J. Math. and Its Appl.
E-ISSN: 2579-8936
P-ISSN: 1829-605X
Vol. 14, No. 1, Mei 2017, 1–15

REGENERASI FUNGSI POLINOMIAL DALAM RANCANGAN ALGORITMA BERBASIS CSPRNG CHAOS SEBAGAI PEMBANGKIT KUNCI PADA KRIPTOGRAFI BLOCK CIPHER

Alz Danny Wowor
Program Studi Teknik Informatika, Fakultas Teknologi Informasi
Universitas Kristen Satya Wacana
Jl. Diponegoro 52-60, Salatiga, Jawa Tengah 50711
alzdanny.wowor@staff.uksw.edu

Abstrak

Penelitian ini mencari model baru dari fungsi polinomial yang dapat digunakan sebagai pembangkit bilangan acak berbasis CSPNRG *chaos*, kemudian dijadikan sebagai kunci pada kriptografi *block cipher*. Proses dilakukan dengan meregenerasi polinomial menggunakan *fixed point iteration* menjadi fungsi iteratif, dan pengambilan integer pada mantissa untuk memperoleh bilangan acak dari setiap iterasi.

Setiap fungsi polinomial derajat-1, derajat-2, dan derajat-3 dapat digunakan sebagai fungsi pembangkit, tetapi diperlukan pemilihan koefisien dan konstanta yang tepat dan juga ketangkasan dalam proses manipulasi aljabar pada *fixed point iteration*. Secara spesifik, algoritma yang dirancang merupakan proses yang ampuh karena dapat menghasilkan bilangan acak walaupun secara fungsi iterasi tidak dapat menghasilkan bilangan acak.

Pengujian korelasi pada *block cipher* menggunakan kunci dari bilangan acak berada pada kategori 'rendah', sehingga secara kriptografi kunci tersebut dapat membuat plainteks dan cipherteks tidak berhubungan secara statistik, kondisi ini akan mempersulit kriptanalis untuk melakukan krip-

tanalisis. Fungsi polinomial yang menjadi pembangkit dan menghasilkan bilangan acak dapat menjadi embrio dalam membangun konsep *unbreakble cipher*.

Kata kunci : *Fungsi Polinomial, CSPRNG Chaos, Fixed Point Iteration, Pembangkit Kunci, Block Cipher.*

1. Pendahuluan

Kunci memainkan peran yang sangat penting dan menjadi indikator utama dalam merancang kriptografi *block cipher*. Setiap algoritma yang dirancang harus memperhatikan operasi pembangkitan kunci yang tidak memudahkan penebakan secara langsung dan menghilangkan korespondensi satu ke satu antara plainteks dan cipherteks. Dengan demikian algoritma akan berhasil memberikan efek konfusi dari Shannon, karena perubahan kecil pada kunci akan mengakibatkan perubahan yang signifikan terjadi pada cipherteks.

Cryptographically Secure Pseudorandom Generator (CSPNRG) berbasis *Chaos* merupakan proses pembangkitan kunci yang dapat menjamin efek konfusi pada algoritma. Salah satu generator yang sering digunakan adalah fungsi logistik dari Lorenz $f(x) = rx(1-x)$ yang dibuat dalam iterasi $x_{i+1} = rx_i(1-x_i)$. Fungsi ini dalam bentuk polinomial derajat dua, dan mempunyai proses yang sederhana untuk mendapatkan bilangan acak. Setiap iterasi akan mewakilkan sebuah nilai dengan *range* atau daerah hasil berada antara 0 dan 1. Hasil iterasi dalam bentuk desimal sehingga diperlukan fungsi pemotongan untuk menghsilkan *integer* yang kemudian digunakan sebagai kunci pada kriptografi.

Penelitian ini mencari model baru yang dapat digunakan sebagai generator dengan meregenerasi fungsi polinomial menggunakan metode *fixed point iteration*. Setiap fungsi yang berhasil di lakukan manipulasi aljabar, akan melalui proses iterasi dengan melihat kecenderungan grafik akan berhasil membentuk *chaos* atau hanya menghasilkan nilai yang membentuk sebuah pola. Apabila fungsi iteratif membentuk pola atau grafiknya menuju pada sebuah titik, maka data akan mempunyai sifat periodik yang lebih pendek. Hasil ini tentu menunjukkan fungsi gagal untuk menjadi pembangkit. Sebaliknya apabila membentuk *chaos*, maka fungsi tersebut dapat dijadikan sebagai pembangkit pada kriptografi *block cipher*.

Pengujian statistik seperti korelasi dilakukan pada *block cipher* setelah setiap kunci yang diperoleh dari pembangkitan bilangan acak, sehingga dapat diketahui seberapa baik fungsi pembangkit dari polinomial mampu untuk membuat plainteks dan cipherteks tidak berhubungan secara statistik. Tujuan dari penelitian ini adalah memperoleh fungsi polinomial yang dapat menjadi pembangkit dan menghasilkan bilangan acak berbasis CSPNRG *chaos* sehingga dapat digunakan dalam membangun konsep kriptosistem yang *unbreakable cipher*.

2. Kajian Pustaka

2.1. Fungsi Polinomial

Sebuah fungsi p merupakan sebuah polinomial [1], dimana n adalah bilangan bulat non-negatif dan nilai $a_0, a_1, a_2, \cdots, a_n$ adalah konstanta yang disebut sebagai koefisien dari polinomial.

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0 \tag{1}$$

Domain dari setiap polinomial adalah bilangan rill $R = (-\infty, \infty)$. Jika koefisien $a_n \neq 0$ maka derajat atau pangkat tertinggi dari polinomial adalah n [2].

