Algoritma Enkripsi Citra dengan *Pseudo One-Time Pad* yang Menggunakan Sistem *Chaos*

Rinaldi Munir
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung (ITB)
Jl. Ganesha 10, Bandung 40132 Indonesia
Email : rinaldi-m@stei.itb.ac.id

ABSTRACT

Karateristik sistem chaos yang sensitif pada perubahan kecil parameter nilai awalnya telah banyak digunakan untuk aplikasi kriptografi. Makalah ini menyajikan algoritma sederhana untuk mengenkripsi citra digital dengan memanfaatkan sistem chaos. Enkripsi dilakukan dengan algoritma pseudo one-time pad, yang dalam hal ini barisan kunci dibangkitkan dengan sistem chaos. Sistem chaos membangkitkan bilangan-bilangan pseudo-random. Hasil eksperimen memperlihatkan algoritma dapat mengenkripsi sembarang citra (citra grayscale atau citra berwarna) dengan baik sehingga citra terenkripsi terlihat acak sempurna. Pengujian dengan mengubah sangat kecil pada nilai awal fungsi chaos memperlihatkan keamanan algoritma dari serangan exhaustive attack.

Keywords

Enkripsi citra, pseudo one-time pad, chaos, pseudo-random

1. PENDAHULUAN

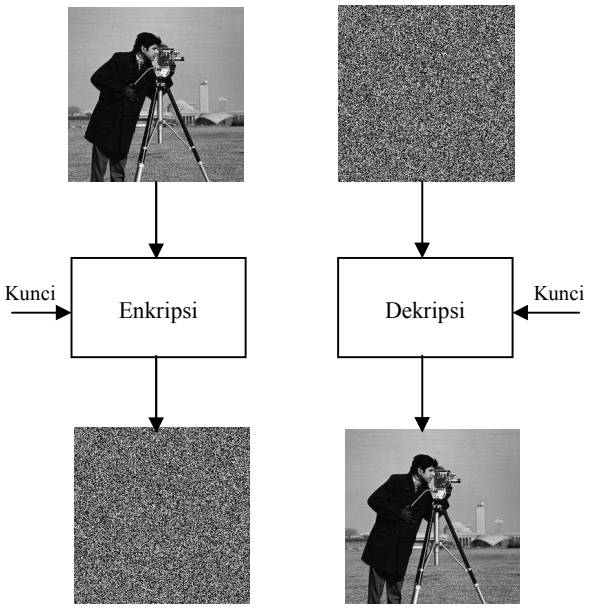
Penyimpanan dan pengiriman citra melalui saluran komunikasi publik rentan terhadap pengaksesan dan penyadapan oleh pihak-pihak yang tidak berhak. Enkripsi citra merupakan teknik yang umum digunakan untuk melindungi citra dari pengaksesan ilegal. Obyektif dari enkripsi citra adalah mentransformasikan citra ke dalam bentuk lain yang sehingga tidak dapat dipersepsi secara visual. Dengan kunci yang sama, citra terenkrpsi dapat dkembalikan menjadi bentuk citra semula (Gambar 1).

Sebenarnya, sembarang algoritma enkripsi tradisionil (DES, AES, Blowfish, RSA, dll) dapat digunakan untuk mengenkripsi citra, namun kebanyakan algoritma tersebut kurang cocok digunakan untuk mengenkripsi citra. Alasannya adalah citra pada umumnya mengandung volume data yang relatif sangat besar dibandingkan dengan data teks, sehingga proses enkripsi citra dengan algoritma-algoritma tradisionil tersebut memerlukan waktu yang lebih lama [1].

Riset untuk menghasilkan algoritma enkripsi citra yang mangkus sekaligus aman terus dilakukan. Kinerja beberapa algoritma enkripsi citra digital dapat ditemukan di dalam [2].

Dibandingkan dengan algoritma *block cipher*, algoritma *stream cipher* relatif lebih mangkus dalam mengenkripsi citra. Hal ini karena *stream cipher* tidak menggunakan skema perulangan (*round*) sebagaimana pada *block cipher*. Selain itu, operasi enkripsi pada *stream cipher* lebih sederhana karena hanya menggunakan operasi XOR atau aritmertika modular.

