Enkripsi Gambar Parsial dengan Kombinasi Metode *Stream Cipher RC4* dan *Chaotic Function*

I Putu Arya Dharmaadi[1], Ari M. Barmawi[2], Gandeva Bayu S.[3]

Fakultas Informatika, Institut Teknologi Telkom, Bandung
aryadharmaadi@yahoo.com[1], mbarmawi@melsa.net.id[2], gandeva.bayu.s@gmail.com[3]

Abstrak

Keamanan data merupakan masalah yang penting dalam pertukaran data, baik untuk data berupa gambar maupun teks. Khusus untuk mengamankan data berupa gambar diperlukan proses enkripsi untuk seluruh gambar akan membutuhkan waktu yang cukup panjang karena jumlah data yang harus dienkriupsi cukup banyak. Dalam rangka menghemat waktu, maka gambar maupun foto tersebut perlu dienkripsi secara parsial sedemikian rupa sehingga informasi yang dianggap penting saja yang tidak dapat dimanfaatkan oleh orang-orang yang tidak bertanggungjawab. Sistem enkripsi yang akan digunakan untuk melakukan enkripsi secara parsial ini adalah *stream cipher* RC4 yang dikombinasikan diperkuat dengan fungsi *Chaotic*. Algoritma RC4 dipilih karena algoritma tersebut sederhana dan membutuhkan waktu yang relatif pendek. Walaupun demikian, metode ini memerlukan kunci enkripsi yang cukup panjang sesuai dengan ukuran bagian gambar atau foto yang akan dienkripsi. *User* tentu merasa kesulitan dalam mengingat kunci yang terlalu panjang tersebut. Apabila kunci yang digunakan terlalu pendek, maka akan diperlukan pengulangan kunci agar dihasilkan kunci yang memiliki ukuran sesuai dengan ukuran gambar yang dienkripsi. Pengulangan kunci ini akan memperbesar kemungkinan kunci dapat dipecahkan. Untuk mengatasi hal ini pada Tugas Akhir ini dirancang sistem enkripsi *stream cipher* RC4 yang akan diperkuat dengan *Chaotic function*. Fungsi ini akan bekerja sebagai pembangkit bilangan acak secara otomatis sesuai panjang yang kunci yang diperlukan, dengan masukan kunci awal yang pendek. Walaupun kunci masukan pendek, tetapi *chaotic function* akan membangkitkan kunci *keystream* yang tingkat keacakannya lebih tinggi sehingga keamanan sistem enkripsi RC4 semakin tinggi.

Kata kunci : *stream cipher*, RC4, *chaotic function*, enkripsi.

1. Pendahuluan

Dewasa ini terjadi banyak sekali kejahatan di dunia digital. Salah satunya adalah penyalinan data oleh orang yang tidak berhak. Sebagai contoh, seseorang mengirimkan gambar hasil scan buku rekening tabungan ke seorang temannya melalui *email*. Pengirim maupun temannya yang sebagai penerima tidak akan sadar bahwa gambar hasil scan buku tabungan tersebut telah disadap oleh pihak-pihak yang tidak bertanggungjawab untuk mendapatkan nomor rekeningnya yang kemudian akan digunakan untuk tindak kejahatan.

Untuk mencgah terjadinya hal tersebut, maka gambar maupun foto yang sifatnya penting dan rahasia perlu diamankan agar orang yang menyadap di tengah jalan tidak bisa memahami gambar maupun foto tersebut. Pengamanan gambar tersebut dapat dilakukan dengan cara mengenkripsi gambar tersebut. Mengingat enkripsi gambar secara keseluruhan akan membutuhkan waktu yang cukup lama, maka diperlukan cara untuk mengurangi waktu enkripsi dan dekripsinya. Salah satu cara untuk mengurangi waktu enkripsi tersebut adalah dengan mengenkripsi bagian yang penting saja atau dengan kata lain adalah melakukan enkripsi gambar sebagian. Sistem yang akan dibuat ini akan memudahkan user untuk melakukan enkripsi maupun dekripsi. *User* tidak perlu mengenkripsi satu gambar penuh, namun cukup meng-"*crop*" bagian pada gambar yang dianggap perlu dirahasiakan.

Untuk mengenkripsi gambar biasanya digunakan metode *stream cipher* karena metode ini dapat mengenkripsi dan mendekripsi dengan waktu yang relative cepat. Salah satu jenis *stream cipher* adalah RC4. Algoritma ini sangat sederhana dan mudah dipahami.

Walaupun demikian, algoritma RC4 memiliki kelemahan yaitu data yang dienkripsi tidak sepenuhnya aman. Hal ini dipengaruhi oleh panjang kunci yang digunakan untuk mengenkripsi data. Untuk kunci yang sangat panjang, tentu akan menjadi masalah bagi *user* untuk mengingat dan menyimpannya. Jika kunci yang digunakan terlalu pendek, maka akan terjadi perulangan kunci agar panjang kunci sesuai dengan *plaintext*. Pengulangan ini akan mengurangi tingkat keamanan system kripto yang digunakan.

Permasalahan yang akan dibahas pada jurnal ini adalah sebagai berikut:

a. Bagaimana cara mengatasi kelemahan panjang kunci yang terlalu pendek dari metode *stream cipher* RC4?
b. Bagaimana tingkat keamanan sistem enkripsi yang dibangun jika dibandingkan dengan metode konvensional *stream cipher* RC4?
c. Bagaimana performansi sistem enkripsi yang dibangun, dilihat dari segi waktu pemrosesan?

