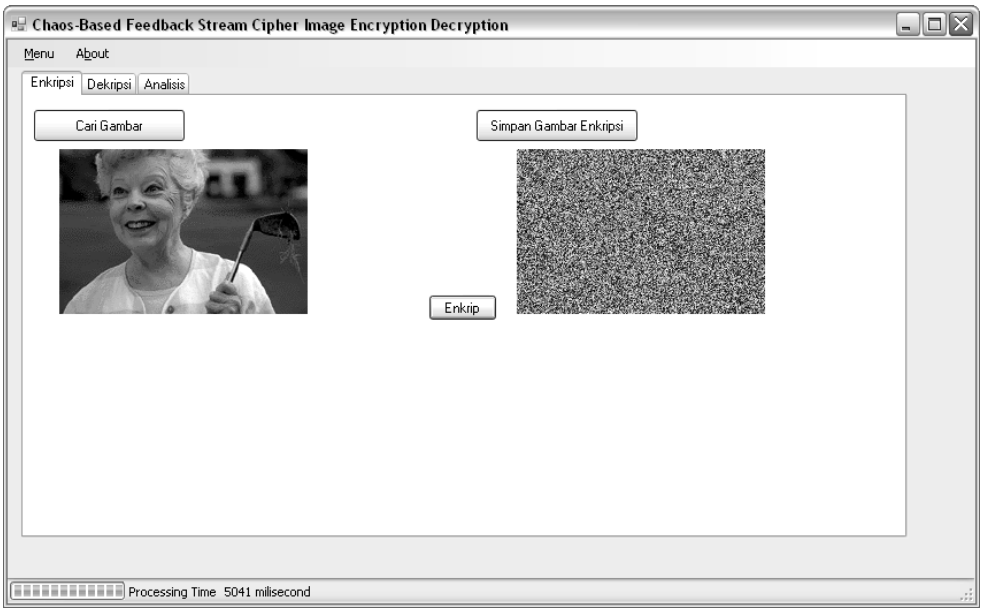
Form utama dalam aplikasi ini adalah *formChaos*, dapat dilihat pada Gambar 8.



Gambar 8. *formChaos*

Pada *form chaos* terdapat 2 (dua) pilihan *menu* dan 3 (tiga) *tabcontrol,* yaitu : Pilihan Menu *Menu*, terdapat pilihan untuk *Password* dan *Exit*. Menu *About*, terdapat pilihan tentang *about program*. *Tabcontrol* enkripsi untuk melakukan proses enkripsi. *Tabcontrol* dekripsi, untuk melakukan proses dekripsi. *Tabcontrol* Analisis, untuk melakukan proses analisis.

Proses Enkripsi

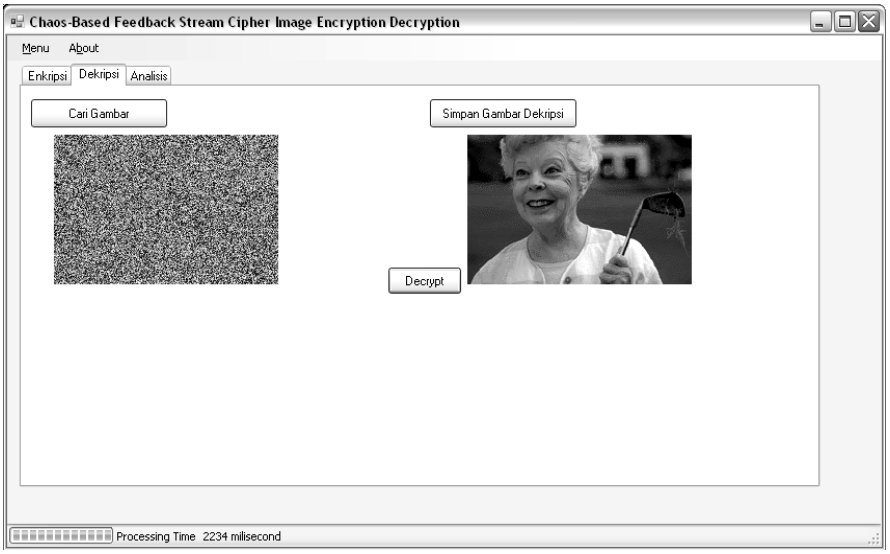


Gambar 9. *FormChaos*, Proses Enkripsi

Pada proses enkripsi (Gambar 9), sistem berfungsi untuk melakukan proses *enciphering* dengan kunci yang telah di*input*kan. Prosesnya adalah :