One-time pad adalah salah satu *stream cipher* klasik yang secara matematis terbukti sempurna aman [6]. Ciphertеksnya tidak mungkin dapat dipecahkan. Keamanan algoritma *one-time pad* terletak pada (1) penggunaan barisan bilangan acak sejati (*trully random*) sebagai kunci enkripsi, (2) panjang kunci sama dengan panjang pesan dan tidak ada perulangan kunci sebagaimana pada pada *Vernam cipher* atau *Vigenere cipher*.



Gambar 1. Enkripsi dan dekripsi citra digital

Sayangnya *one-time pad* tidak dapat diimplementasikan secara praktis sebab pembangkitan bilangan acak sejati tidak dapat diulang kembali di sisi penerima pesan. Oleh karena itu kunci (*pad*) harus dikirim melalui saluran komunikasi yang kedua (misalnya melalui kurir), sayangnya saluran kedua itu umumnya lambat dan ongkosnya mahal.

One-time pad masih dapat diterapkan namun kunci yang berupa barisan bilangan acak diganti dengan barisan bilangan semi-acak (*pseudo-random*) dengan syarat barisan kunci itu tidak boleh berulang (tidak mempunyai periode).

Pembangkit bilangan semi-acak yang tidak mempunyai periode perulangan adalah sistem *chaos*. *Chaos* sudah digunakan di dalam kriptografi [3]. Karakteristik fungsi *chaos* adalah sensitivitasnya terhadap perubahan kecil parameter nilai awal (*sensitive dependence on initial condition*). Sensitivitas ini berarti bahwa perbedaan kecil pada nilai awal fungsi --setelah fungsi diiterasi sejumlah kali-- menghasilkan perbedaan yang sangat besar pada nilai fungsinya.

Selain sifat sensitivitas tersebut, sistem *chaos* tidak mempunyai periode, artinya barisan nilai yang dihasilkannya tidak pernah terulang kembali. Sifat ini penting agar barisan nilai-nilai kunci yang dihasilkan dari sistem *chaos* memenuhi persyaratan algoritma *one-time pad*. Meskipun demikian sistem *chaos* adalah deterministik, artinya nilai-nilai acak yang dihasilkannya dapat dibangkitkan kembali asalkan nilai awal yang digunakan tetap sama. Oleh karena itu versi algoritma enkripsi yang dibahas di sini dinamakan *pseudo one-time pad*.

Penggunaan *chaos* untuk enkripsi citra sudah banyak diteliti. *Review* beberapa teknik enkripsi citra dengan menggunakan skema *chaos* dapat dibaca di dalam [4]. Sistem chaos yang digunakan bervariasi, antara lain *logistic map*, *Baker map*, *Arnold cat map*, dan lain-lain.

Makalah ini mendeksripsikan sebuah algoritma enkripsi sederhana untuk citra digital dengan menggunakan *pseudo one-time pad* berbasiskan pada sistem *chaos Logistic Map*. *Logistic map* dipilih karena komputasinya relatif sederhana sehingga ia cocok digunakan untuk aplikasi enkripsi yang membutuhkan waktu proses yang cepat namun tetap aman secara kriptografis.

Logistic map berperan untuk membangkitkan barisan bilangan semi-acak sebagai kunci *one-time pad*. Barisan bilangan acak ini kemudian dienkripsi dengan *pixel-pixel* citra.

2. *CHAOS* DAN *LOGISTIC MAP*

Logistic map adalah sistem *chaos* yang paling sederhana yang berbentuk persamaan iteratif sebagai berikut:

$$x_{i+1} = r\, x_i\, (1 - x_i) \qquad (1)$$

dengan $0 \le x_i \le 1$, $i = 0, 1, 2, ...$ dan $0 \le r \le 4$. Nilai awal (*seed*) persamaan iterasi adalah x_0. Persamaan (1) bersifat deterministik sebab jika dimasukkan nilai x_0 yang sama maka dihasilkan barisan nilai chaotik (x_i) yang sama pula. Oleh karena itu, pembangkit bilangan acak dengan sistem chaos disebut *pseudo-random generator*.