Berdasarkan rumusan masalah di atas, maka tujuan dari Tugas Akhir ini adalah sebagai berikut:

a. Meningkatkan keamanan pengiriman data berupa gambar atau foto yang menggunakan metode *stream cipher* RC4.
b. Membandingkan dan menganalisis tingkat keamanan antara sistem enkripsi yang dibangun dengan metode konvensional *stream cipher* RC4.
c. Menganalisis kinerja enkripsi dengan menghitung waktu pemrosesan.

2. RC4 dan *Chaotic Function*

RC4 dan *Chaotic Function* merupakan dua algoritma utama yang akan digunakan untuk membangun sistem enkripsi yang dirancang pada jurnal ini. *Chaotic function* digunakan untuk memperkuat masukkan kunci yang akan digunakan untuk mengenkripsi atau mendekripsi oleh algoritma *stream cipher* RC4.

2.1 RC4

RC4 merupakan algoritma yang sederhana dan mudah diimplementasikan yang didesain oleh Ron Rivest pada tahun 1987 dari Laboratorium RSA. RC4 merupakan protokol enkripsi standar yang sering digunakan dalam berbagai standar keamanan, seperti standarisasi *802.11 wireless network* pada protokol WEP (*Wired Equivalent Privacy*) dan WPA (*WiFi Protected Access*) [7]. RC4 juga diimplementasikan pada protokol SSL (*Secure Socket Layer*), sebuah protokol untuk memproteksi trafik internet. Alasan dibalik kesuksesan RC4 adalah kecepatan dan kesederhanaannya sehingga mudah untuk mengembangkan implementasi yang efisien ke dalam perangkat keras maupun perangkat lunak.

Algoritma RC4 akan membangkitkan aliran kunci (*keystream*) dari sebuah *keystream generator*. Pembangkit kunci ini terdiri dari status internal yang terdiri dari dua bagian, yaitu:

1. Array *S* yang terdiri dari permutasi angka 0 sampai 255. Permutasi merupakan fungsi dari kunci *U* dengan panjang variabel[7].
2. Dua buah variabel, yaitu *i* dan *j*, yang berfungsi sebagai pencacah indeks[7].

Ada dua tahapan algoritma pada RC4 yang akan dilalui untuk membangkitkan aliran kunci, yaitu *Key-Scheduling Algorithm* (KSA) dan *Pseudo-Random Generator Algorithm* (PRGA). Pada KSA, akan terjadi inisialisasi permutasi berdasarkan kunci *U* pada array *S* dengan rentang nilai antara 0 sampai 255. Rentang ini dipilih karena RC4 mengenkripsi pada mode *byte* ($255 = 2^8$ dan 8 bit = 1 *byte*). Artinya, panjang kunci maksimal yang disimpan pada array *U* adalah 256 karakter (256 *byte* atau 2048 bit). Berikut adalah algoritma KSA[7]:

RC4 Key Scheduling Algorithm

```
1: for i=0 to 255 do
2:    S[i]← i
3: end for
4: j ← 0
5: for i=0 to 255 do
6:    j ← ( j + S[i] + U [i] ) mod 256
7:    swap(S[i],S[j])
8: end for
```

Selanjutnya, hasil dari array *S* yang telah melalui KSA akan diproses lagi pada PRGA. Pada tahapan ini, akan terjadi modifikasi *state* dan *output* sebuah *byte* dari aliran kunci. Aliran kunci dipilih dengan mengambil nilai *S[i]* dan *S[j]* dan menjumlahkannya dalam modulo 256. Hasil penjumlahan akan menjadi indeks sehingga *S[indeks]* menjadi aliran kunci *K* yang kemudian digunakan untuk mengenkripsi plaintext ke-*idx*. Berikut adalah algoritma PRGA[7]:

RC4 Pseudo-Random Generator Algorithm

```
1: i ← 0
2: j ← 0
3: loop
4:    i ← i + 1 mod 256
5:    j ← j + S[i] mod 256
6:    swap(S[i],S[j])
7:    keystream word K ← S[S[i] + S[j] mod
256]
8: end for
```

RC4 memiliki kelemahan, yaitu ada kemungkinan nilai-nilai di dalam array *S* ada yang sama. Hal ini disebabkan karena karakter-karakter kunci di-copy berulang-ulang untuk memenuhi 256 *byte* kunci *U* jika panjang kunci *U* kurang dari 256 *byte*[3]. Sangat jarang ditemukan user yang menggunakan kunci tepat 256 byte karena sangat sulit mencari kombinasinya dan juga sulit untuk mengingatnya.

2.2 *Chaotic Function*

Teori *chaos* berasal dari teori sistem yang memperlihatkan kemunculan yang tidak teratur atau digunakan juga untuk menjelaskan kemunculan data acak. Sampai saat ini, teori *chaos* tersebut banyak diimplementasikan dalam kriptografi. Sistem chaos ini sangat berguna karena bisa menghasilkan bilangan semi acak yang tidak memiliki periode perulangan. Sistem *chaos* memiliki sifat yang sangat berharga yang bisa diterapkan dalam kriptografi, yaitu peka pada perubahan kecil pada kondisi awal sistem. Salah satu dari dua prinsip Shannon yang dijadikan panduan dalam perancangan algoritma kriptografi adalah difusi (*diffusion*), yang artinya adalah menyebarkan pengaruh 1 bit (atau digit) *plaintext* ke seluruh bit (digit) *ciphertext* dengan maksud untuk menyembunyikan hubungan statistik antara *plaintext* dengan *ciphertext*[3]. Bisa juga artinya untuk menyebarkan 1 bit kunci (*key*) ke seluruh bit *ciphertext*. Apabila bit kunci diubah sedikit saja, maka *ciphertext* yang dihasilkan akan berbeda secara signifikan. Hal ini akan menyebabkan kriptanalisis kesulitan mendekripsikan *ciphertext* dengan menganalisis kunci dan mencari pola hubungan antara *plaintext* dengan *ciphertext*.