- Fungsi *Mapping*
  Hasil dari nilai *chaos* berupa nilai *real* (*continuous*), di sisi lain komputer memproses enkripsi dan dekripsi dalam bentuk bilangan diskrit 0 dan 1. Untuk itu diperlukan suatu fungsi lain yang akan mengkonversi nilai hasil *chaos* menjadi berbentuk *integer*. Dalam hal ini, digunakanlah dua fungsi *mapping session*, yaitu M1 dan M2. M1 memetakan panjang kunci dan semua *integer* dari *interval* [0...255] ke dalam peta *logistic map* dalam *interval* [0...1]; sedangkan M2 mengembalikan interval [0...1] dari *logistic map* ke dalam interval [0...255].
- Nilai awal (*Initial Condition*)
  Memberikan nilai awal untuk *logistic map* X_0 dan kondisi inisial C_0. X_0 dan C_0 digunakan untuk menghasilkan *random seed*. *Random seed* yang dihasilkan akan digunakan untuk perhitungan pada proses enkripsi.
- Proses Enkripsi
  Proses enkripsi dimulai setelah *byte* ke-60, hal ini dimaksudkan untuk menyimpan informasi data *header* yang ada pada *file* gambar tersebut. Pada proses enkripsi, teknik *chaos* dipakai sebagai penghasil nilai acak atau *random*, supaya nilai *Cipher* yang dihasilkan juga memiliki faktor acak, maka digunakan nilai *r* yang didapatkan dari perhitungan *logistic map*. Nilai *r* ini akan dijumlahkan dengan nilai P_i (*Plain Image*) yang tentu saja juga dikombinasikan dengan nilai X_i.

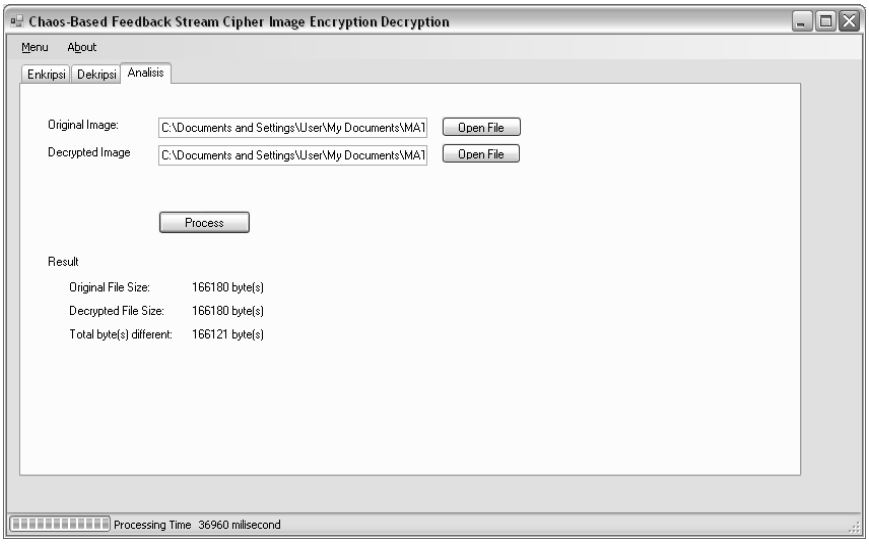
Proses Dekripsi



Gambar 10. *FormChaos*, Proses Dekripsi

Pada proses dekripsi (Gambar 10), sistem berfungsi untuk melakukan proses *deciphering* yaitu proses untuk mengembalikan *cipher image* ke *plain image*. Modul dekripsi menerima *cipher image* dan kunci sepanjang *256-bit* atau 32 karakter (ASCII). Pada proses dekripsi, *output* dari *logistic map* diambil dari *pixel* yang dimiliki oleh *cipher image*.