2.2. *Block Cipher*

Rangkaian bit-bit plainteks (P) dibagi menjadi *block* bit, dan akan menggunakan kunci (K) yang mempunyai panjang sama dengan panjang *block* bit untuk menghasilkan cipherteks (C). Fungsi enkripsi memetakan P ke C, dan fungsi dekripsi memetakan C ke P, diberikan pada Persamaan 2 [3].

$$E_K(P) = C, \qquad D_K(C) = P \tag{2}$$

Setiap *block* plainteks berukuran m bit yang tersususun n *block*, ditulis $P = (p_1, p_2, \cdots, p_n)$: $p_i \in \{0, 1\}$ dan *block* cipherteks, $C = (c_1, c_2, \cdots, c_n)$: $c_i \in \{0, 1\}$. Ukuran *block* biasanya 64, 128, 256 bit, dan juga dikembangkan menjadi 512 bit [4].

2.3. CPSNRG *Chaos*

CPSNRG atau *cryptographically secure pseudorandom generator chaos* merupakan pembangkit bilangan acak yang dapat menghasilkan bilangan yang tidak dapat diprediksi [4]. Ditemukan oleh Edward Lorenz pada tahun 1960, menggunakan model perkiraan cuaca, fungsi tersebut adalah $f(x) = rx(1 - x)$ yang dinyatakan dalam bentuk iteratif menjadi

$$x_{i+1} = rx_i(1 - x_i) \tag{3}$$

CPSNRG *Chaos* memiliki sifat (*butterfly effect*) dimana, perubahan kecil pada nilai *input* berakibat terjadi perubahan yang sangat signifikan pada nilai *output*.

2.4. *Fixed Point Iteration*

Metode *fixed point iteration* (FPI) merupakan salah satu metode terbuka (*open method*) dari pencarian akar persamaan non-linier. Metode ini mengambil sebuah

fungsi dengan $f(x) = 0$, selanjutnya menyusun kembali fungsi tersebut dengan menempatkan x terpilih di sisi sebelah kiri sehingga akan diperoleh $x = g(x)$.

$$x_i = g(x_{i-1}) \tag{4}$$

Tahapan berikutnya dilakukan manipulasi aljabar untuk dapat menentukan sebuah formula dalam mencari nilai x dari nilai x sebelumnya. Fungsi iterasi ditunjukkan pada Persamaan 4, dimana $i = 1, 2, \cdots$, dengan tebakan awal x_0 sebagai nilai inisialisasi [5].

2.5. Korelasi

Korelasi merupakan salah satu nilai statistik yang digunakan untuk melihat hubungan antara variabel bebas (x) dan tak bebas (y) yang bersifat kuantitatif. Persamaan korelasi diformulasikan sebagai berikut [6]:

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}} \tag{5}$$

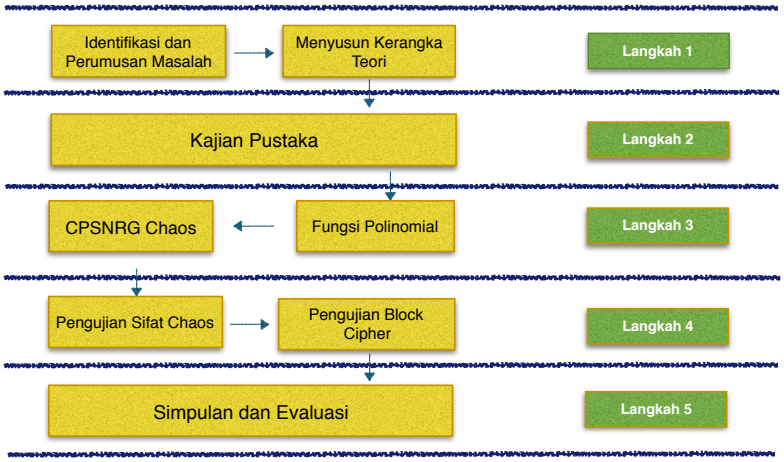
dengan ketentuan $-1 \leq r \leq r$ dan interpretasi koefisien korelasi nilai r ini dapat dirangkum dalam tabel berikut:

Tabel 1: Tingkat Hubungan Nilai Korelasi

Interval Koefisien	**Tingkat Hubungan**
0,00 - 0,19	Sangat Rendah
0,20 - 0,39	Rendah
0,40 - 0,59	Cukup
0,60 - 0,79	Kuat
0,80 - 1,00	Sangat Kuat

3. Metode Penelitian

Secara umum metode penelitian diberikan dalam beberapa langkah yang ditunjukan pada Gambar 1. Masing-masing langkah merupakan parsial dari kajian atau rencana dari penelitian secara keseluruhan yang menghantarkan masalah penelitian menuju pada tujuan penelitian. Langkah 1 dan langkah 2 merupakan proses awal, dimana perenungan akan pembangkitan bilangan acak yang dapat dibangkitkan dengan skema berbeda dan juga dari fungsi iterasi yang berbeda selain fungsi Lorentz. Rancangan penelitian diberikan pada Gambar 2, yang adalah



Gambar 1: Tahapan Penelitian

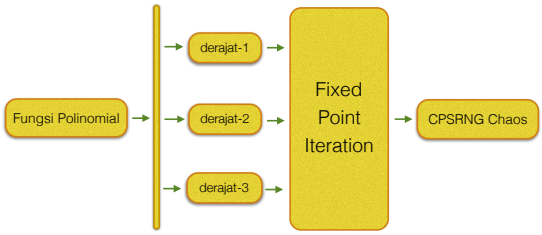
penjelasan rinci dari langkah 3. Selanjutnya pengujian akan sebuah urutan bilangan adalah sebagai bilangan acak berbasis CSPRNG *chaos* dilakukan pada langkah 4. Uji grafik berdasarkan diagram Scatter dan uji korelasi untuk melihat keterhubungan plainteks dan cipherteks. Sedangkan langkah terakhir adalah bagaimana membuat sebuah simpulan dan melakukan evaluasi dari riset yang dilakukan.