Meskipun sedemikian acak, sistem *chaos* tetap bersifat deterministik, yang berarti nilai-nilai acak yang dihasilkan bisa dibangkitkan kembali untuk menghasilkan nilai-nilai acak yang sama, dengan syarat nilai awal yang digunakan juga sama persis. Salah satu sistem chaos yang paling sederhana dan banyak digunakan adalah fungsi logistik (*logistic map*). *Logistic map* berupa persamaan iteratif yang dijabarkan sebagai berikut:

$$x_{i+1} = r\, x_i\, (1 - x_i) \qquad (2.1)\ [3]$$

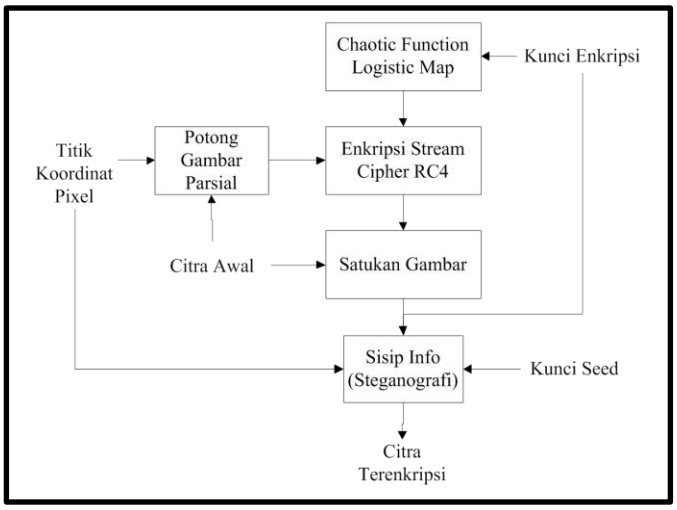
Pada persamaan di atas, variabel x_i akan menjadi nilai awal. Konstanta r berupa laju pertumbuhan dengan rentang nilai $0 \leq r \leq 4$. Ketika $r = 4$, iterasi bergantung sepenuhnya pada nilai awal x_i dan nilai-nilai yang dihasilkan muncul acak meskipun sistem ini deterministik[3]. Artinya, konstanta yang tepat agar nilai-nilai *chaos* benar-benar acak adalah konstanta dengan nilai 4.

Agar barisan nilai chaotik dapat dipakai untuk enkripsi dan dekripsi dengan stream cipher, maka nilai-nilai chaos tersebut dikonversi ke nilai integer. Ada beberapa teknik konversi dari nilai-nilai chaotic menjadi nilai integer. Salah satunya, nilai chaos dikalikan dengan 10 berulangkali sampai ia mencapai panjang angka (size) yang diinginkan, selanjutnya potong hasil perkalian tersebut untuk mengambil bagian integer-nya saja[3].

3. Perancangan Sistem

Adapun prosedur sistem enkripsi yang akan dibangun sebagai berikut.

1. Pengguna memasukkan gambar atau foto yang akan dienkripsi dan kunci enkripsi.
2. Setelah gambar atau foto yang dimasukkan muncul, maka pengguna diminta untuk menge-"*crop*" bidang gambar yang akan dienkripsi.
3. Sistem akan mengolah kunci enkripsi dengan algoritma *chaotic function logistic map* untuk menghasilkan kunci yang teracak sempurna sepanjang 256 *byte*.
4. Kunci acak tersebut kemudian digunakan oleh algoritma *stream cipher* RC4 untuk mengenkripsi bidang gambar yang telah dipilih oleh pengguna.
5. Terakhir, sistem akan mengeluarkan gambar yang telah terenkripsi parsial.



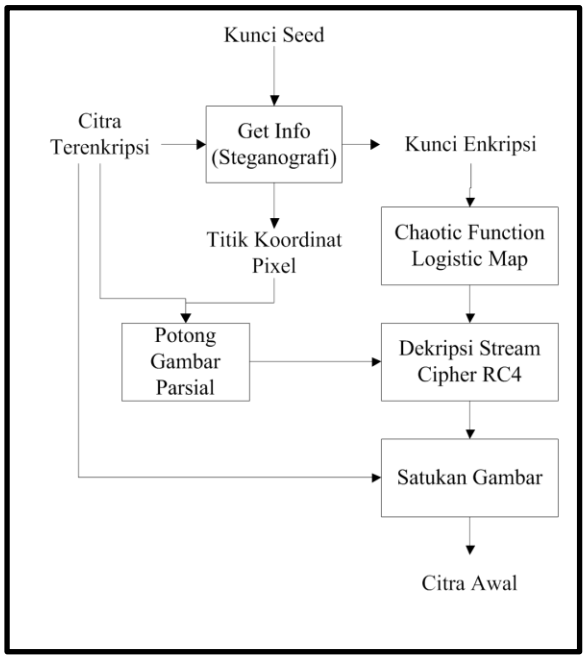
Gambar 1 Skema proses enkripsi

Sedangkan untuk prosedur sistem dekripsi yang akan dibangun sebagai berikut.