Sebagaimana disebutkan sebelumnya bahwa properti sistem *chaos* yang paling penting adalah sensitivitasnya pada perubahan kecil nilai awal. Tabel 1 memperlihatkan jumlah iterasi yang diperlukan untuk memperoleh nilai *chaos* yang berbeda lebih besar dari 0.0625 jika nilai awal x_0 yang digunakan diubah dengan galat (*error*) yang sangat kecil [5]. Dari tabel tersebut kita bisa melihat dengan pengubahan sebesar 10^{-30} dibutuhkan 100 kali iterasi untuk memperoleh nilai *chaos* yang berbeda > 0.0625.

Tabel 1. Jumlah iterasi untuk memperoleh nilai *chaos* > 0.0625 dengan pengubahan nilai awal sebesar galat yang diberikan [6].

Galat pada nilai awal	Jumlah iterasi yang diperlukan untuk memperoleh nilai *chaos* > 0.0625
10^{-2}	5
10^{-5}	14
10^{-10}	26
10^{-15}	42
10^{-19}	59
10^{-30}	99

Nilai awal *logistic map* (x_0) di dalam algoritma kriptografi berperan sebagai kunci rahasia. Dengan nilai awal yang tepat sama maka proses dekripsi menghasilkan plainteks semula .

Sayangnya nilai-nilai *chaos* tidak dapat langsung dioperasi-modulokan dengan plainteks karena masih berbentuk bilangan riil antara 0 dan 1. Agar barisan nilai chaotik dapat dipakai untuk enkripsi dan dekripsi dengan *stream cipher*, maka nilai-nilai *chaos* tersebut dikonversi ke nilai *integer*.

Di dalam makalah ini, konversi nilai *chaos* ke *integer* dilakukan dengan menggunakan fungsi pemotongan yang diusulkan di dalam [7]. Caranya adalah dengan mengalikan nilai *chaos* (x) dengan 10 berulangkali sampai ia mencapai panjang angka (*size*) yang diinginkan, selanjutnya potong hasil perkalian tersebut untuk mengambil bagian *integer*-nya saja. Secara matematis fungsi konversi tersebut adalah:

$$T(x, size) = \left\| x * 10^{count} \right\|, x \neq 0 \qquad (2)$$

yang dalam hal ini *count* dimulai dari 1 dan bertambah 1 hingga $x * 10^{count} > 10^{size-1}$. Hasilnya kemudian diambil bagian *integer* saja (dilambangkan dengan pasangan garis ganda pada persamaan 4). Sebagai contoh, misalkan x = 0.003176501 dan *size* = 4, maka dimulai dari *count* = 1 sampai *count* = 6 diperoleh

$$0.003176501 * 10^6 = 3176.501 > 10^3$$

kemudian ambil bagian *integer*-nya dengan

$$\|3176.501\| = 3176$$

Prosedur yang sama dilakukan untuk nilai-nilai chaotik lainnya.

3. *ONE-TIME PAD*

One-time pad (*OTP*) adalah *stream cipher* yang melakukan enkripsi dan dekripsi satu karakter setiap kali. Algoritma ini ditemukan pada tahun 1917 oleh Major Joseph Mauborgne sebagai perbaikan dari Vernam cipher untuk menghasilkan keamanan yang sempurna. Mauborgne mengusulkan penggunaan *one-time pad* (*pad* = kertas bloknot) yang berisi deretan karakter-karakter kunci yang dibangkitkan secara acak. Satu *pad* hanya digunakan sekali (*one-time*) saja untuk mengenkripsi pesan, setelah itu *pad* yang telah digunakan dihancurkan supaya tidak dipakai kembali untuk mengenkripsi pesan yang lain.