Proses Analisa



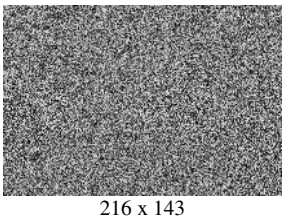
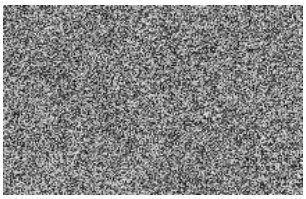
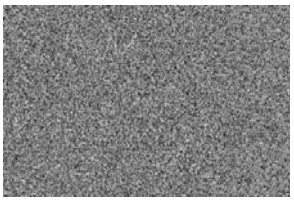
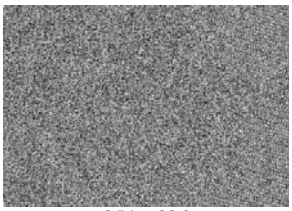
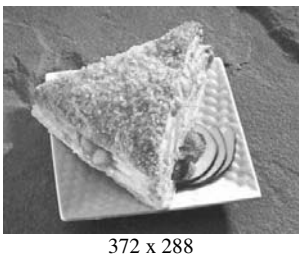
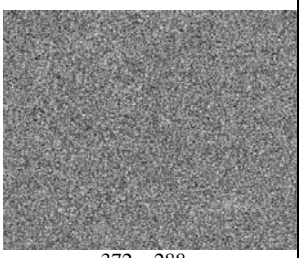
Gambar 11. *FormChaos*, Proses Analisa

Tahap analisa pada Gambar 11 digunakan unutk melihat selisih *byte* yang berbeda antara *file original* dengan *file* hasil enkripsi, karena proses enkripsi dimulai setelah *byte* yang ke-60, tahap analisa ini hanya untuk membuktikan bahwa *byte* yang beda antara *file original* dengan *file* hasil dekripsi adalah setelah *byte* yang ke-60.

Setelah sistem selesai dibuat, dilakukan 2 (dua) pengujian terhadap sistem ini. Pengujian dilakukan untuk melihat perbedaan hasil enkripsi dengan kunci yang berlainan, perbedaan waktu terhadap ukuran *file* gambar dan pengujian untuk melihat apakah *file* hasil dekripsi sama dengan *file original*

1. Pengujian terhadap *plain image* yang sama, tapi dengan kunci yang berbeda (*key sensitive*). Hasil pengujian terlihat pada Tabel 1.

Tabel 1. Tabel Pengujian *Plain Image* yang Sama dengan Kunci yang Berbeda

Original Image	Enkrip dengan kunci abcdefghijklmnopqrstuvwxyzabcdef	Enkrip dengan kunci a1b2c3d4e5f6g7h8i9j10k11l12m13n8
 216 x 143 31.2 KB	 216 x 143 31.2 KB	 216 x 143 31.2 KB
 354 x 236 83.1 KB	 354 x 236 83.1 KB	 354 x 236 83.1 KB
 372 x 288 105 KB	 372 x 288 105 KB	 372 x 288 105 KB

Berdasarkan Pengujian *plain image* yang sama dengan kunci yang berbeda pada Tabel 1 dapat disimpulkan bahwa kunci mempengaruhi hasil enkripsi. Hal ini disebabkan karena kunci yang di*input*kan memiliki nilai ASCII sendiri-sendiri, misalnya:

Kunci 1 : abcdefg
Nilai ASCII : 97, 98, 99, 100, 101, 102, 103
Kunci 2 : hijklmn
Nilai ASCII : 104, 105, 106, 107, 108, 109, 110