1. Pengguna memasukkan gambar atau foto yang memiliki bidang gambar terenkripsi beserta kunci dekripsi. Kunci dekripsi sama dengan kunci enkripsi karena sistem ini adalah kriptografi kunci simetri.
2. Sistem akan menemukan titik koordinat *pixel* yang terenkripsi pada gambar melalui *data hiding* pada LSB.
3. Sistem akan mengolah kunci dekripsi dengan algoritma *chaotic function logistic map* untuk menghasilkan kunci yang teracak sempurna sepanjang 256 *byte*.
4. Kunci acak tersebut kemudian digunakan oleh algoritma *stream cipher* RC4 untuk

mendekripsi bidang gambar yang terenkripsi.

5. Terakhir, sistem akan menghasilkan gambar yang seperti semula.



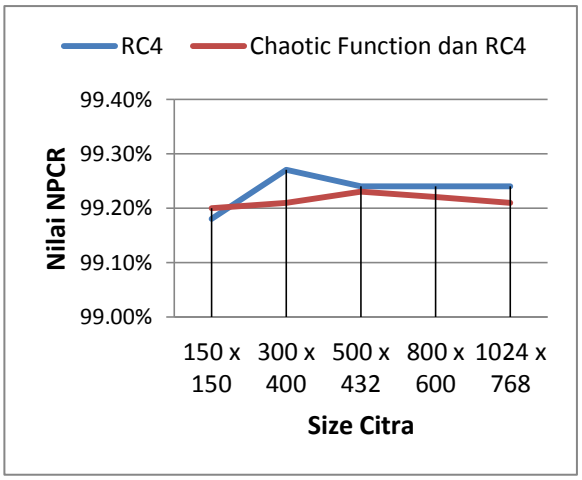
Gambar 2 Skema proses dekripsi

4. Pengujian dan Analisis

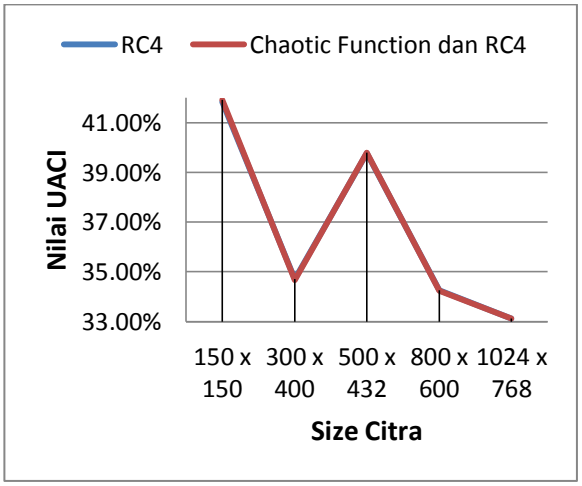
Pengujian yang pertama dilakukan untuk melihat tingkat keteracakan keystream yang dihasilkan oleh sistem enkripsi ini. Caranya dengan menghitung *P-value* melalui *NIST Statistical Test*. Hasilnya bisa dilihat di gambar 3.

Berdasarkan grafik yang dihasilkan pada gambar tersebut, bisa dilihat bahwa nilai-nilai yang dihasilkan oleh *Statistical Test* untuk sistem enkripsi *Chaos* dan RC4 lebih besar dari 0.01. Artinya, sistem enkripsi *Chaos* dan RC4 lolos dari 15 macam *Statistical Test* yang dikeluarkan oleh NIST sehingga deretan bit-bit yang dihasilkan oleh sistem enkripsi ini tergolong *random*[6]. Kemudian dengan menggunakan kunci "1991", maka rata-rata *P-value* sistem enkripsi *Chaotic Function* dan RC4 lebih tinggi 0.0537 jika dibandingkan dengan sistem enkripsi konvensional.

Selanjutnya dilakukan pengujian yang kedua yaitu menguji statistik data citra asli dengan data citra terenkripsi. Caranya adalah dengan menggunakan parameter NPCR (*Number of Pixel Change Rate*) dan UACI (*Unified Average Changing Intensity*).