Enkripsi dapat dinyatakan sebagai penjumlahan modulo 26 dari satu karakter plainteks dengan satu karakter kunci *one-time pad*:

$$c_i = (p_i + k_i) \bmod 26 \qquad (3)$$

Jika karakter yang digunakan adalah anggota himpunan 256 karakter (seperti karakter dengan pengkodean ASCII), maka persamaan enkripsinya menjadi

$$c_i = (p_i + k_i) \bmod 256 \qquad (4)$$

Setelah pengirim mengenkripsi pesan dengan kunci, ia menghancurkan kunci tersebut. Penerima pesan menggunakan *pad* yang sama untuk mendekripsikan karakter-karakter

cipherteks menjadi karakter-karakter plainteks dengan persamaan:

$$p_i = (c_i - k_i) \bmod 26 \quad (5)$$

untuk alfabet 26-huruf, atau

$$p_i = (c_i - k_i) \bmod 256 \quad (6)$$

untuk alfabet 256-karakter.

Perhatikan bahwa panjang kunci harus sama dengan panjang plainteks, sehingga tidak ada kebutuhan mengulang penggunaan kunci selama proses enkripsi (seperti halnya pada *Vernam cipher*).

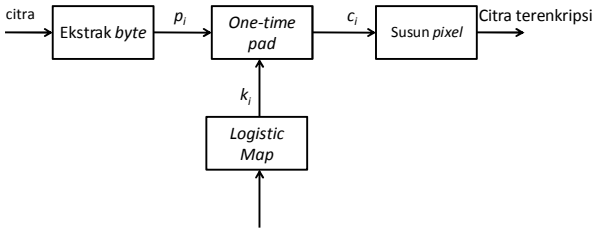
Algoritma *OTP* ini tidak dapat dipecahkan (*unbreakable*) karena dua alasan:

1. Barisan kunci acak yang ditambahkan ke pesan plainteks yang tidak acak menghasilkan cipherteks yang seluruhnya acak. Cipherteks ini tidak mempunyai hubungan statistik dengan plainteks [6].
2. Karena cipherteks tidak mengandung informasi apapun perihal plainteks, maka tidak mungkin ada cara untuk memecahkan cipherteks. Beberapa barisan kunci yang digunakan untuk mendekripsi cipherteks mungkin menghasilkan plainteks yang mempunyai makna, sehingga kriptanalis tidak punya cara untuk menentukan plainteks mana yang benar.

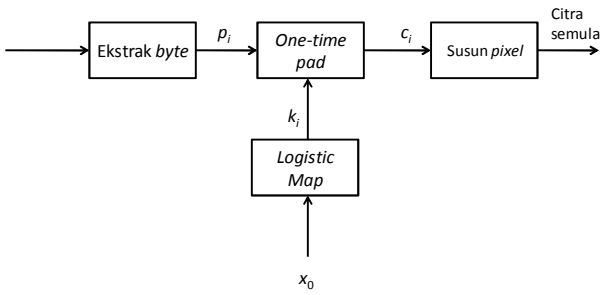
4. USULAN ALGORITMA ENKRIPSI

Algoritma enkripsi sederhana yang diusulkan di dalam makalah ini disebut algoritma *pseudo one-time pad* karena tidak menggunakan barisan bilangan yang ebnar-benar acak sebagai barisan kunci *one-time pad*, tetapi bilangan semi-acak.

Algoritma menerima masukan sembarang citra *bitmap* (baik citra *grayscale* maupun citra berwarna) dan nilai awal *logistic map*. *Pixel-pixel* di dalam citra *bitmap* berukuran sejumlah *byte*. Pada citra 8-bit satu *pixel* berukuran 1 *byte* (8 bit), sedangkan pada citra 24-bit satu *pixel* berukuran 3 *byte* (24 bit). Pada citra berwarna setiap *pixel* disusun oleh komponen warna *R* (*red*), *G* (*green*), dan *B* (*blue*), masing-masing panjangnya 1 *byte*. Operasi enkripsi dan dekripsi dilakukan pada setiap *byte* dengan melakukan operasi penjumlahan modulo *byte* tersebut dengan setiap elemen kunci *one-time pad*. Gambar 2 dan Gambar 3 memperlihatkan diagram masing-masing skema enkripsi dan dekripsi.



