JIKO (JURNAL INFORMATIKA DAN KOMPUTER)
September 2024, Volume: **8**, No. **2** | Pages 367–379
doi: 10.26798/jiko.v8i2.1313
e-ISSN : 2477-3964 – p-ISSN : 2477-4413

ARTICLE

Optimasi Pembangkit Bilangan Acak Dengan Fungsi Polinomial dan Kombinasi Metode Iterasi

Optimizing Random Number Generator With Polynomial Function And Combination of Iteration Methods

Siska Angelina dan Alz Danny Wowor*

Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana, Salatiga, Indonesia

*Penulis Korespondensi: alzdanny.wowor@uksw.edu

(Disubmit 24-04-25; Diterima 24-05-13; Dipublikasikan online pada 24-09-05)

Abstrak

Penelitian ini menggunakan fungsi polinomial dan memadukan metode iterasi untuk menghasilkan bilangan acak. Simulasi dengan nilai awal $x_0 = -1.4$ yang memiliki nilai optimal dan menghasilkan empat nilai yang dapat dijadikan kunci. Hasil visual merepresentasikan bahwa setiap kumpulan data menghasilkan bentuk grafik yang *chaotic* dan diasumsikan acak berdasarkan uji statistik yaitu *run test*, *mono bit* dan *block bit*. Setiap nomor acak yang dihasilkan melewati uji kriptografi dan dapat menghasilkan *plaintext* dan *ciphertext* yang independen secara statistik. Fungsi polinomial ini telah terbukti sebagai pembangkit bilangan acak yang efektif dan bisa digunakan untuk memperoleh bilangan acak bersifat CSPRNG berbasis *chaos*.

Kata kunci: Fungsi polinomial; Newton Raphson; Fixed Point Iteration; CSPRG Chaos

Abstract

This research uses a polynomial function and combines the iteration method to generate random numbers. Simulation with an initial value of $x_0 = -1.4$ which has an optimal value and produces four values that can be used as keys. The visual results represent that each data set produces a graphical shape that is chaotic and is assumed to be random based on statistical tests namely run test, mono bit and block bit. Each random number generated passes the cryptographic test and can produce statistically independent plaintext and ciphertext. This polynomial function has been proven to be an eective random number generator and can be used to generate chaos-based CSPRNG random numbers.

KeyWords: Polynomial Function; Newton Raphson; Fixed Point Iteration; CSPRG Chaos

1. Pendahuluan

Pembangkit bilangan acak merupakan salah satu algoritma yang digunakan untuk menghasilkan kunci yang baik. Setiap kunci yang baik memberikan sebuah deretan angka acak sehingga dapat menciptakan relasi antara plainteks dan cipherteks sulit untuk diprediksi, dengan demikian kunci tersebut menjadi optimum dan lebih menjamin keamanan sebuah informasi. Sebuah fungsi yang bisa digunakan sebagai pembangkit bilangan acak adalah fungsi yang menghasilkan bilangan acak yang bersifat *Cryptographically Secure Pseudorandom Generator* (CSPNRG) berbasis *Chaos* sehingga bilangan yang dihasilkan tidak dapat diprediksi dengan mudah karena sensitif terhadap perubahan input yang kecil sehingga memberikan efek difusi dan konfusi pada algoritma[1][2][3][4]

How to Cite: S. Angelina *et al.*, "Optimasi Pembangkit Bilangan Acak Dengan Fungsi Polinomial dan Kombinasi Metode Iterasi", *JIKO (JURNAL INFORMATIKA DAN KOMPUTER)*, Volume: **8**, No.**2**, Pages 367–379, September 2024, doi: 10.26798/jiko.v8i2.1313.

Penelitian [5] mendesain pembangkit kunci *block cipher* berbasis CSPRNG Chaos menggunakan fungsi trigonometri dan menggunakan *Fixed Point Iteration* (FPI) untuk mengubah fungsi menjadi beberapa fungsi iterasi. Dari hasil penelitian tersebut diperoleh bahwa fungsi trigonometri dapat digunakan sebagai pem- bangkit dalam menghasilkan kunci pada kriptogra *block cipher*. Penelitian selanjutnya [6] memilih $30x^{3}$˘$19x$˘6 sebagai fungsi pembangkit bilangan acak dan menggunakan metode Newton Raphson untuk membuat polinomial menjadi fungsi iterasi. Dari hasil penelitian tersebut dipeoleh bahwa fungsi tersebut menghasilkan angka acak yang sukses dan dapat digunakan untuk menghasilkan angka acak berdasarkan CSPRNG *Chaos*.