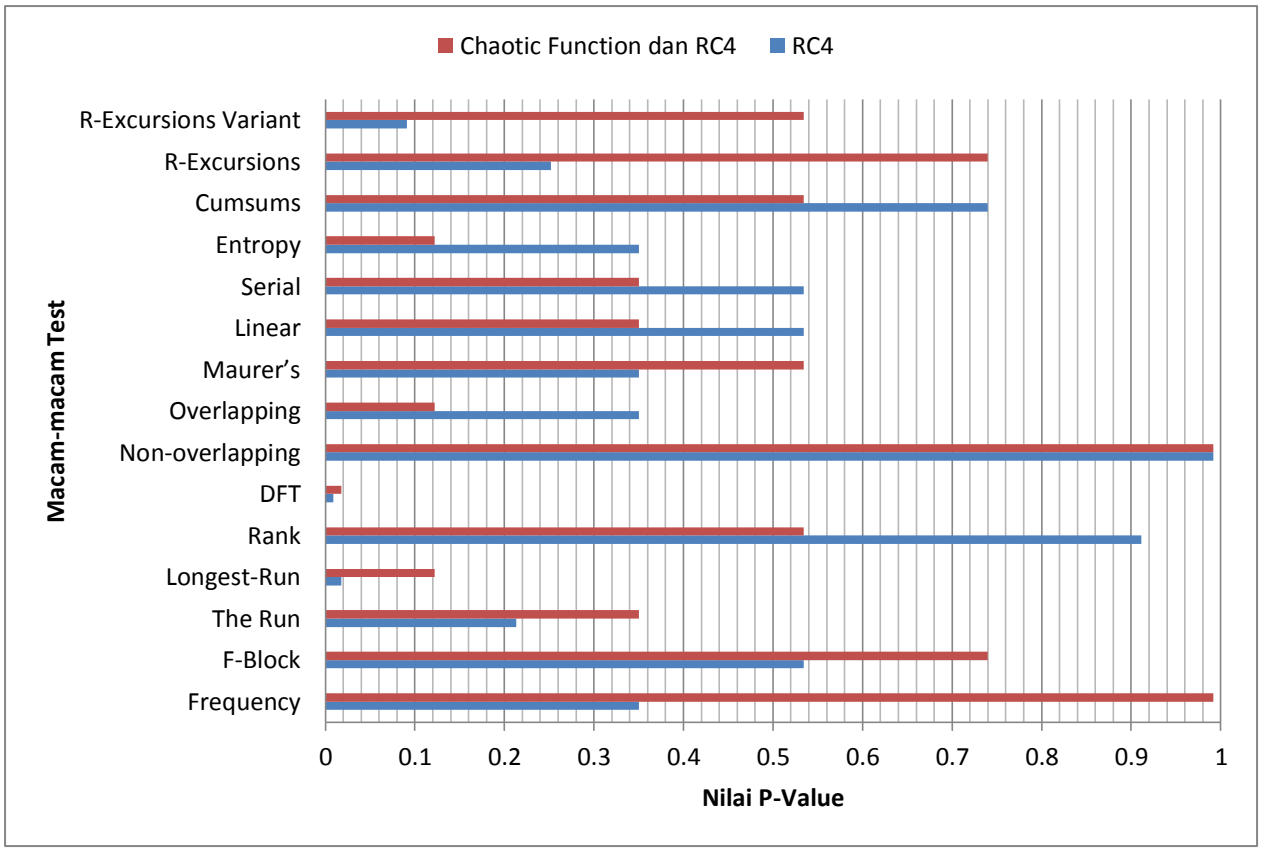


Gambar 3 Grafik perbandingan nilai NPCR



Gambar 4 Grafik perbandingan nilai UACI

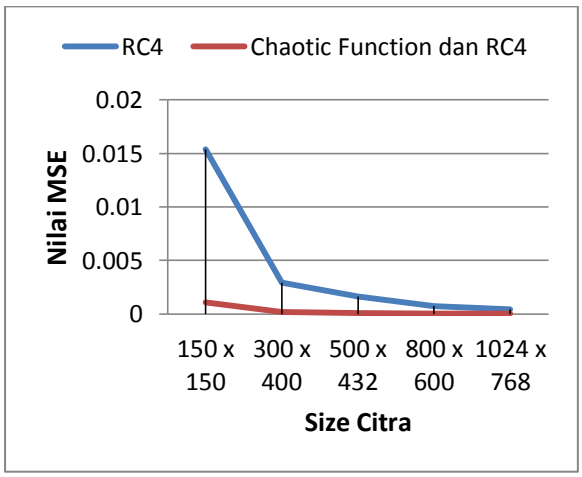
Nilai NPCR dan UACI berkisar dari 0 – 100%. Ketika NPCR dari algoritma enkripsi mencapai nilai terendah, yakni 0%, maka artinya citra asli sama persis dengan citra yang terenkripsi. Namun, ketika sistem enkripsi menghasilkan NPCR 100%, maka keseluruhan *pixel* dari citra awal berbeda dengan citra terenkripsi. Sedangkan nilai UACI menggambarkan seberapa besar perubahan setiap *pixel*, antara citra asli dengan citra terenkripsi, terhadap nilai maksimal yang mungkin dari *pixel* tersebut. Jika hasil UACI di atas 30%, maka rata-rata nilai *pixel* yang asli berbeda cukup jauh dengan *pixel* yang terenkripsi. Hasil NPCR>90% dan UACI>30% ini akan menyulitkan kriptanalisis dalam mencari hubungan statistik antara citra asli dengan citra terenkripsi. Namun, biasanya sangat jarang ada sistem enkripsi yang bisa mencapai NPCR atau UACI 100%. Dengan rata-rata nilai NPCR dan UACI dari gabungan algoritma enkripsi *Chaotic Function* dan *RC4* ini sebesar 99.21% dan 36.74%,



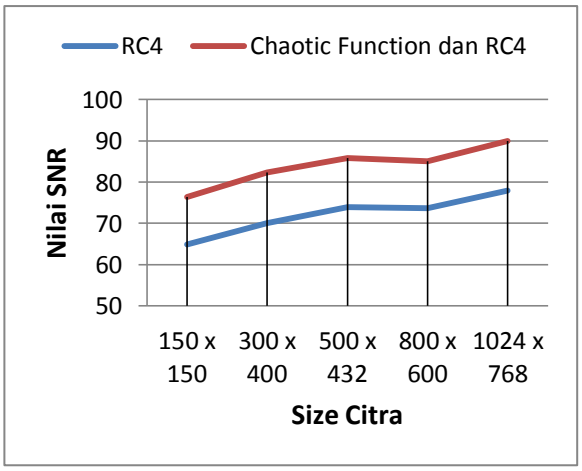
Gambar 5 Grafik perbandingan nilai P-value dengan nilai "1991"

dapat disimpulkan sistem enkripsi yang dibangun ini tahan terhadap *differential attacks.*