Gambar 2. Skema enkripsi dengan *Pseudo One-Time Pad*



Gambar 3. Skema dekripsi dengan *Pseudo One-Time Pad*

Algoritma enkripsi dan dekripsi *pseudo one-time pad* dapat diringkas dalam urutan langkah-langkah (*step*) sebagai berikut:

(a) Enkripsi

Masukan: citra, x_0

Keluaran: citra terenkripsi

Step 1: Baca *pixel-pixel* citra dan simpan di dalam sebuah matriks. Jika citra tersebut citra 24-bit, maka setiap komponen *R*, *G*, dan *B* disimpan di dalam tiga matriks berbeda.

Step 2: Ekstraksi *byte* setiap *pixel* citra, misalkan *byte-byte* tersebut adalah $p_1, p_2, \ldots, p_n$.

Step 3: Iterasikan *logistic map* dengan nilai awal x_0 (kunci rahasia). Konversikan setiap nilai *chaotic* menjadi *integer* dengan persamaan (2). Misalkan barisan *integer* yang dihasilkan adalah $k_1, k_2, \ldots, k_n$.

Step 4: Enkripsi setiap p_i dengan k_i menggunakan persamaan (4). Hasil enkripsi adalah $c_1, c_2, \ldots, c_n$.

Step 5: Letakkan kembali *byte-byte* hasil enkripsi pada posisi *pixel* semula. Hasil dari langkah terakhir ini adalah citra terenkripsi.

(b) Dekripsi

Masukan: citra terenkripsi, x_0

Keluaran: citra semula

Step 1: Baca *pixel-pixel* citra dan simpan di dalam sebuah matriks. Jika citra tersebut citra 24-bit, maka setiap komponen *R*, *G*, dan *B* disimpan di dalam tiga matriks berbeda.

Step 2: Ekstraksi *byte* setiap *pixel* citra, misalkan *byte-byte* tersebut adalah $p_1, p_2, \ldots, p_n$.

Step 3: Iterasikan *logistic map* dengan nilai awal x_0 (kunci rahasia). Konversikan setiap nilai *chaotic* menjadi *integer* dengan persamaan (2). Misalkan barisan *integer* yang dihasilkan adalah $k_1, k_2, \ldots, k_n$.

Step 4: Enkripsi setiap p_i dengan k_i menggunakan persamaan (6). Hasil enkripsi adalah $c_1, c_2, \ldots, c_n$.

Step 5: Letakkan kembali *byte-byte* hasil enkripsi pada posisi *pixel* semula. Hasil dari langkah terakhir ini adalah citra semula.

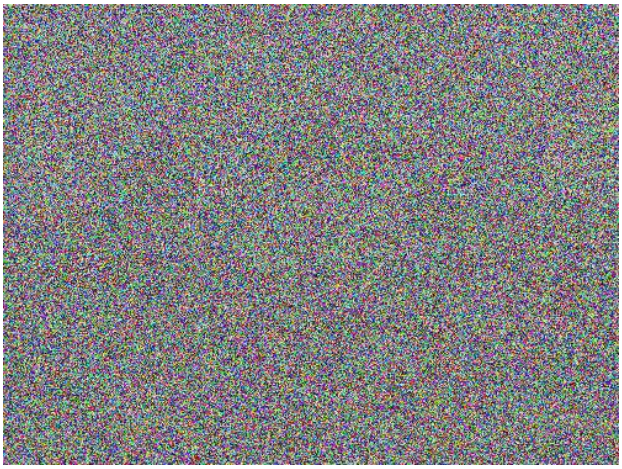
5. HASIL-HASIL EKSPERIMEN

Untuk mengetahui kinerja algoritma *pseudo one-time pad*, maka algoritma tersebut diprogram dengan menggunakan kakas Matlab. Pengujian dilakukan pada sejumlah citra, baik citra *grayscale* maupun citra berwarna. Semua citra dapat dienkripsi menjadi citra terenkripsi (*encrypted-image*) dan dapat didekripsi kembali menjadi citra semula. Kunci enkripsi adalah parameter nilai awal *logistic map*, yaitu $x_0 = 0.625$.

