Perancangan Algoritma Kriptografi *Stream Cipher* dengan *Chaos*

Rinaldi Munir, Bambang Riyanto, Sarwono Sutikno

Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
E-mail: rinaldi@informatika.org, briyanto@lskk.ee.itb.ac.id, ssarwono@ieee.org

Abstraksi

Sistem dinamis *chaos* mempunyai properti yang berharga untuk keamanan data, yaitu kepekaan pada perubahan kecil pada kondisi awal sistem. Sifat peka ini berarti bahwa dua nilai awal dipilih sangat dekat satu sama lain, maka setelah sejumlah iterasi tertentu barisan nilai yang dihasilkannya akan berbeda secara signifikan. Makalah ini mempresentasikan rancangan algoritma kriptografi *stream cipher* dengan menggunakan persamaan *chaos* untuk pembangkitan elemen-elemen kunci yang berupa barisan bilangan acak. Algorima diimplementasikan dengan menggunakan Bahasa *C* dan diuji dengan bermacam-macam berkas dengan format apapun. Melalui pengujian diperoleh hasil bahwa serangan yang dilakukan dengan mencari nilai awal yang berapapun dekatnya dengan nilai awal semula tidak akan berhasil mendekripsi data dengan benar.

Kata Kunci: *chaos*, kriptografi, *stream cipher*, peka, nilai awal, difusi.

1. Pendahuluan

Salah satu aspek penting dalam sebuah sistem informasi adalah masalah keamanan data. Kriptografi sudah sejak lama digunakan sebagai metode pemgamanan data, salah satunya adalah kerahasiaan data dengan mekanisme enkripsi/dekripsi.

Stream cipher adalah algoritma kriptografi yang mengenkripsi atau mendekrisi aliran pesan bit per bit (atau *byte* per *byte*). Satu-satunya algoritma *stream cipher* yang sempurna aman, karena ia tidak dapat dipecahkan (*unbreakable cipher*), adalah *one-time pad* (*OTP*). *OTP* mempunyai dua karakteristik penting yaitu [3]: 1) panjang kunci sama dengan panjang pesan, dan 2) kunci adalah barisan bilangan acak.

Bilangan acak sangat penting di dalam kriptografi. Barisan bilangan acak yang dibangkitkan oleh *PRNG* (*Pseudo-random Number Generator*) mempunyai periode yang terbatas sehingga tidak aman untuk kriptografi karena pihak lawan dapat dengan mudah memprediksi barisan bilangan acak.

Sistem *chaos* sudah banyak digunakan di dalam pembangkitan bilangan acak. Barisan bilangan acak dibangkitkan dari iterasi terhadap fungsi *chaos* berdasarkan sebuah nilai awal dan satu atau lebih parameter.

Karakteristik yang paling penting dari sistem *chaos* adalah peka terhadap nilai awal; yaitu jika dua nilai awal dipilih sangat dekat satu sama lain, maka setelah sejumlah iterasi tertentu barisan nilai yang dihasilkannya akan berbeda secara signifikan.

Sifat peka semacam ini sangat berharga untuk algoritma kriptografi karena salah satu fitur yang diinginkan dari algoritma kriptografi adalah bila nilai awal yang digunakan untuk mengenkripsi data diubah sedikit, maka cipherteks yang dihasilkan akan berbeda signifikan [7]. Hal ini akan mengurangi daya serangan *brute force*, yaitu mencoba semua kemungkinan kunci (dalam hal ini nilai awal). Barisan nilai yang dibangkitkan dengan fungsi *chaos* sangat sulit diprediksi secara analitik tanpa mengetahui kunci rahasianya. Inilah yang menjadi faktor keamanan algoritma kriptografi yang berbasis sistem *chaos*.

2. *Stream Cipher*

Stream cipher adalah algoritma kriptografi yang mengenkripsi plainteks menjadi chiperteks satu bit atau satu karakter (1 *byte*), yang dalam hal ini panjang kunci sama dengan panjang pesan.

Contoh klasik *stream cipher* adalah *One-time pad* (*OTP*) [3]. *One-time pad* (*OTP*) adalah *stream cipher* yang melakukan enkripsi dan dekripsi satu karakter setiap kali. *Pad* adalah barisan 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.

Jika karakter yang digunakan adalah anggota himpunan 256 karakter (seperti karakter dengan pengkodean ASCII), maka enkripsi dapat dinyatakan sebagai penjumlahan modulo 256 dari satu karakter plainteks dengan satu karakter kunci *one-time pad*:

$$c_i = (p_i + k_i) \bmod 256 \qquad (1)$$

Penerima pesan menggunakan *pad* yang sama untuk mendekripsikan cipherteks menjadi plainteks dengan persamaan:

$$p_i = (c_i - k_i) \bmod 256 \qquad (2)$$

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 [2].
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.