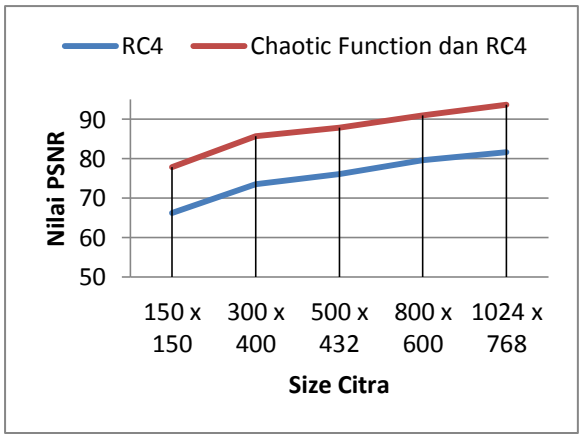
Pengujian yang selanjutnya adalah menguji kualitas citra yang dihasilkan. Parameter yang akan digunakan adalah MSE (*Mean Square Error*), SNR (*Signal to Noise Ratio*), dan PSNR (*Peak Signal to Noise Ratio*).



Gambar 6 Grafik perbandingan nilai MSE



Gambar 7 Grafik perbandingan nilai SNR



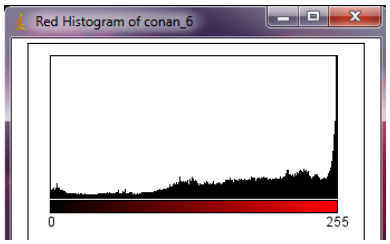
Gambar 8 Grafik perbandingan nilai PSNR

Berdasarkan hasil pengujian di atas, terdapat hubungan antara ukuran citra dengan nilai MSE. Semakin besar ukuran citra, maka MSE dari citra tersebut semakin kecil. Selain itu, nilai SNR dan PSNR juga menunjukkan kualitas citra. Semakin tinggi nilainya, maka citra tersebut semakin bagus kualitasnya terhadap citra aslinya. Sesungguhnya, yang menyebabkan adanya perbedaan pada citra asli dengan citra yang telah didekripsi adalah penggunaan teknik steganografi dalam penyisipan kunci enkripsi dan koordinat *pixel-pixel* yang dienkripsi. Jadi, semakin besar ukuran citra, maka akan menurunkan nilai perbandingan antara pixel yang digunakan untuk steganografi dengan jumlah total pixel. Pada akhirnya, perbandingan ini akan berbanding lurus dengan nilai MSE.

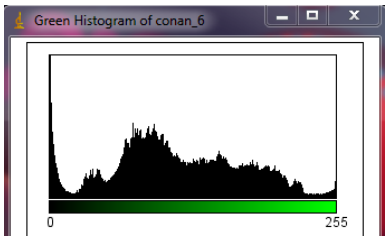
Pada sistem enkripsi RC4, nilai-nilai MSE pada sistem enkripsi ini cukup tinggi. Penyebab utamanya adalah penggunaan kunci enkripsi sepanjang 2048 bit. Kunci sepanjang ini akan disisipkan pada citra sehingga otomatis akan mengurangi kualitas dari citra tersebut. Hal ini bisa dilihat pada citra *conan_shinichi.bmp* yang memiliki nilai MSE lebih dari 1%. Kesimpulannya, dengan sistem enkripsi ini, kualitas citra akan menurun.

Sedangkan pada sistem enkripsi *chaotic function* dan RC4, nilai-nilai MSE pada sistem enkripsi ini jauh lebih kecil dibandingkan dengan sistem enkripsi yang sebelumnya. Rata-rata nilai MSE berkurang di atas 90%. Hal ini disebabkan oleh penggunaan panjang kunci yang jauh lebih kecil dibandingkan dengan sistem yang sebelumnya, yaitu hanya 64 bit, sehingga nilai MSE bisa bernilai di bawah 1 %. Walaupun sama-sama menurunkan kualitas citra, namun dengan sistem enkripsi ini penurunan kualitas bisa ditekan. Dengan rata-rata nilai MSE sistem enkripsi *Chaotic Function* dan RC4 yang kurang dari 1%, maka bisa dikatakan bahwa sistem enkripsi ini bisa menghasilkan citra yang terdekripsi dengan kualitas yang hampir sama dengan citra awal. Kesimpulannya, sistem enkripsi *Chaotic Function* dan RC4 jauh lebih baik dibandingkan dengan sistem enkripsi RC4.

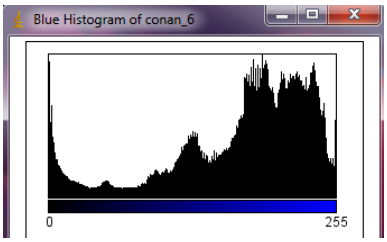
Pengujian yang keempat adalah histogram citra. Jika setelah dienkripsi, citra menghasilkan histogram yang *flat*, maka hal mengindikasikan tidak adanya hubungan statistik antara *pixel* citra original dengan *pixel* citra terenkripsi, sehingga serangan dengan menggunakan analisis statistik menjadi sangat sulit dilakukan. Harapannya adalah persebaran histogram citra setelah dienkripsi tersebar merata[2].