Penelitian terkait selanjutnya [7] dengan pembangkitan kunci *Beaufort Cipher* dengan teknik blum-blum shub pada pengamanan citra digital, lalu memperoleh sebuah kunci yang dibangkitkan berdasarkan teknik pembangkitan *Blum-Blum Shub* menghasilkan kunci yang acak dan tidak menyebabkan kunci yang berulang. Penelitian [8] merancang pembangkit bilangan acak pendekatan LFSR dengan skema A5/1 dan empat fungsi umpan balik untuk proses pembangkit bilangan acak. Setiap fungsi umpan balik digunakan operasi XOR untuk menentukan nilai bit yang baru pada iterasi berikutnya. Hasil penelitian tersebut menghasilkan kunci yang baik dan keluaran acak yang sangat aman untuk digunakan dalam sebuah Stream Cipher.

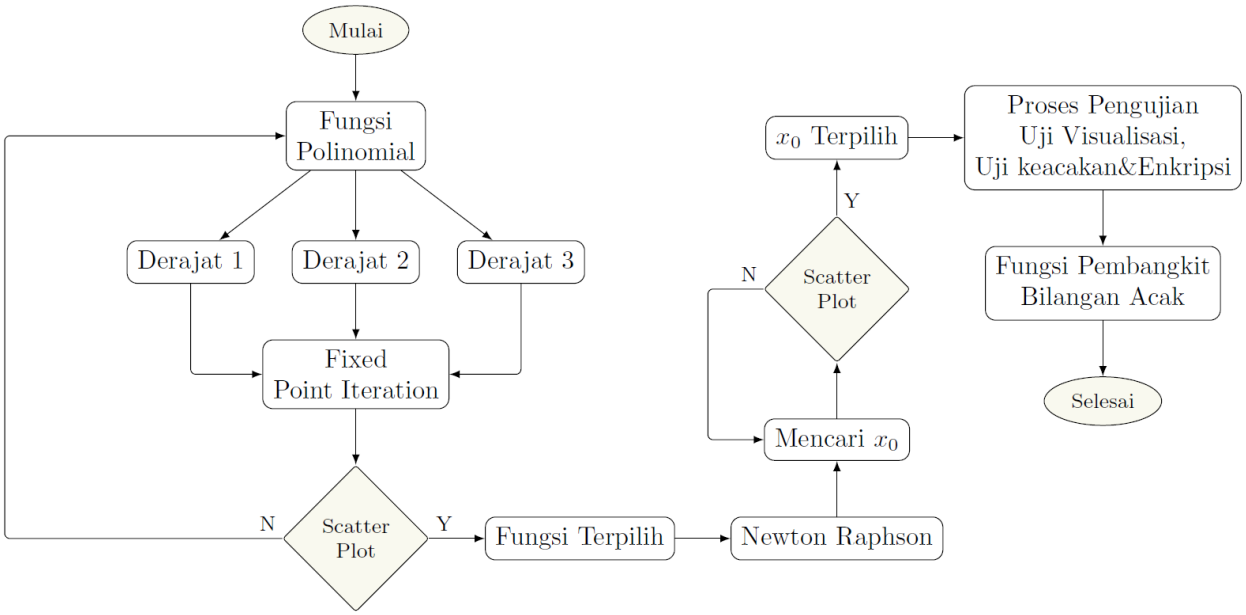
Penelitian [9] mengimplementasikan algoritma *Lucifer* dengan pembangkit kunci *Multiply With Carry Generator* untuk mengamankan data citra digital. Hasil dari penelitian ini, algoritma *Multiply With Carry Generator* sangat optimal dalam pengacakan kunci karena karakter bilangan yang dihasilkan sangat sulit diketahui polanya dan sangat minim perulangan bilangan yang sama, dalam pengimplementasian algoritma *Lucifer* dengan pembangkit kunci *Multiply With Carry Generator* dalam mengamankan citra digital menghasilkan nilai Pixel yang berbeda sehingga sulit bagi penyerang untuk memecahkannya.