3. Teori *Chaos*

Teori *chaos* berasal dari teori sistem yang memperlihatkan kemunculan yang tidak teratur, meskipun sebenarnya teori ini digunakan untuk menjelaskan kemunculan data acak [8].

Fenomena yang umum di dalam teori *chaos* adalah peka terhadap perubahan nilai awal, yang juga dikenal sebagai **ketergantungan yang peka pada nilai awal** (*sensitive dependence on initial condition*). Kepekaan ini berarti bahwa perbedaan kecil pada nilai awal fungsi, misalnya perubahan sebesar 10^{-100}, setelah fungsi diiterasi sejumlah kali, akan menghasilkan perbedaan yang sangat besar pada nilai fungsinya. Sebagai contoh, jika persamaan *chaos* dimulai dengan nilai awal 32, dan pada kali yang lain 32.000001, maka setelah 100 kali iterasi nilai persamaan dengan nilai awal pertama mungkin 137.54, sedangkan nilai persamaan dengan nilai awal kedua mungkin 1160.934.

Salah satu fungsi *chaos* sederhana adalah persamaan logistik di dalam ekologi yang digunakan untuk mensimulasikan pertumbuhan populasi spesies, yaitu

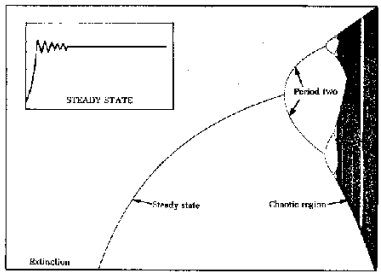
$$f(x) = r\,x(1 - x) \qquad (3)$$

Fungsi ini dapat dinyatakan dalam bentuk iteratif

$$x_{i+1} = r\,x_i\,(1 - x_i) \qquad (4)$$

Di dalam persamaan (3) dan (4) di atas x adalah populasi spesies pada interval waktu yang ditentukan dengan x_0 adalah nilai awal iterasi. Daerah asal x adalah dari 0 sampai 1, yang dalam hal ini 1 menyatakan populasi maksimum yang dan 0 menyatakan kepunahan, sedangkan $0 \le r \le 4$. Konstanta r menyatakan laju pertumbuhan. Konstanta r juga menyatakan bagian

nirlanjar dari persamaan. Ketika *r* meningkat, maka kenirlanjaran sistem juga naik.



Gambar 1. Diagram *bifurcation* untuk persamaan $x_{i+1} = r\, x_i\, (1 - x_i)$ [8]

Gambar 1 memperlihatkan kelakuan fungsi yang dalam hal ini sumbu-*x* menyatakan nilai *r* sedangkan sumbu *y* menyatakan status sistem, yaitu nilai-nilai *x*. Bila $0 < r < 1$, nilai awal berapapun akan menghasilkan kepunahan. Bila $1 < r < 3$, fungsi konvergen ke sebuah nilai (*fixed-point*), yaitu nilai *r* yang menghasilkan sistem mempunyai periode satu siklus. Ketika $r = 3$, kurva fungsi terpecah menjadi dua (*bifurcation*) menghasilkan dua nilai populasi berbeda, yang berarti nilai *x* secara periodik berosilasi dari status tinggi ke status rendah. Periode sistem pada nilai *r* ini adalah dua. Ketika *r* meningkat lagi, kurva fungsi terpecah lagi menjadi empat, yang berarti nilai-nilai *x* yang dihasilkan berosilasi di antara 4 nilai. Periode sistem pada nilai *r* ini adalah empat.

Demikianlah seterusnya *bifurcation* menjadi lebih cepat lagi dengan meningkatnya nilai *r* sampai tiba pada suatu nilai *r* tertentu sifat *chaos* pun mucul. Pada titik ini tidak mungkin lagi memprediksi kelakuan sistem. Kita dapat melihat bahwa ketika $r > 3.75$ sistem mulai melaju dengan cepat menuju area *chaos* (di dalam Gambar 1 area tersebut diarsir hitam) [9]. Akhirnya, ketika $r = 4$, iterasi bergantung sepenuhnya pada nilai awal x_0 dan nilai-nilai yang dihasilkan muncul acak meskipun sistem ini deterministik [10]. Nilai-nilai *chaos* yang dihasilkan akan berada di dalam rentang yang lengkap antara 0 dan 1.

4. Aplikasi *Chaos* untuk Kriptografi

Meskipun sistem *chaos* muncul dengan ketidakteraturan yang tinggi, tetapi ia deterministik artinya dimungkinkan membangkitkan nilai-nilai *chaos* dengan kepastian [13]. Hal ini adalah fitur yang menjanjikan untuk komunikasi secara aman.