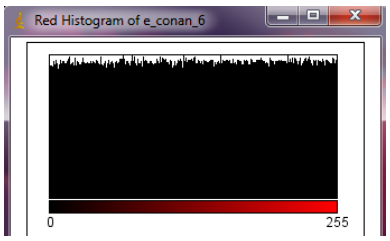
(a)



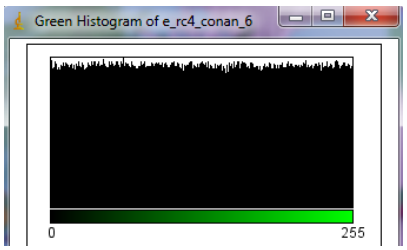
(b)



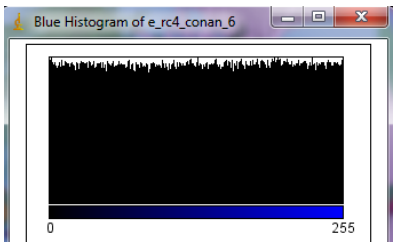
(c)



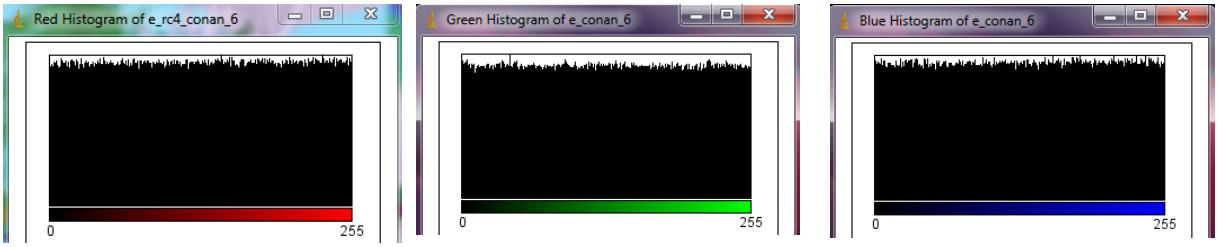
(d)



€



(f)



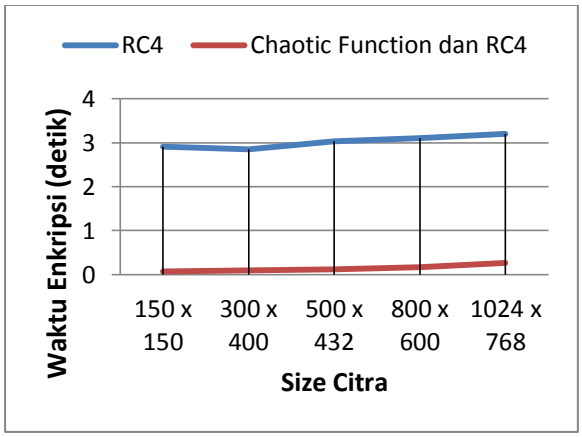
(g) *(h)* *(i)*

Gambar 9 Histogram citra conan_6.bmp
(a),(b),(c) sebelum dienkripsi
(d),(e),(f) setelah dienkripsi dengan algoritma RC4
(g),(h),(i) dengan algoritma Chaotic Function dan RC4

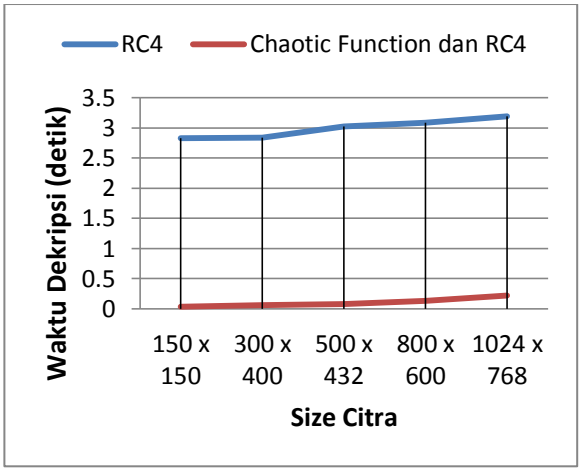
Berdasarkan hasil histogram citra sebelum dan sesudah enkripsi pada gambar di atas, terlihat perubahan yang cukup drastis. Awalnya, histogram terlihat tidak merata dengan mayoritas *pixel* berada pada intensitas yang tinggi pada masing-masing warna. Setelah dilakukan proses enkripsi, maka histogram terlihat datar (*flat*) dan berimbang pada kedua sisi (kanan dan kiri). Histogram bergeser ke kiri karena nilai-nilai pixel yang tinggi banyak yang berubah menjadi lebih kecil akibat di-*XOR*-kan dengan suatu bilangan acak. Histogram yang *flat* ini akan menyulitkan kriptanalisis dalam melakukan analisis statistik untuk menemukan kunci enkripsi karena jumlah *pixel* yang bernilai tinggi dan jumlah *pixel* bernilai rendah seimbang banyaknya[2].

Pada kedua sistem enkripsi, akan menghasilkan pola kunci yang hampir mirip sehingga keduanya sama-sama menghasilkan histogram yang datar. Artinya, kekuatan algoritma enkripsi *Chaotic Function* dan RC4 sama kuat dengan algoritma enkripsi RC4.

Untuk pengujian yang terakhir, akan dilakukan pengukuran terhadap performansi sistem yang dibangun. Performansi ini meliputi waktu pemrosesan untuk proses enkripsi dan proses dekripsi.