4. Hasil dan Pembahasan

4.1. Rancangan Penelitian

Pencarian fungsi pembangkit dari polinomial dilakukan pada derajat-1, derajat-2, dan derajat-3 seperti yang diberikan pada Gambar 1. Setiap derajat fungsi yang terpilih diubah menjadi fungsi iteratif menggunakan *fixed point iteration* (FPI), sehingga dapat dilakukan iterasi untuk memperoleh luaran berupa bilangan yang dapat dijadikan sebagai kunci. Setiap urutan bilangan luaran dari FPI akan diuji berdasarkan skema CSPNRG *chaos*.

Kajian *chaos* dapat terlihat apabila setiap luaran FPI di visualisasi ke dalam koordinat Cartesius, dimana iterasi sebagai absis dan ordinatnya adalah hasil iterasi. Apabila pada diagram menunjukkan sebuah pola, atau bentuk tertentu yang memudahkan penebakan maka fungsi tersebut gagal sebagai pembangkit. Sebaliknya, grafik yang membentuk *chaos* akan terlihat setiap hasil iterasi nampak pecah dan tidak mempunyai kecenderungan membentuk sebuah pola.



Gambar 2: Skema Global Pencarian Bilangan Acak

4.2. Pencarian pada Polinomial Derajat-1

Bentuk umum dari polinomial derajat-1 adalah $p(x) = a_1x + a_0$. Setiap fungsi yang akan selalu menghasilkan hubungan berupa garis lurus antara variabel bebas dan tak bebas. Kondisi ini yang menarik untuk di kaji, apakah sifat linierisasi yang memudahkan penebakan, akan masih nampak pada pembangkitan bilangan acak. Misalkan dipilih $p(x) = x - 5$, manipulasi aljabar dilakukan dengan memperhatikan syarat $p(x) = 0$ dari metode FPI, maka $x-5 = 0 \Leftrightarrow 17x - 16x - 5 = 0 \Leftrightarrow 17x = 16x + 5 \Leftrightarrow x = (16x + 5)/17$. Sehingga fungsi iteratif dengan polinom derajat-1 diberikan pada Persamaan 6.

$$x_i = (16x_{i-1} + 5)/17 \tag{6}$$

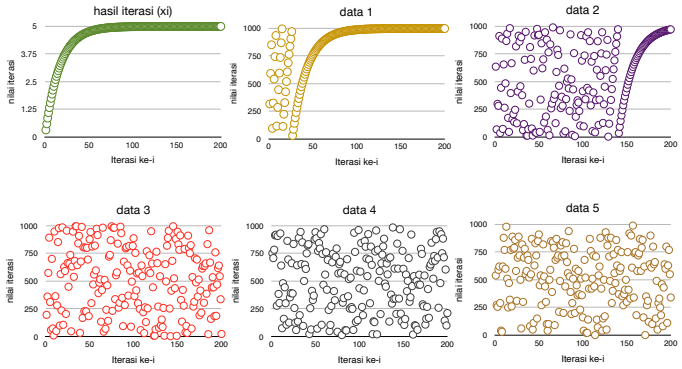
Inisialisasi dipilih $x_0 = 0,024988457987$ dan Persamaan (6) digunakan sebagai fungsi iterasi, maka diperoleh 5 iterasi pertama yang diberikan pada Tabel 3. Hasil iterasi berada dalam 15 *digit* sehingga memungkinkan untuk diambil lima data untuk dijadikan sebagai kunci. Pengambilan bilangan pada mantissa berdasarkan urutan ke 1, 2, dan 3 dinamakan *data* 1, sedangkan pengambilan pada urutan ke 4, 5, dan 6 adalah *data* 2, begitu seterusnya untuk *data* 3, *data* 4 dan sampai pada *data* 5 untuk pengambilan pada urutan 13, 14, dan 15. Tabel 2 menunjukkan ba-

Tabel 2: Hasil Pembangkitan Bilangan Acak $x_i = (16x_{i-1} + 5)/17$

i	x_i	*data* 1	*data* 2	*data* 3	*data* 4	*data* 5
1	0,317636195752539	317	636	195	752	539
2	0,593069360708272	593	69	360	708	272
3	0,852300574784256	852	300	574	784	256
4	1,096282893914590	96	282	893	914	590
5	1,325913311919620	325	913	311	919	620

gaimana mengambil *integer* dari mantissa, pada fungsi iterasi $x_i = (16x_{i-1}+5)/17$

menghasilkan 5 data. Penelitian ini mengambil tiga digit untuk setiap data, karena memperhatikan maksimum *digit* dari karakter ASCII sebanyak 256 karakter. Hasil iterasi diberikan pada Gambar 3 untuk 200 iterasi pertama. Hasil iterasi dari $x_i = (16x_{i-1}+5)/17$ tidak menghasilkan bilangan acak, karena menghasilkan diagram yang berpola seperti fungsi logaritma. Tetapi dengan pengambilan bilangan pada mantissa untuk *data* 1 belum menghasilkan bilangan *chaos*. Untuk *data* 2, hanya menghasilkan 27 bilangan *chaos*. Kecuali pada *data* 3, *data* 4, dan *data* 5 memperoleh urutan bilangan yang tidak dapat diprediksi, sehingga skema CSPNRG *chaos* terjadi pada tiga data ini. Dengan demikian pembangkitan fungsi $x_i = (16x_{i-1}+5)/17$ dapat menjadi fungsi pembangkit. Selain Persamaan (6), ma-