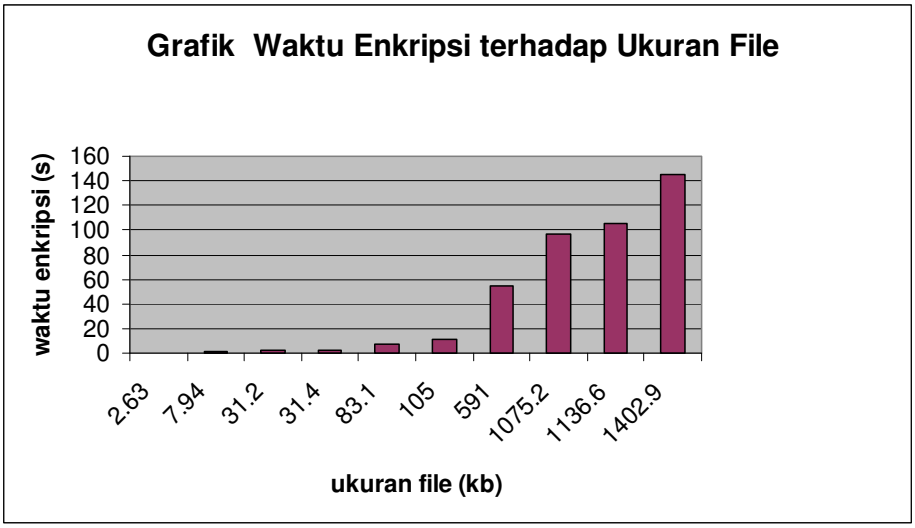
Nilai-nilai tersebutlah yang akan digunakan dalam perhitungan untuk mencari nilai X_0 dan C_0. X_0 dan C_0 adalah *random seed* sehingga bila kunci berbeda maka semua perhitungannya juga akan berbeda.

2. Pengujian perbedaan waktu terhadap *file* gambar. Hasil pengujian terlihat pada Tabel 2

Tabel 2. Pengujian Terhadap Waktu dengan Besarnya Ukuran *File*

Nama *file* gambar (dalam *.bmp)	Ukuran *file* gambar (dalam píxel)	Kedalaman Warna (dalam bit)	Waktu enkripsi (dalam milisecond)	Waktu dekripsi (dalam milisecond)
Iruin	111 x 63	8	766	574
Bigfoot	134 x 38	4	281	210
Girl	1600 x 1200	32	716181	598767
Dress	389 x 529	32	74140	91819
Short	320 x 480	32	76527	48054
Kagaya	1024 x 768	16	132469	111662
Avril	500 x 333	24	44381	44059
triangulasi1	1280 x 1024	32	468973	398718
DSC00492	1654 x 1181	32	708066	566971
DSC00476	2048 x 1536	32	1142152	953400
DSC00526	1536 x 2048	24	848557	68440

Dari Tabel 2 dapat disimpulkan bahwa waktu proses enkripsi lebih lama daripada proses dekripsi. Hal ini dikarenakan pada saat melakukan proses enkripsi terdapat beberapa langkah yang harus dilakukan daripada proses dekripsi yaitu pembentukan kunci dan melakukan proses inisialisasi kondisi X_0 dan C_0. Sedangkan pada proses dekripsi kunci diambil dari kunci yang terbentuk dari proses enkripsi. Waktu proses enkripsi berbanding lurus dengan besarnya ukuran sebuah *file*. Semakin besar ukuran *file* semakin banyak waktu yang dibutuhkan, begitu juga dengan sebaliknya semakin kecil ukuran *file* semakin cepat proses enkripsinya. Untuk lebih jelasnya pengujian terhadap waktu dengan besarnya ukuran *file* dapat dilihat pada grafik Gambar 12

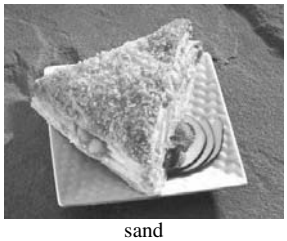
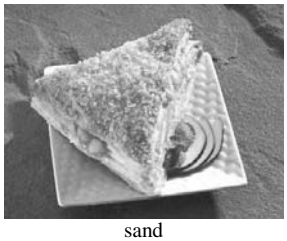