Penelitian ini mencari fungsi polinomial yang digunakan sebagai fungsi pembangkit bilangan acak berbasis CSPRNG *Chaos* dengan meregenerasikan setiap fungsi polinomial menggunakan metode *Fixed Point Iteration* (FPI), dilakukan pengujian visualisasi menggunakan scatter plot untuk memilih fungsi yang menghasilkan grafik tanpa pola yang jelas. Selanjutnya meregenerasikan kembali fungsi yang terpilih menggunakan metode Newton Raphson, diikuti dengan pengujian visualisasi menggunakan *scatter plot* kemudian dilakukan pengujian statistika untuk memastikan bahwa rangkaian bilangan acak yang dihasilkan oleh fungsi tersebut memenuhi kriteria CSPRNG dengan sifat *Chaos* yang diharapkan. Kemudian yang terakhir dilakukan pengujian enkripsi untuk memastikan bahwa kunci yang dihasilkan oleh fungsi tersebut memberikan efek difusi dan konfusi yang diinginkan pada proses enkripsi.

Penelitian ini bertujuan untuk mendapatkan fungsi pembangkit bilangan dan menghasilkan bilangan acak berbasis CSPNRG *Chaos*. Perbedaan penelitian ini dengan penelitian sebelumnya adalah penelitian sebelumnya hanya menggunakan satu metode iterasi yaitu *Fixed Point Iteration* (FPI) atau metode Newton Raphson. Pada penelitian ini menggunakan kombinasi dua metode iterasi sekaligus yaitu metode *Fixed Point Iteration* dan metode Newton Raphson guna menghasilkan sebaran data acak sehingga memiliki grafik yang tidak membentuk pola, sehingga diharapkan penelitian ini dapat berguna bagi perkembangan kriptogra dalam pembangkit bilangan acak berbasis CSPNRG *Chaos* yang sulit sehingga tidak dapat diprediksi dengan mudah oleh Kriptonalisis.

2. Metode

Dalam penelitian ini, berbagai langkah metodologis dijalankan untuk mencapai tujuan penelitian yang ditetapkan, yang secara umum dijelaskan dalam sebuah bagan pada Gambar 1. Langkah pertama adalah mengidentikasi fungsi-fungsi polinomial dengan derajat 1, 2, dan 3. Setelah itu, setiap fungsi polinomial yang dipilih akan diubah menjadi fungsi iterasi menggunakan metode *Fixed Point Iteration* (FPI). Visualisasi dari fungsi-fungsi polinomial tersebut dievaluasi menggunakan *scatter plot* dan kemudian digunakan sebagai *generator* angka acak. Oleh karena itu, penting untuk melakukan proses penentuan nilai awal yang tepat x_0 sebagai inisialisasi dalam proses ini. Melakukan pencarian nilai x_0 melalui pendekatan *trial-and-error* pada fungsi yang telah diubah menjadi fungsi iterasi, diikuti dengan pencarian rata-rata nilai *p-value* tertinggi untuk setiap x_0.



Gambar 1. Skema Umum Proses Penelitian

Metode Newton Raphson digunakan untuk mengubah fungsi tersebut menjadi fungsi iterasi dan menerima x_0 sebagai input untuk menentukan jumlah iterasi yang berbeda yang dihasilkan dari proses iterasi. Pengujian iterasi melibatkan pencarian nilai x_0 yang dapat menghasilkan sejumlah besar iterasi yang beragam. Setiap nilai awal yang dipilih dianggap sebagai input optimal dalam fungsi, yang kemudian dapat digunakan sebagai kunci untuk menghasilkan variasi bilangan yang berbeda.

Pengujian setiap angka acak merupakan tahap akhir untuk memastikan apakah angka-angka yang dihasilkan sesuai dengan standar angka acak berbasis CSPRNG atau tidak. Uji visual menggunakan *Scatter Plot* bertujuan untuk mengevaluasi pola dalam sebaran data. Uji keacakan melibatkan pengujian statistik menggunakan metode seperti *run test*, *mono bit* dan *block bit*. Langkah terakhir adalah menggunakan setiap angka acak sebagai kunci dalam proses enkripsi.

Langkah metodologis dalam penelitian ini dijelaskan juga dalam sebuah Pseudocode yang ditunjukkan pada Algoritma 1. Terlihat bahwa fungsi polinomial yang akan digunakan sebagai pembangkit bilangan acak melaui dua kali pengubahan fungsi iterasi, yaitu dengan menggabungkan dua metode iterasi: metode *Fixed Point Iteration* (FPI) dan metode Newton Raphson untuk memperoleh bilangan yang acak.