Gambar 3: Iterasi Fungsi $x_i = (16x_{i-1}+5)/17$ dengan $x_0 = 0,024988457987$.

nipulasi aljabar juga dapat dilakukan untuk memperoleh calon fungsi pembangkit lain seperti $x_i = 2x_{i-1} - 5$, $x_i = (3x_{i-1} - 5)/2$, dan $x_i = (2x_{i-1} + 5)/3$. Berdasarkan pengujian untuk $x_i = 2x_{i-1} - 5$ dan $x_i = (3x_{i-1} - 3)/2$ tidak dapat dijadikan sebagai pembangkit, karena hasil iterasi meningkat secara singnifikan, sehingga pada pengambilan *integer* pada mantissa menjadi tidak dapat dilakukan. Selain itu penelitian ini berharap hasil iterasi berada antara $0 < x < 10$. Sedangkan untuk fungsi 3 dari Tabel 3, dapat diregenerasi menjadi fungsi pembangkit, hanya saja dengan inisialisasi yang sama tidak diperoleh bilangan acak untuk 200 iterasi pertama. Hasil iterasi dan pengambilan data membentuk pola dan mempunyai kecenderungan menuju pada sebuah titik.

4.3. Pencarian pada Polinomial Derajat-2

Bentuk umumnya $p(x) = a_2x^2 + a_1x + a_0$, dimana $a_2 \neq 0$. Diambil fungsi $p(x) = x^2 - 3$, dengan menggunakan metode FPI dan manipulasi aljabar maka

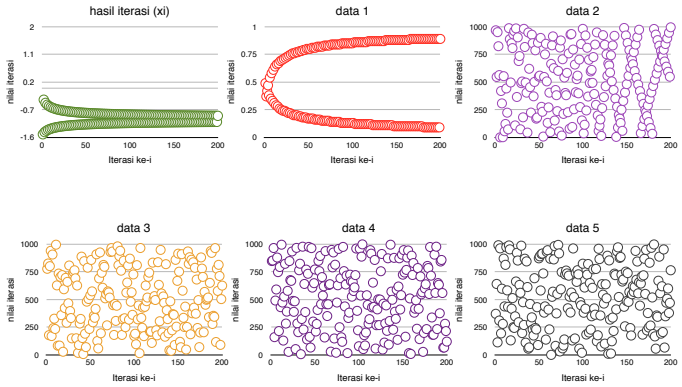
diperoleh fungsi iteratif yang diberikan pada Tabel 3. Setiap fungsi iteratif pada Tabel 4 menggunakan nilai inisialisasi yang sama yaitu $x_0 = 0,024988457987$, dan pengambilan bilangan menggunakan *data* 1 sampai *data* 5 seperti pada polinomial derajat-1. Fungsi 1, $x_i = 3/x_{i-1}$ dalam proses iterasi hanya menghasilkan dua nilai untuk masing-masing data dan membentuk dua garis lurus. Walaupun berhasil menjadi fungsi iterasi, tetapi tidak dapat digunakan sebagai fungsi pembangkit, karena gagal untuk menjadi CSPNRG *chaos*. Fungsi kedua $x_i = (x_{i-1}^2 - 3)/2$

Tabel 3: Fungsi Iteratif dari $p(x) = x^2 - 3$

Fungsi	**Hasil FPI**	**Fungsi Iteratif**
fungsi 1	$x = 3/x$	$x_i = 3/x_{i-1}$
fungsi 2	$x = (x^2 - 3)/2$	$x_i = (x_{i-1}^2 - 3)/2$
fungsi 3	$x = x - (x^2 - 3)$	$x_i = x_{i-1} - (x_{i-1}^2 - 3)$
fungsi 4	$x = x - (x^2 - 3)/2$	$x_i = x_{i-1} - (x_{i-1}^2 - 3)/2$
fungsi 5	$x = x - (x^2 - 3)/176$	$x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$