Gambar 10 Grafik waktu proses enkripsi



Gambar 11 Grafik waktu proses dekripsi

Berdasarkan data di atas, terlihat bahwa sistem enkripsi dengan RC4 memerlukan waktu proses yang lebih lama dengan sistem enkripsi *Chaotic Function* dan RC4. Hal ini disebabkan oleh proses steganografi dalam penyisipan kunci enkripsi ke dalam citra maupun ketika pengambilan kunci enkripsi dari citra terenkripsi. Pada sistem enkripsi dengan RC4, kunci yang akan disisipkan sejumlah 2048 bit. Pemrosesannya tentu jauh lebih lambat jika dibandingkan dengan sistem enkripsi dengan *Chaotic Function* dan RC4, karena hanya menyisipkan kunci sejumlah 64 bit.

Kesimpulannya, sistem enkripsi dengan *Chaotic Function* dan RC4 lebih cepat 95% dibandingkan dengan sistem enkripsi dengan RC4.

5. Kesimpulan dan Saran

Jika dibandingkan dengan sistem enkripsi RC4 konvensional, maka sistem enkripsi RC4 dan *Chaotic Function* memiliki tingkat keteracakan *keystream* yang lebih tinggi pada kunci dengan panjang kurang

dari 5 digit. Artinya, pada kunci-kunci tersebut tingkat keamanan sistem enkripsi RC4 dan *Chaotic Function* lebih tinggi dibandingkan dengan sistem enkripsi RC4 konvensional.

Kemudian pada pengujian parameter NPCR dan UACI, sistem enkripsi *Chaotic Function* dan RC4 memiliki nilai NPCR yang lebih rendah rata-rata 0.03% dan memiliki nilai UACI yang lebih rendah rata-rata 0.01% dibandingkan dengan sistem enkripsi RC4 konvensional. Karena perbedaanya sangat kecil, maka kedua sistem ini memiliki ketahanan yang sama terhadap *differential attack*.

Selanjutnya, pada pengujian histogram, kedua sistem enkripsi akan menghasilkan pola kunci yang teracak sehingga keduanya sama-sama menghasilkan histogram yang datar. Artinya, kekuatan algoritma enkripsi *Chaotic Function* dan RC4 sama kuat dengan algoritma enkripsi RC4.

Berdasarkan parameter kecepatan pemrosesan, sistem enkripsi RC4 dan *Chaotic Function* lebih cepat dibandingkan dengan sistem enkripsi RC4 konvensional. Sistem enkripsi RC4 dan *Chaotic Function* lebih cepat masing-masing 95.49% dan 96.59% untuk waktu proses enkripsi dan dekripsi.

Pada tingkat performansi sistem, semakin besar ukuran (dimensi) citra akan memperlambat proses enkripsi maupun dekripsi. Semakin besar ukuran (dimensi) citra, maka nilai SNR dan PSNR akan membesar pula. Namun, dimensi citra ternyata berbanding terbalik dengan nilai MSE.

Sistem enkripsi RC4 dan *Chaotic Function* memiliki keunggulan pada sisi kualitas citra dibandingkan dengan sistem enkripsi RC4 konvensional. Sistem enkripsi RC4 dan *Chaotic Function* menghasilkan kualitas yang lebih baik dengan menurunkan MSE 93,41%, dan SNR dan PSNR lebih tinggi masing-masing 16,46% dan 15,76%.

Adapun saran ditujukan untuk mengurangi luas bagian yang di*cropping* agar enkripsi lebih efisien. Untuk mengurangi luas bagian yang mengalami enkripsi perlu ditambahkan algoritma *edge detection* agar sistem tidak hanya bisa menge-*crop* bagian gambar dengan bentuk *rectangle*, namun juga bisa menge-*crop* gambar sesuai dengan bentuk objek gambar tersebut.

Daftar Pustaka

[1] H.S. Kwok., Wallace K.S. Tang. 2005. *A Fast Image Encryption System Based on Chaotic Maps with Finite Precision Representation*. Department of Electronic Engineering, City University of Hong Kong, Hong Kong.

[2] Munir, Rinaldi. *Enkripsi Selektif Citra Digital dengan Stream Cipher Berbasiskan pada Fungsi Chaotik Logistik Map.* Seminar Nasional dan ExpoTeknik Elektro 2011: 7 – 12

[3] Munir, Rinaldi. 2006. *Kriptografi*. Bandung : Informatika

[4] Piarsa, I Nyoman. *Steganografi pada Citra JPEG dengan Metode Sequential dan Spreading*. Lontar Komputer Vol 2 No 1 Juni 2011.

[5] Riad, Alaa M., et all. *Evaluation of the RC4 Algorithm as a Solution for Converged Networks*. Journal of Electrical Engineering, Vol. 60, No 3, 2009, 155-160.

[6] Rukhin, Andrew., et all. 2010. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. Special Publication 800-22.

[7] Stallings, William. 2011. *Cryptography and Network Security: Principles and Practice, fifth edition*. Upper Saddle River, NY : Prentice Hall.

[8] Sutoyo, T., Mulyanto, Edi., Suhartono, Vincent., Nurhayati, O.D., Wijanarto. 2009. *Teori Pengolahan Citra Digital*. Yogyakarta : Andi

[9] Wu, Yeu., Noonan, J.P., Agaian, Sos. *NPCR and UACI Randomness Tests for Image Encryption*. Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), April Edition, 2011