(a) Enkripsi citra berwarna

Gambar 4 memperlihatkan contoh hasil enkripsi untuk citra berwarna (citra 'Gedung Sate', ukuran 500 × 374, kedalaman warna 24-bit). Dekripsi menghasilkan citra yang tepat sama dengan citra 'Gedung Sate' semula.

(a) Citra 'Gedung Sate'

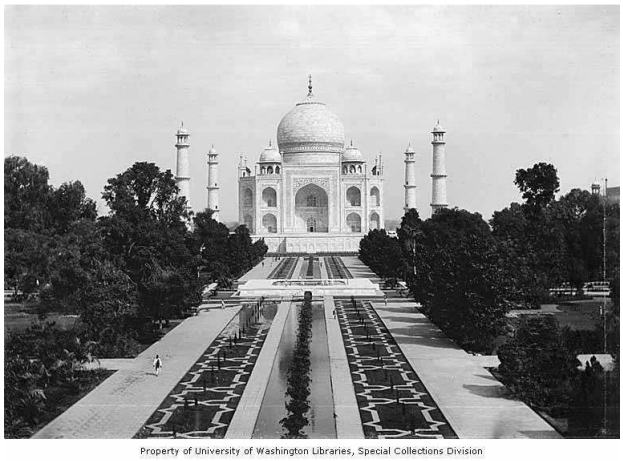


(b) Citra 'Gedung Sate' terenkripsi

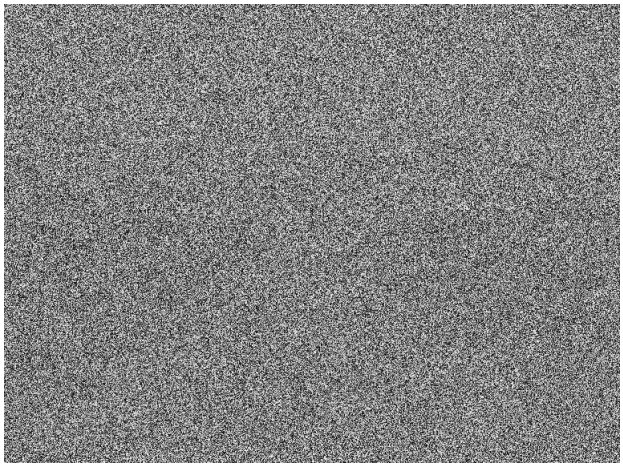
Gambar 4. Enkripsi citra berwarna ('Gedung Sate')

(b) Enkripsi citra *graysacle*

Gambar 5 memperlihatkan contoh hasil enkripsi untuk citra *grayscale* (citra 'Taj Mahal', ukuran 768 × 573, kedalaman warna 8-bit). Dekripsi menghasilkan citra yang tepat sama dengan citra 'Taj Mahal' semula.



(a) Citra 'Taj Mahal'



(b) Citra 'Taj Mahal' terenkripsi

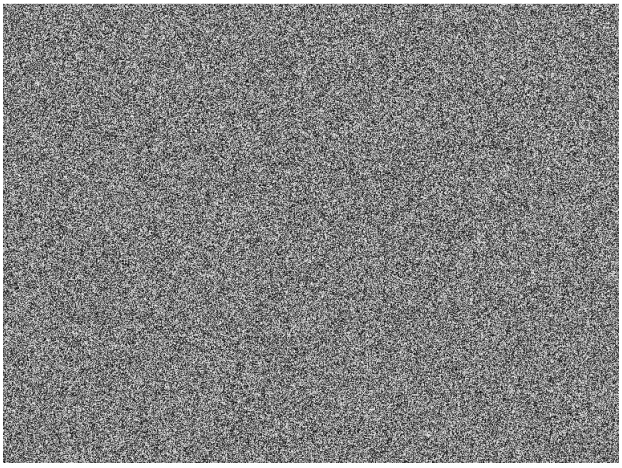
Gambar 5. Enkripsi citra *grayscale* ('Taj Mahal')