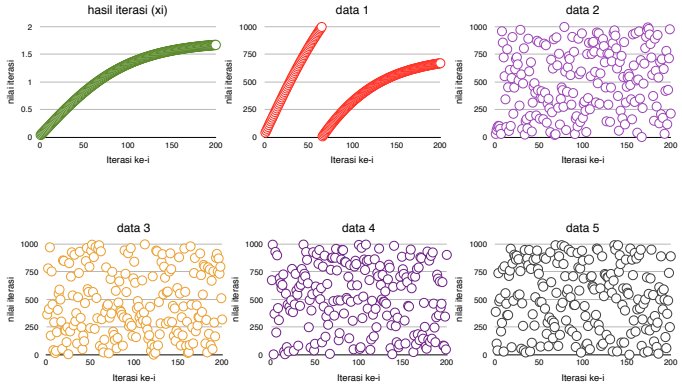
memberikan hasil iterasi yang baik karena grafik fungsi sudah terjadi *biffurcation* atau memecah manjadi dua bagian. *data* 1 masih belum membentuk bentuk *chaos*, tetapi menampilkan keindahan grafik logaritma yang simetris terhadap garis $x = 0,5$. Bilangan acak untuk pengambilan 200 iterasi pertama terjadi pada *data* 3, *data* 4, dan *data* 5, dan berhasil menjadi CSPRNG *chaos*. Hasil secara keseluruhan dapat dilihat pada Gambar 4. Hasil pada *data* 3 sudah membentuk sebagian bilangan *chaos*, hanya saja setelah iterasi ke-131 sudah kembali membentuk pola. Untuk *data* 3, *data* 4, dan *data* 5 sudah jelas membentuk bilangan acak berbasis CSPNRG *chaos* karena diagram nampak memecah dan tidak memiliki pola yang memudahkan penebakan. Dengan demikian fungsi $x_i = (x_{i-1}^2 - 3)/2$ dapat digunakan sebagai fungsi pembangkit bilangan acak. Fungsi iteratif yang ketiga $x_i = x_{i-1} - (x_{i-1}^2 - 3)$ tidak dapat menghasilkan bilangan acak. Hasil iterasi naiknya terlalu cepat, pada iterasi ke-5 sudah diperoleh nilai yang melebihi 10^6. Peningkatan yang signifikan merupakan proses yang kurang baik dalam pencarian bilangan acak, hal ini disebabkan karena pengambilan *interger* dari mantissa, bukan pada pokok bilangan. Sehingga diharapkan nilai yang diperoleh kurang dari 10^1, untuk semua iterasi. Untuk fungsi iterasi selanjutnya $x_i = x_{i-1} - (x_{i-1}^2 - 3)/2$, secara proses dapat menghasilkan luaran yang baik, tetapi belum dapat menghasilkan *chaos* pada 200 iterasi pertama. Hasil dari *data* 1 hanya menghasilkan 17 bilangan acak, 39 bilangan berbeda diperoleh pada *data* 2, sedangkan untuk *data* 3 menghasilkan 59 bilangan acak, pada *data* 4 meningkat menjadi 78 bilangan, dan untuk *data* 5 diperoleh sebanyak 97 bilangan yang berbeda.

Hasil iterasi dari fungsi $x_{n+1} = x_n - ({x_n}^2 - 3)/176$ adalah fungsi kedua yang



Gambar 4: Iterasi Fungsi $x_i = (x_{i-1}^2 - 3)/2$ dengan $x_0 = 0,024988457987$.

berhasil memberikan hasil dalam bilangan CSPNRG *chaos*, khususnya pada *data* 2 sampai *data* 5. Hasil secara grafik menunjukkan nilai yang tidak akan melebihi 9, sehingga masih aman untuk pengambilan mantissa dari setiap hasil iterasi. Hasil secara lengkap dalam diagram Scatter diberikan pada Gambar 5. Persamaan



Gambar 5: Iterasi Fungsi $x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$ dengan $x_0 = 0,024988457987$.

$x_i = (x_{i-1}^2 - 3)/2$ dan $x_{n+1} = x_n - ({x_n}^2 - 3)/176$ dari Tabel 4, merupakan fungsi iterasi yang menghasilkan bilangan CSPRNG berbasis *chaos* dari Polinomial derajat-2 $p(x) = x^2 - 3$. Hal menarik yang terjadi pada fungsi 2 adalah range dari fungsi iterasi berada pada nilai negatif, tetapi sekali lagi karena penelitian ini

dirancang untuk mengambil *integer* pada mantissa. Sehingga hasil iterasi dalam negatif, tetap menghasilkan bilangan acak CSPRNG *chaos*.

4.4. Pencarian pada Polinom Derajat-3

Polinom derajat-3 atau sering disebut sebagai fungsi kubik, dengan bentuk umum $p(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Seperti pada fungsi linier dan kuadrat, digunakan metode FPI untuk memperoleh fungsi iteratif untuk mencari urutan bilangan yang digunakan sebagai kunci. Dipilih $f(x) = x^3+6x^2+19x-20$, maka dengan metode FPI dapat diperoleh beberapa fungsi yang diberikan pada Tabel 5.

Tabel 4: Fungsi Iteratif dari $f(x) = x^3 + 6x^2 + 19x - 20$

Fungsi	**Hasil FPI**	**Fungsi Iteratif**
fungsi 1	$x = x^3 + 6x^2 + 20x - 20$	$x_i = x_{i-1}^3 + 6x_{i-1}^2 + 20x_{i-1} - 20$
fungsi 2	$x = (20 - x^3 - 6x^2)/19$	$x_i = (20 - x_{i-1}^3 - 6x_{i-1}^2)/19$
fungsi 3	$x = \sqrt{(20 - x^3 - 19x)/6}$	$x_i = \sqrt{(20 - (x_{i-1})^3 - 19x_{i-1})/6}$
fungsi 4	$x = (20 - x^3 - 19x)/6x$	$x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$
fungsi 5	$x = (20 - 6x^2 - 19x)/x^2$	$x_i = (20 - x_{i-1}^2 - 19x_{i-1})/x^2$
fungsi 6	$x_i = \sqrt[3]{20 - 6x^2 - 19x}$	$x_i = \sqrt[3]{20 - 6(x_{i-1})^2 - 19x_{i-1}}$

Manipulasi aljabar yang dilakukan dapat menghasilkan 6 fungsi iteratif. Hasil iterasi dari *fungsi* 1 pada Tabel 4 turun terlalu cepat, sebagai contoh pada iterasi ketiga sudah lebih kecil dari -10^{11}. Hasil ini bertentangan dengan rancangan algoritma yang mengharapkan nilai iterasi berada pada interval $0 \leq x \leq 9$. Sehingga $x_i = x_{i-1}^3 + 6x_{i-1}^2 + 20x_{i-1} - 20$ tidak dapat digunakan sebagai fungsi pembangkit.