Algoritma 1 Skema Umum Proses Penelitian

```
Mulai
Pilih Fungsi Polinomial
i ← 1
for i ≤ 3 do
    Fungsi Polinomial derajat-i
    Fixed Point Interation
    if ScatterPlot!=Acak then
    else Sebagai Fungsi Terpilih
    end if
end for
Fungsi Terpilih
Newton Raphson
Mencari x0
if ScatterPlot!=Acak then
    Kembali mencari x0
end if
x0 terpilih
Lakukan Proses Pengujian: Uji visualisasi, Uji keacakan, dan Enkripsi
Fungsi Pembangkit Bilangan Acak
Selesai
```

3. Hasil

3.1 Pencarian Fungsi Iterasi

Proses pencarian pembangkit bilangan acak diterapkan pada fungsi polinomial derajat-1, derajat-2, dan derajat-3. Proses ini mengikuti langkah-langkah yang ditunjukkan pada Gambar 1 Setiap fungsi polinomial yang dipilih diubah menjadi fungsi iterasi menggunakan metode *Fixed Point Iteration* (FPI). Tujuan dari FPI adalah mencari akar-akar suatu fungsi secara sederhana dengan melakukan iterasi penggunaan rumus tertentu hingga diperoleh nilai yang konvergen[10]. Hal ini akan menghasilkan angka yang akan digunakan sebagai kunci dalam proses iterasi.

Tabel 1. Fungsi Polinomial yang Dipilih

No	Derajat Fungsi	Fixed-Point Iteration	Fungsi Iterasi
1	$f(x) = x - 2$	$x = \frac{(15x+2)}{16}$	$x_i = \frac{(15x_{i-1}+2)}{16}$
2	$f(x) = x^2 - 4$	$x = \frac{(x-(x^2-4))}{2}$	$x_i = \frac{(x_{i-1}-(x_{i-1}^2-4))}{2}$
3	$f(x) = x^3 + 4x^2 + 7x + 7$	$x = \frac{(7-x^3-4x^2)}{7}$	$x_i = \frac{(7-x_{i-1}^3-4x_i^2)}{7}$

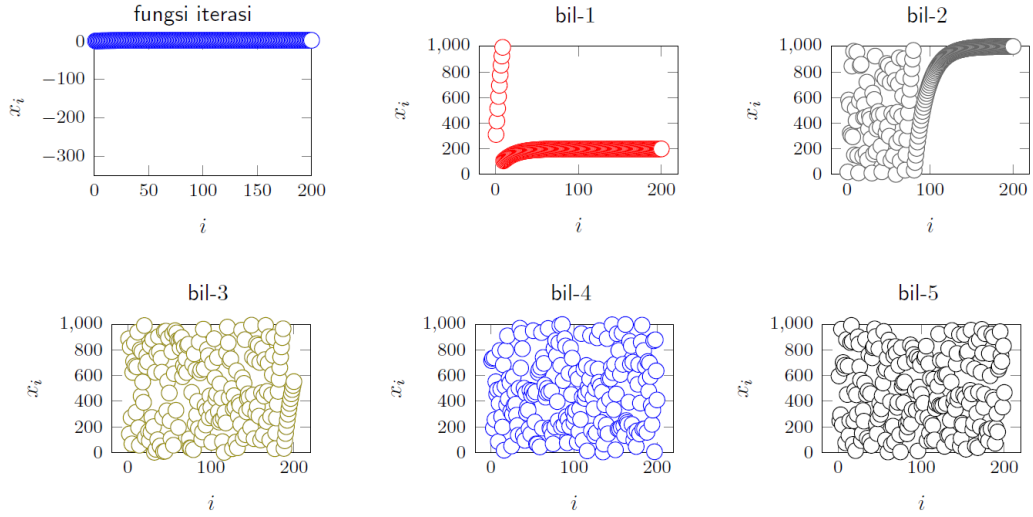
Tabel 1 menunjukkan fungsi polinomial yang dipilih, pada fungsi tersebut akan dilakukan pengujian visualisasi menggunakan Scatter Plot yang bertujuan untuk melihat apakah sebaran datanya membentuk pola hubungan atau tidak dari hasil tersebut[11]. Dengan menggunakan Scatter Plot untuk mengacak sebaran bilangan dilakukan dengan menginisialisasi nilai x_0. Inisialisasi nilai x_0 yang dipilih adalah 0.198421213565437. Apabila fungsi tersebut dapat menghasilkan urutan bilangan dan lolos uji visualisasi, maka fungsi tersebut dapat dilanjutkan dengan metode Newton Raphson untuk mengubah fungsi sehingga menjadi fungsi iterasi[12].