(c) Analisis perubahan nilai awal *logistic map*

Algoritma *pseudo one-time pad* menggunakan sistem *chaos* sebagai properti keamanan yang penting. Jika nilai awal fungsi diubah sedikit saja, nilai-nilai acak yang dihasilkannya berubah secara signifikan setelah *logistic map* diiterasi sejumlah kali. Hal ini sangat bagus untuk keamanan sebab lawan yang mencoba menemukan kunci akan frustasi karena perbedaan nilai awal fungsi *chaos* yang sekecil apapun tidak menghasilkan citra semula.

Pada eksperimen ini nilai awal *logistic map* diubah sebesar Δ sehingga menjadi $x_0 + \Delta$, kemudian citra didekripsi dengan kunci $x_0 + \Delta$. Misalkan $\Delta = 10^{-10}$ sehingga nilai awal *logistic map* adalah 0.625000001. Gambar 6 memperlihatkan citra 'Taj Mahal' hasil dekripsi yang ternyata tetap teracak (tidak kembali menjadi citra semula). Perubahan kecil nilai awal *chaos* membuat nilai chaotik yang dihasilkan berbeda signifikan. Karena nilai-nilai ini dipakai sebagai kunci *one-time pad* maka hasilnya adalah hasil dekripsi yang tidak benar dan tetap teracak.

Hasil eksperimen ini membuktikan bahwa karakteristik *chaos* yang sensitif terhadap nilai awal memang memberikan keamanan yang bagus dari serangan *exhaustive attack*. Eksperimen ini juga menyiratkan bahwa perubahan sangat kecil pada kunci menyebabkan hasil dekripsi salah, apalagi jika kunci dekripsi yang diberikan berbeda jauh nilainya dengan kunci yang sebenarnya.



(a) Citra Taj Mahal hasil dekripsi (dengan pengubahan nilai awal *logistic map* sebesar $\Delta = 10^{-10}$)

(b) Citra Taj Mahal hasil dekripsi dengan nilai awal *logistic map* yang benar

Gambar 6. Hasil ekperimen dekripsi dengan pengubahan nilai awal *logistic map* sebesar $\Delta = 10^{-10}$.

6. KESIMPULAN

Di dalam makalah ini telah dipresentasikan algoritma enkripsi citra dengan algoritma *pseudo one-time pad* berbasiskan pada sistem *chaos*. Fungsi *chaos* yang digunakan adalah *logistic map*. Nilai awal *logistic map* berperan sebagai kunci rahasia.

Hasil eksperimen memperlihatkan bahwa algoritma enkripsi ini dapat mengenkripsi citra dengan baik dan mendekripsinya kembali tepat sama seperti citra semula. Eksperimen perubahan parameter nilai *chaos* memperlihatkan bahwa algoritma ini aman dari serangan *exhaustive attack*.

REFERENSI

[1] Lala Krikor, Sama Baba, Thawar Arif, Zyad Shaaban, *Image Encryption Using DCT and Stream Cipher*, European Journal of Scientific Research, Vol. 32, No. 1 (2009), pp 47-57

[2] Jolly Shah, Vikas Saxena, *Performance Study on Image Encryption Schemes*, IJSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011.

[3] Krish M. Roskin dan Jonathan B. Casper, *From Chaos to Cryptography*, http://www.gaianxaos.com/pdf/unsorted, diakses tanggal 9 November 2011 pukul 14.00.

[4] Monisha Sharma, Manoj Kumar Kowar, *Image Encryption Technique Using Chaotic Schemes: A Review*, International Journal of Engineering, Science, and Technology Vol 2 (6) 2010.

[5] Ranjan Bose dan Amitabha Banerjee, *Implementing Symmetric Cryptography Using Chaos Function,* Indian Institute of Technology.

[6] William Stalling, *Cryptography and Network Security, Principle and Practice 3rd Edition*, Pearson Education, Inc., 2003.

[7] James Lampton, *Chaos Cryptography: Protecting data Using Chaos,* Missisippi School for Mathematics and Science.