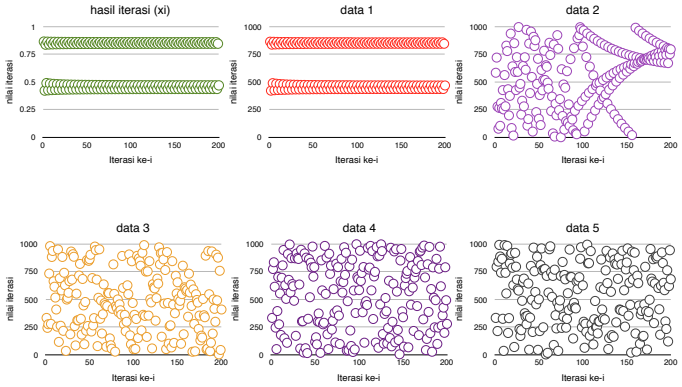
Fungsi iterasi kedua $x_i = (20 - x_{i-1}^3 - 6x_{i-1}^2)/19$ dari polinomial derajat-3 berhasil menjadi fungsi iteratif, tetapi bilangan berbeda yang dihasilkan sangat terbatas, secara berturut-turut untuk *data* 1 sampai *data* 5 adalah 13, 27, 42, 55, dan 71 bilangan. Walaupun menghasilkan bilangan acak, tetapi tidak berhasil dalam menghasilkan bilangan acak untuk 200 iterasi pertama. Fungsi iterasi ini tidak digunakan sebagai kunci karena *range* terlalu kecil atau < 200.

Fungsi iterasi ketiga, $x = \sqrt{(20 - x^3 - 19x)/6}$ adalah fungsi sepotong (*piecewise function*) hanya terdefenisi pada interval $-2,336 \leq x \leq 1,620$ dan juga untuk $x \leq -5,283$. Sehingga ketika digunakan sebagai sebuah fungsi iterasi, ada sejumlah bilangan yang tidak terdefenisi. Hal ini yang mengakibatkan pada iterasi kedua dan seterusnya mengakibatkan fungsi tidak terdefenisi. Sehingga *fungsi* 3 tidak dapat digunakan sebagai pembangkit bilangan acak.

Fungsi iterasi $x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$ adalah fungsi yang dapat menghasilkan bilangan acak dengan pengambilan 200 iterasi pertama. Hasil iterasi dan

pengambilan *data* 1 sampai *data* 5 ditunjukkan pada Gambar 6.

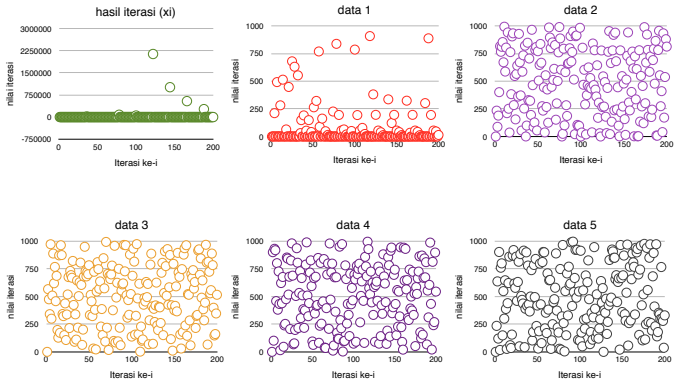
Hasil iterasi dari $x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$ secara visual terlihat membentuk sebuah pola, walaupun sudah terpecah dalam 4 garis namun kecendrungan grafiknya menuju pada titik tertentu. Pola dari hasil iterasi juga terlihat pada *data* 1 belum membentuk *chaos*. *data* 2 sudah diperoleh 91 nilai yang berbeda. Deretan bilangan yang membentuk *chaos* adalah *data* 3, *data* 4, dan *data* 5. Ketiga data tersebut sudah tentu dapat dijadikan sebagai kunci dalam kriptografi, karena dari kunci sudah menunjukkan kondisi keacakan.



Gambar 6: Fungsi $x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$, dengan $x_0 = 0,5$

$x_i = (20 - x_{i-1}^2 - 19x_{i-1})/x^2$ adalah fungsi iterasi yang menghasilkan bilangan acak dengan skema CSPRNG *chaos* selain *fungsi* 4. Pengambilan bilangan untuk *data* 1 sampai *data* 5 diberikan pada Gambar 7. Tidak ada yang berbeda dengan fungsi sebelumnya dalam menghasilkan bilangan acak, hanya saja hasil iterasi ada beberapa yang negatif, hasil ini terlihat pada grafik. Fungsi iterasi ini juga dapat menghasilkan bilangan acak berbasis CSPRNG *chaos* yang lebih banyak daripada fungsi lain di polinomial derajat-3.