Properti *chaos* yang paling bernilai bagi algoritma kriptografi adalah kepekaannya pada nilai awal. Sebagaimana dijelaskan di dalam [1], salah satu dari dua prinsip Shannon yang dijadikan panduan dalam perancangan algoritma kriptografi adalah difusi (*diffusion*), yaitu menyebar pengaruh 1 bit (atau digit) plainteks ke seluruh bit (digit) cipherteks dengan maksud untuk menyembunyikan hubungan statistik antara plainteks dan cipherteks. Perluasan prinsip ini adalah menyebar pengaruh 1 bit (digit) kunci ke seluruh bit cipherteks [12]. Prinsip difusi ini relevan dengan properti *chaos* di atas, sebab jika kondisi awal yang digunakan untuk mengenkripsi data diubah sedikit, misalnya 1 bit, cipherteks yang dihasilkan akan berbeda secara signifikan. Sifat peka menjamin bahwa jika pihak lawan mencoba mendekripsi data dengan kondisi awal berbeda dan mencari pola hubungan plainteks dan cipherteks, cara itu akan gagal [7].

Kebanyakan *stream cipher* menggunakan pembangkit bilangan acak untuk enkripsi. Pembangkit bilangan acak terdapat pada dua sisi, sisi pengirim dan sisi penerima. Dalam hal ini, sistem *chaos* dapat digunakan untuk membangkitkan aliran kunci bilangan acak, selanjutnya bilangan acak ini digunakan untuk me-"*mask*" plainteks.

7. Rancangan Algoritma

Algoritma *stream cipher* yang dirancang di dalam makalah ini me-*mask* karakter-karakter plainteks p_i

dengan aliran kunci k_i yang dibangkitkan dari persamaan chaos. Algoritma *stream cipher* yang dirancang terdiri atas beberapa bagian:

i) fungsi pemotongan,
ii) pembangkitan aliran kunci,
iii) enkripsi dan dekripsi

(i) Fungsi Pemotongan

Enkripsi dan dekripsi beroperasi dalam himpunan bilangan bulat yang nilainya dari 0 sampai 255, sedangkan barisan nilai *chaos* yang digunakan sebagai aliran kunci adalah bilangan riil antara 0 dan 1. Agar barisan nilai *chaos* dapat dipakai enkripsi dan dekripsi, maka nilai *chaos* dikonversi ke nilai integer. Ada beberapa teknik konversi yang dapat digunakan, teknik yang umum misalnya mengambil 3 angka terakhir pada bagian mantissa bilangan riil.

Di dalam makalah ini, konversi nilai *chaos* ke *integer* dilakukan dengan menggunakan fungsi pemotongan yang diusulkan oleh [9]. Caranya, nilai *chaos* dikalikan dengan 10 berulangkali sampai ia mencapai panjang angka (*size*) yang diinginkan, lalu memotong hasil perkalian tersebut untuk mengambil bagian *integer*-nya saja. Secara matematis, nilai *chaos* x dikonversi ke *integer* dengan menggunakan persamaan berikut:

$$T(x, size) = \left\| x * 10^{count} \right\|, x \neq 0 \qquad (5)$$

yang dalam hal ini *count* mulai dari 1 dan bertambah 1 sampai $x * 10^{count} > 10^{size-1}$. Hasilnya kemudian diambil bagian *integer* saja (dilambangkan dengan pasangan garis ganda pada persamaan 11).

(ii) Pembangkitan aliran kunci

Kunci yang digunakan untuk me-*mask* plainteks dinamakan *pad*. Setiap elemen *pad* dihasilkan dari konversi nilai *chaos* ke *integer*, seperti yang sudah diterangkan di atas. Nilai-nilai *chaos* dibangkitkan dari persamaan (4) dengan mengambil konstanta r = 4.0. Normalnya, nilai x_i dihitung langsung dari nilai *chaos* sebelumnya, x_{i-1}. Ini berarti bila seseorang mengetahui sebuah nilai x_i dari barisan nilai *chaos*, maka ia dapat menggunakan x_i untuk membangkitkan x_{i+1}, x_{i+2}, …, yang selanjutnya digunakan untuk mendekripsi cipherteks.

Untuk menambah kekuatan sistem, maka nilai x_i dibangkitkan setelah sejumlah iterasi tertentu. Tujuannya adalah untuk menghilangkan korelasi antara nilai-nilai *chaos*. Jumlah iterasi yang dibutuhkan untuk menghitung nilai *chaos* pertama, x_1, ditentukan oleh nilai awal, x_0. Nilai awal ini dikonversi ke *integer* dengan menggunakan persamaan (5), hasilnya adalah jumlah iterasi yang diperlukan untuk mengiterasi persamaan (4). Nilai x yang diperoleh pada akhir iterasi berlaku sebagai "x_0" yang baru untuk menghitung x_1. Untuk x_2, x_3, dan seterusnya, jumlah iterasinya ditentukan dari jumlah iterasi untuk nilai *chaos* sebelumnya ditambah dengan *size*.