Tabel 2. Hasil 10 Iterasi Pertama dari $x_i = \frac{(15x_{i-1}+2)}{16}$

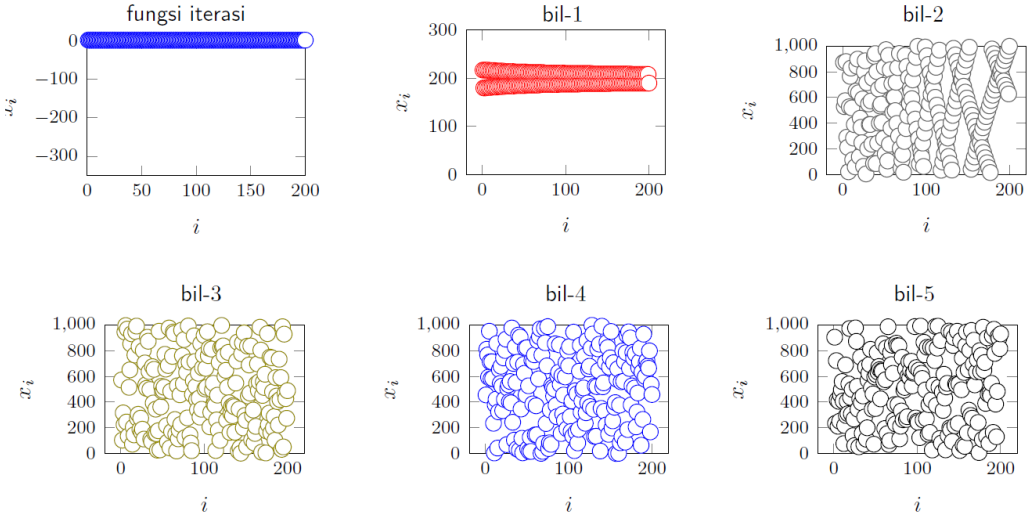
i	x_i	bil-1	bil-2	bil-3	bil-4	bil-5
1	0.311019887717597	311	19	887	717	597
2	0.416581144735247	416	581	144	735	247
3	0.515544823189294	515	544	823	189	294
4	0.608323271739963	608	323	271	739	963
5	0.695303067256216	695	303	67	256	216
6	0.776846625552702	776	846	625	552	702
7	0.853293711455658	853	293	711	455	658
8	0.924962854489680	924	962	854	489	680
9	0.992152676084050	992	152	676	84	75
10	1.055143133828820	105	514	313	382	882

Tabel 2 adalah hasil 10 Iterasi Pertama dari $x_i = \frac{15x_{i-1}+2}{16}$ dengan nilai x_0 = 0.198421213565437. Tabel 2 juga menunjukkan bagaimana mengambil integer dari mantissa. Hasil iterasi memiliki 15 digit sehingga memungkinkan terdapat 5 kunci yang berasal dari 5 bilangan. Jumlah karakter ASCII maksimum adalah 256, sehingga setiap iterasi menghasilkan 3 digit. Jika diambil 3 angka mantissa berdasarkan urutan 1, 2, 3 maka bil-nya adalah 1, selanjutnya 4, 5, 6 menjadi bil-2, dan sampai urutan 13, 14, dan 15 menjadi bil-5. Hasil dari 200 iterasi pertama dengan fungsi $x_i = \frac{15x_{i-1}+2}{16}$ digambarkan dalam sebuah grak *Scatter Plot* diberikan pada Gambar 2.