Hasil berbeda pada fungsi $x_i = \sqrt[3]{20 - 6(x_{i-1})^2 - 19x_{i-1}}$, adalah satu-satunya fungsi yang mempunyai akar pangkat tiga. Fungsi ini hanya terdefenisi untuk $-4 \leq x \leq 0,833$ sehingga ketika digunakan sebagai fungsi itersi tidak akan berhasil karena banyak bilangan yang tidak terdefenisi. Bila diperhatikan kesamaan dari dua fungsi yang menghasilkan bilangan acak berbasis CSPNRG *chaos* merupakan fungsi rasional, dimana penyebutnya adalah sebuah fungsi. Sehingga perubahan yang terjadi pada pembilang dan penyebut secara bersamaan, maka perbandingan dari keduanya akan tetap terjaga dan akan selalu mengahasilkan bilangan desimal yang unik. Fungsi yang berbasis pada akar tidak berhasil untuk menjadi fung-



Gambar 7: Fungsi $x_i = (20 - 6x_{i-1}^2 - 19x_{i-1})/x_{i-1}^2$ dengan $x_0 = 0,5$

si pembangkit, karena tidak semua nilai x akan terdefenisi. Fungsi yang tidak berhasil untuk mendapatkan bilangaan berbasis CSPRNG *chaos*, tetapi berhasil menjadi fungsi iteratif adalah fungsi yang hanya membagi sebuah nilai yang tetap seperti pada *fungsi* 2 pada polinomial derajat-3.

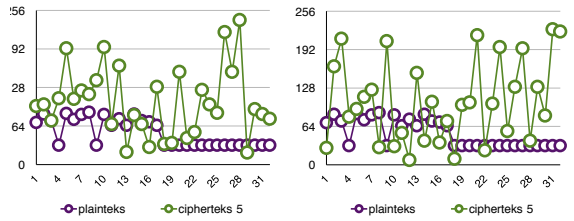
4.5. Pengujian pada Kriptografi *Block Cipher*

Fungsi iterasi yang menghasilkan bilangan acak berbasis CSPNRG *chaos* digunakan sebagai kunci dalam *block cipher*. Pengujian dilakukan dengan proses enkripsi $E_k : P + K = C \bmod (256)$, dimana P, C, dan K secara berturut-turut adalah plainteks, cipherteks dan kunci.

Diambil plainteks FTI UKSW SALATIGA, dan digunakan ukuran *block* sebesar 256 bit atau sebanding dengan 32 karakter. Simulasi dilakukan dengan memilih $x_i = (16x_{i-1} + 5)/17$, khususnya pada *data* 3 untuk menjadi kunci. Riset ini juga mengolah inputan berupa karakter yang kemudian menjadi nilai inisialisasi x_0. Proses dilakukan mengubah karakter inputan ke ASCII, kemudian dilakukan proses perkalian sehingga memperoleh sebuah nilai. Inisialisasi $x_0 = 0,024988457987$ yang digunakan pada derajat-1 dan derajat-2 adalah hasil olahan dari inputan 'jam 9 pm'.

Kekuatan dari CSPRNG *chaos* adalah uji *butterfly effect*, oleh karena itu pada Gambar 8 dilakukan perbandingan inisialisasi antara $x_0 = 0,024988457987$ dengan $x_0 =$ 0,024919429098 yang diperoleh dari inputan berbeda 1 bit 'jam 9 pn'. Hasil yang diperoleh adalah perbedaan sangat signifikan pada bilangan acak karena perbedaan inisialisasi 1 bit. Hal ini terlihat dari berbedanya grafik cipherteks. Dengan

demikian *butterfly effect* terjadi pada fungsi pembangkit, dan mengakibatkan algoritma enkripsi juga memberikan efek konfusi karena dapat menyamarkan pola dari plainteks berdasarkan kunci.



Gambar 8: Enkripsi dengan Inisialisai Berbeda pada $x_i = (16x_{i-1} + 5)/17$

Pengujian korelasi dilakukan untuk melihat kekuatan kunci dalam proses enkripsi sehingga dapat diketahui seberapa baik kunci dapat mangamankan informasi pada *block cipher*. Korelasi digunakan untuk melihat algoritma (dalam penelitian ini hanya kunci) mampu membuat plainteks dan cipherteks tidak berhubungan atau mendekati 0. Hasil korelasi negatif maupun positif tidak terlalu diperhatikan, yang dilihat hanya seberapa dekat dengan 0.

Tabel 6 adalah hasil uji korelasi antara plainteks dan cipherteks berdasarkan fungsi iterasi. Kolom korelasi-1 digunakan $x_0 = 0,024988457987$, sedangkan korelasi-2 digunakan $x_0 = 0{,}024919429098$. Hanya saja pada polinomial derajat-3, secara berturut-turut $x_0 = 0,5$ dan $x_0 = 0{,}05001$. Hasil korelasi untuk tiap fungsi cukup bervariasi, tetapi selalu $< 0,5$, tetapi secara rata-rata diperoleh 0,208 untuk korelasi-1 sedangkan 0,215 untuk korelasi-2 (tanda negatif diabaikan karena dalam kriptografi tidak memberikan pengaruh). Hasil ini berdasarkan klasifikasi korelasi pada Tabel 1, maka keduanya berada dalam kategori 'rendah', sehingga pembangkitan bilangan acak dapat membuat plainteks dan cipherteks tidak berhubungan secara statistik. Hasil ini memberikan informasi bahwa kekuatan kunci menjamin bahwa kriptanalisis akan kesulitan untuk mencari sifat konsisten kunci, dengan mengubah inputan berdasarkan keseragaman pola. Naik turun nilai korelasi menunjukkan sensitifitas fungsi pembangkit terhadap inisialisasi.