Dengan cara ini, seseorang yang mengetahui suatu nilai x_i tertentu tidak mungkin dapat menghitung x_{i+1} tanpa mengetahui jumlah iterasi yang diperlukan untuk mengiterasi persamaan (4). Jumlah iterasi awal ditentukan oleh x_0. Jadi, nilai awal merupakan nilai yang sangat menentukan keamanan *stream cipher*. Terdapat sejumlah tidak berhingga nilai-nilai antara 0 dan 1, oleh karena itu *exhaustive key search* untuk menemukan x_0 menjadi sesuatu yang tidak mungkin dilalukan. Selain itu, seperti yang sudah dijelaskan di bagian 4, fungsi *chaos* peka terhadap perubahan kecil pada nilai awal, sehingga jika nilai awal yang dicoba pihak lawan sangat dekat dengan nilai awal yang digunakan untuk mengenkripsi data, pihak lawan masih akan memperoleh keluaran yang salah.

(iii) Enkripsi dan dekripsi

Enkripsi dan dekripsi dikerjakan dengan menggunakan persamaan (1) dan (2).

8. Implementasi *Stream Cipher*

Rancangan algoritma *stream cipher* yang telah dijelaskan di atas diimplementasikan dalam bahasa *C* menjadi dua buah program. Panjang angka *pad* didefinisikan sebagai konstanta `size` yang nilainya 3, sedangkan parameter *chaos r* didefinisikan sebagai konstanta `r` yang nilainya 4.0.

Program dapat mengenkripsi berkas bertipe apapun (`.doc`, `.jpg`, `.xls`, `.bmp`, `.ppt`, `.pdf`, dan sebagainya).

9. Hasil-hasil Eksperimen

Program stream cipher dengan *chaos* diujicoba dengan sebuah berkas teks yang berukuran sedang.. Plainteks dienkripsi dengan menggunakan nilai awal r = 4.0, *size* = 3, dan x_0 = 0.00230872. Jika cipherteks didekripsi dengan nilai awal yang sama seperti pada waktu enkripsi, plainteks yang dihasilkan tepat sama seperti semula. Namun, jika cipherteks didekripsi dengan nilai awal yang sedikit berbeda, yaitu x_0 = 0.002308716, maka plainteks yang dihasilkan salah karena barisan nilai *chaos* yang dihasilkan berbeda (peka terhadap perubahan kecil pada nilai awal).

10. Kesimpulan

Makalah ini sudah mempresentasikan rancangan algoritma *stream cipher* dengan *chaos*. *Chaos* dapat digunakan untuk membangkitkan barsian kunci acak untuk mengenkripsi plainteks dengan algoritma *stream cipher*. *Chaos* mempunyai properti yang berguna untuk kriptografi yaitu kepekaan terhadap perubahan kecil pada nilai awal. Melalui pengujian dibuktikan bahwa perubahan nilai awal yang sangat kecil menghasilkan barisan kunci yang berbeda secara signifikan.

Referensi

[1] Bruce Schneier, *Aplied Cryptography 2nd*, John Wiley & Sons, 1996

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

[3] Rinaldi Munir, *Diktat Kuliah IF5054 Kriptografi*, Departemen Teknik Informatika ITB, 2005.

[4] Fred Piper dan Sean Murphy, *Cryptography, A Very Short Introduction*, Oxford University Press, 2002.

[5] Tanfeng Sun, Lili Cui, dan Shuxun Wang, *Research on Technology of Chaos Secrecy Comunication in Digital Watermarking*, Spinger-Verlag Berlin Heidelberg, 2002.

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

[7] Krish M. Roskin dan Jonathan B. Casper, *From Chaos to Cryptography*, 1999.

[8] www.yahoo.com, *Chaos Theory: A Brief Introduction*, diakses pada bulan November 2005

[9] James Lampton, *Chaos Cryptography: Protecting Data Using Chaos*, Mississippi School for Mathematics and Science.

[10] R. Clarck Robinson, *An Introduction to Dynamical Systems, Continuous and Discrete*, Pearson Prentice Hall, 2004.

[11] Shujun Li, *Analysis and New Designs of Digital Chaotic Chipers*, Disertasi Ph.D University of Beijing.

[12] Ljupco Kocarev, *Chaos Based Cryptography: A Brief Overview*, IEEE 2001.

[13] Ninan Sajeeth Philip dan K. Babu Joseph, *Chaos for Stream Cipher*, Department of Physics, Cochin University of Science and Technology, 2001.