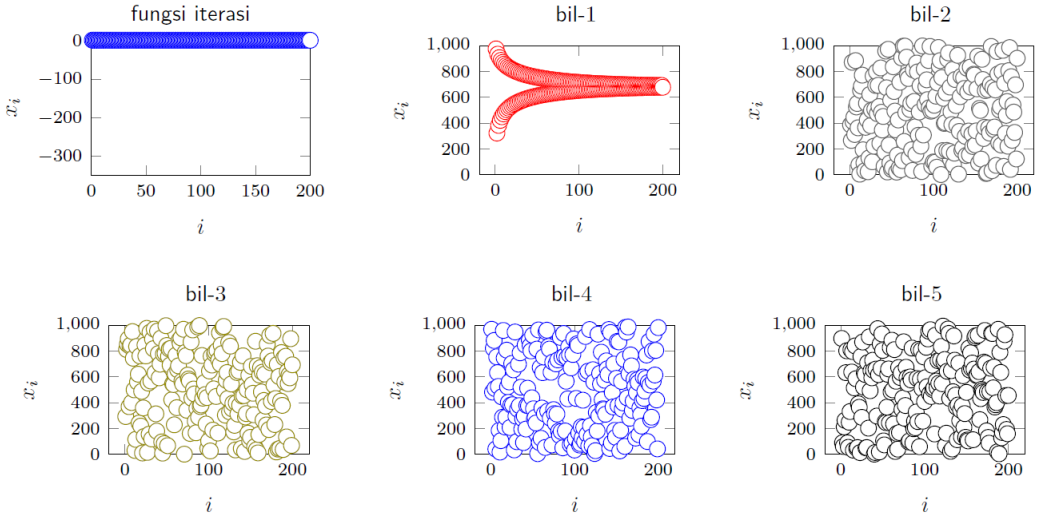
Hasil iterasi dari $x_i = \frac{15x_{i-1}+2}{16}$ seperti yang terlihat di Gambar 2, menunjukkan secara visual angka-angka yang terus membentuk pola dan yang tidak. Oleh karena itu, bilangan 3, 4, dan 5 berhasil melewati uji visualisasi, tetapi bilangan 1 dan 2 gagal dalam uji visualisasi. Grafik *Scatter Plot* yang menunjukkan hasil 200 iterasi pertama dari $x_i = \frac{x_{i-1}-(x_{i-1}^2-4)}{2}$ pada Gambar 3 berhasil menunjukkan pola acak dari bilangan 2, 3, 4, dan 5, namun bil 1 gagal dalam uji visualisasi.



Gambar 2. Scatter Plot 200 Iterasi Pertama dari $x_i = \frac{15x_{i-1}+2}{16}$



Gambar 3. Scatter Plot 200 Iterasi Pertama dari $x_i = \frac{x_{i-1}-(x_{i-1}^2-4)}{2}$



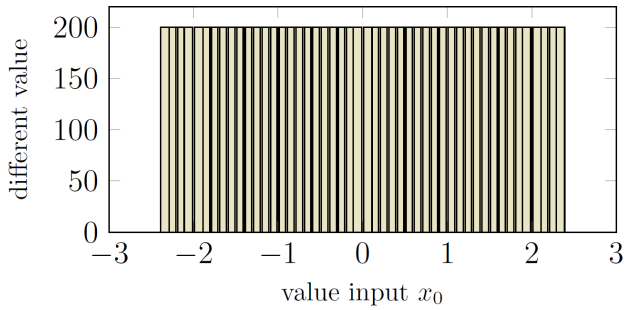
Gambar 4. Scatter Plot 200 Iterasi Pertama dari $x_i = \frac{(7-x_{i-1}^3-4x_i^2)}{7}$

Hasil dari 200 iterasi pertama dengan fungsi $x_i = \frac{(7-x_{i-1}^3-4x_i^2)}{7}$ digambarkan dalam sebuah grafik *Scatter Plot* diberikan pada Gambar 4, diperoleh bahwa fungsi tersebut berhasil menghasilkan grafik *Scatter Plot* yang sudah acak, kecuali untuk bil 1 yang gagal karena membentuk pola.

Hasil pengujian dari 3 fungsi polinomial di atas yang sebelumnya dipilih telah diuji menggunakan *Scatter Plot*. Hasilnya menunjukkan bahwa $f(x) = x^{2\check{}}4$ merupakan fungsi yang terpilih karena graknya tidak membentuk pola. Iterasi yang dihasilkan oleh metode Newton Raphson berperan sebagai sebuah alat untuk mendekati akar atau solusi dari sebuah persamaan secara berulang-ulang. Metode ini memanfaatkan turunan dari fungsi yang diberikan untuk memperkirakan nilai akar yang lebih mendekati secara berulang hingga mendapatkan hasil yang diinginkan. Metode ini biasanya digunakan untuk menemukan solusi numerik dari persamaan non-linear[12]. Menerapkan metode Newton-Raphson untuk memilih fungsi dapat menghasilkan pembangkit bilangan acak yang dapat diandalkan, serta memungkinkan visualisasi Scatter Plot yang dihasilkan. *Scatter Plot* ini mewakili sebaran angka untuk menciptakan kesan keacakan, namun proses ini juga melibatkan inisialisasi nilai x_0.