Algoritma kunci yang dirancang dapat menghasilkan bilangan acak yang dapat menjamin kompleksitas waktu dan ruang apabila kriptanalisis melakukan kriptanalisis dengan mencoba semua kemungkinan kunci. Dengan diperoleh bilangan yang acak dari fungsi sederhana tentunya akan membuat algoritma yang sederhana juga, sehingga memungkinkan bagi para kriptografer untuk merancang algoritma *One Time Pad* sebagai sebuah kriptosistem yang tidak dapat dipecahkan atau *unbreakable cipher*. Hasil dari penelitian ini juga merupakan bagian kecil

yang dimulai untuk menuju pada kriptosistem tersebut. Secara keseluruhan seti-

Tabel 5: Hasil Pengujian Korelasi pada *Block Cipher* 256 Bit

Fungsi Iterasi	**Kunci**	**Korelasi-1**	**Korelasi-2**
$x_i = (16x_{i-1} + 5)/17$	*data* 3	0,084	-0.460
$x_i = (16x_{i-1} + 5)/17$	*data* 4	0,257	-0.131
$x_i = (16x_{i-1} + 5)/17$	*data* 5	0,134	-0.007
$x_i = (x_{i-1}^2 - 3)/2$	*data* 3	$0,090$	-0.460
$x_i = (x_{i-1}^2 - 3)/2$	*data* 4	$-0,395$	-0.131
$x_i = (x_{i-1}^2 - 3)/2$	*data* 5	$-0,030$	-0.007
$x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$	*data* 2	$-0,150$	0.297
$x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$	*data* 3	$-0,358$	-0.283
$x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$	*data* 4	$-0,035$	-0.143
$x_i = x_{i-1} - (x_{i-1}^2 - 3)/176$	*data* 5	0,487	0.070
$x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$	*data* 3	$-0,451$	0.384
$x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$	*data* 4	$-0,263$	0.385
$x_i = (20 - x_{i-1}^3 - 19x_{i-1})/6x_{i-1}$	*data* 5	$-0,139$	-0.248
$x_i = (20 - 6x_{i-1}^2 - 19x_{i-1})/x_{i-1}^2$	*data* 2	$-0,042$	0.098
$x_i = (20 - 6x_{i-1}^2 - 19x_{i-1})/x_{i-1}^2$	*data* 3	$-0,419$	-0.031
$x_i = (20 - 6x_{i-1}^2 - 19x_{i-1})/x_{i-1}^2$	*data* 4	0,079	-0.425
$x_i = (20 - 6x_{i-1}^2 - 19x_{i-1})/x_{i-1}^2$	*data* 5	$-0,134$	-0.101

ap fungsi yang diambil mewakilkan polinomial derajat-1, derajat-2, dan derajat-3, belum dapat memberikan hasil iterasi dalam bilangan yang acak secara langsung, tidak seperti fungsi logistik dari Lorentz. Tetapi proses yang dilakukan dengan pengambilan *integer* pada mantissa yang membuat bilangan acak berbasis CSPRG *chaos* dapat diperoleh. Hasil ini sangat baik untuk perkembangan pencarian bilangan acak, karena tidak mudah untuk menemukan fungsi pembangkit seperti fungsi Lorentz. Pemilihan konstanta dan koefisien pada fungsi serta manipulasi aljabar dengan FPI untuk mengahasilkan fungsi iteratif menjadi sangat penting dalam menghasilkan CSPNRG *chaos*.

5. Kesimpulan

Simpulan yang dapat diambil dari penelitian regenerasi fungsi polinomial dalam rancangan algoritma berbasis CSPNRG *chaos* sebagai pembangkit kunci adalah: 1) Setiap fungsi polinomial derajat-1, derajat-2, dan derajat-3 dapat digunakan sebagai fungsi pembangkit, tetapi diperlukan pemilihan koefisien dan konstanta yang

tepat dan juga ketangkasan dalam proses manipulasi aljabar pada FPI. 2) Bilangan acak CSPNRG *chaos* tidak hanya dapat dibangkitkan menggunakan fungsi Lorentz, tetapi juga dengan meregenerasi fungsi polinomial dengan berbagai derajat. 3) Secara algoritma, pengambilan *integer* dari mantissa merupakan proses yang ampuh karena dapat menghasilkan bilangan acak berbasis CSPRNG *chaos* walupun hasil dari fungsi iterasi tidak dapat menghasilkan bilangan acak. 4) Pengujian kriptosistem menunjukkan rata-rata korelasi berada pada kategori 'rendah', hal ini menunjukkan penggunaan kunci dengan fungsi polinomial sebagai regenerator dapat membuat plainteks dan cipherteks tidak berhubungan secara statistik. 5) Algoritma kunci yang dirancang dapat menghasilkan bilangan acak yang dapat menjamin kompleksitas waktu dan ruang apabila kriptanalisis melakukan kriptanalisis dengan mencoba semua kemungkinan kunci. 6) Algoritma yang dirancang adalah bagian kecil yang dapat memberikan kontribusi bagi para kriptografer, dalam mengembangkan proses pembangkitan kunci berbasis *One Time Pad* sebagai sebuah kriptosistem yang tidak dapat dipecahkan atau *unbreakable cipher*.

Pustaka

[1] Steward, J., *Calculus; Early Transcendentals*, Belmont: Brooks/Cole, 2012.

[2] Anton, H., *Calculus*, New York: John & Willey , 2012.

[3] Fourouzan, B., *Cryptography and Network Security*, New York: Mc Graw Hill, 2008.

[4] Munir, R., *Kriptografi*, Bandung: Informatika, 2005.

[5] Chapra, S. & Canale, R., *Numerical Methoods for Engineers*, Sixth Edition, New York: Mc Graw Hill, 2010.

[6] Montgomery, D.C. & Runger, G.C., *Applied Statistics and Probability for Engineers*, Third Edition, New York: John Wiley & Sons, 2003.