Gambar 12. Grafik Waktu terhadap Ukuran *File*

Dari grafik pada Gambar 12 dapat disimpulkan bahwa ukuran file berbanding lurus dengan waktu proses enkripsi. Semakin besar ukuran *file* semakin lama waktu yang dibutuhkan untuk proses enkripsi, dan semakin kecil ukuran *file* semakin cepat proses enkripsinya Hal ini disebabkan karena modul enkripsi membaca *byte* dari file gambar, jika ukuran *file* lebih besar berarti lebih banyak *byte* data yang harus diproses, begitu juga sebaliknya.

3. Pengujian untuk melihat apakah *file* hasil dekripsi serupa *file original*. Hasil pengujian terlihat pada Tabel 3.

Tabel 3 . Pengujian *File Original* dengan *File* Hasil Dekripsi

File Original (**.bmp*)	*File* hasil dekripsi (**.bmp*)	*File* Duplikat
 Nenek 216 x 143 31.2 KB	 Nenek_e 216 x 143 31.2 KB	ya
 Wed 354 x 236 83.1 KB	 Wed_e 354 x 236 83.1 KB	Ya
 sand 372 x 288 105 KB	 sand 372 x 288 105 KB	Ya

Dari Tabel 3 disimpulkan *file* hasil dekripsi serupa dengan *file original*. Hal ini dibuktikan dengan menggunakan progam *PictMatch Version 2* yang di-*download* dari *internet*

5. Simpulan

Berdasarkan hasil implementasi algoritma *chaos-based feedback stream cipher* pada enkripsi-dekripsi data citra *digital* dan pengujian dari aplikasi yang telah dibuat, maka dapat ditarik beberapa kesimpulan. Ukuran *file* gambar **.bmp* sebelum dan sesudah enkripsi tetap sama karena proses enkripsi dan dekripsi pada algoritma *chaos-based feedback stream cipher* pada dasarnya sama karena menggunakan kunci yang sama. Waktu proses enkripsi berbanding lurus dengan besarnya ukuran sebuah *file*. Semakin besar ukuran *file* semakin banyak waktu yang dibutuhkan, begitu juga dengan sebaliknya semakin kecil ukuran *file* semakin cepat proses enkripsinya. Perbedaan kunci dapat mempengaruhi hasil enkripsi. Hal ini disebabkan kunci yang digunakan untuk menghasilkan nilai X_0 dan C_0. X_0 dan C_0 adalah *random seed* sehingga bila kunci berbeda maka semua perhitungannya juga berbeda.

Daftar Pustaka

[Put07] Puteri, 2007.; Penyandian *File* Gambar Dengan Menggunakan Metode Substitusi dan Transposisi, FTI, UKSW.

[Wid08] Widodo, 2008.; Perancangan Perangkat Lunak Untuk Penyembunyian Data *Digital* Menggunakan 4-*Least Significant Bit Encoding* dan Visual Kriptografi, FTI, UKSW.

[Gon92] Gonzales Rafael C. and Woods Richard E., 1992, *Digital Image Processing*, second edition, Pretince Hall.

[Sca98] J. Scaringer, 1998, Fast Encrytption of image data using chaotic Kolmogrov flow, *J. Electronic Eng 7 (6), pp 318-437.*

[Wei09] Weisstein, Eric W. Logistic Map. From MathWorld--A Wolfram Web Resource. *http://mathworld.wolfram.com/LogisticMap.html* diakses tanggal 15 April 2009

[Arn68] V. I. Arnold and A. Avez, 1968, Ergodic Problem in Classical Mechanics.

[Hos07] Hossam El-din H. Ahmed, dkk., 2007. An Efficient Chaos-Based Feedback Stram Cipher (ECBFSC) for Image Encryption and Decryption. *Menoufia University, Faculty of Electronic Engineering*, *Dept. of Computer Science & Engineering*, *32952, Menouf*, *Egypt*.