3.1.1 Pencarian Nilai Inisialisasi

Salah satu faktor penting dalam pembangkitan bilangan acak adalah nilai awal dari x_0. Hal ini karena pencarian nilai awal yang tepat sangat penting untuk suatu fungsi tertentu[6]. Dengan mengganti setiap nilai x_0 dalam fungsi iterasi, maka dapat mengetahi jangkauan fungsi yang sesuai untuk mencapai rentang hasil yang optimal, seperti yang ditunjukkan pada Gambar 5, metode *trial-and-error* digunakan, berbagai nilai x_0 dicoba untuk mencapai luaran yang diinginkan[13].



Gambar 5. Input x_0 terhadap fungsi $x_{i+1} = x_i - \frac{x_{i-1}-(x_{i-1}^2-4)}{2}$

Gambar 5 memvisualisasikan hubungan antara x_0 dan berbagai nilai yang diberikan dalam histogram x_0 dari rentang $\check{}2.8 \leq x_0 \leq 2.8$ dengan nilai 200 angka berbeda yang artinya fungsi $\frac{x-(x^2-4)}{2x}$ baik karena menghasilkan nilai-nilai yang optimal[6][14]. Untuk menemukan nilai x0 yang terbaik, perlu dilakukan pengujian dengan mencari nilai *p-value* yang tertinggi.

Tabel 3. Hasil rata-rata p-value

No	x0	Rata-rata	No	x0	Rata-rata	No	x0	Rata-rata
1	-2.8	0.65347451	19	-0.9	0.69457896	37	1.0	0.69574067
2	-2.7	0.61895790	20	-0.8	0.66912252	38	1.1	0.67771485
3	-2.6	0.61532619	21	-0.7	0.71202313	39	1.2	0.73600834
4	-2.5	0.67071440	22	-0.6	0.58367003	40	1.3	0.71739957
5	-2.4	0.65200689	23	-0.5	0.72497031	41	1.4	0.74416287
6	-2.3	0.69390241	24	-0.4	0.66163802	42	1.5	0.64113655
7	-2.2	0.67888133	25	-0.3	0.66817084	43	1.6	0.69819309
8	-2.1	0.65941566	26	-0.2	0.65826023	44	1.7	0.71166984
9	-1.9	0.73630514	27	-0.1	0.65201346	45	1.8	0.69093529
10	-1.8	0.69093529	28	0.1	0.65201346	46	1.9	0.73630514
11	-1.7	0.71166984	29	0.2	0.65826023	47	2.1	0.65941566
12	-1.6	0.69819309	30	0.3	0.66817084	48	2.2	0.67888133
13	-1.5	0.64113655	31	0.4	0.66163802	49	2.3	0.69390241
14	-1.4	0.74416287	32	0.5	0.72497031	50	2.4	0.65200689
15	-1.3	0.71739957	33	0.6	0.58367003	51	2.5	0.67071440

Table 3 lanjutan

16	-1.2	0.73600834	34	0.7	0.71202313	52	2.6	0.61532619
17	-1.1	0.67771485	35	0.8	0.66912252	53	2.7	0.61895790
18	-1.0	0.69574067	36	0.9	0.69457896	54	2.8	0.65347451

Berdasarkan hasil rata-rata *p-value* yang ditunjukkan Tabel 3, p-value terendah dimiliki oleh $x_0 = -0.6$ dan 0.6 dengan nilai rata-rata 0.58367003 dan *p-value* tertinggi dimiliki oleh $x_0 = -1.4$ dan 1.4 dengan nilai rata-rata 0.74416287. Secara statistika[15], jika $p-value > \alpha$ maka dapat dikategorikan baik, karena mempunyai selisi yang tinggi. Oleh karena itu, x_0 yang terpilih adalah −1.4 atau 1.4.

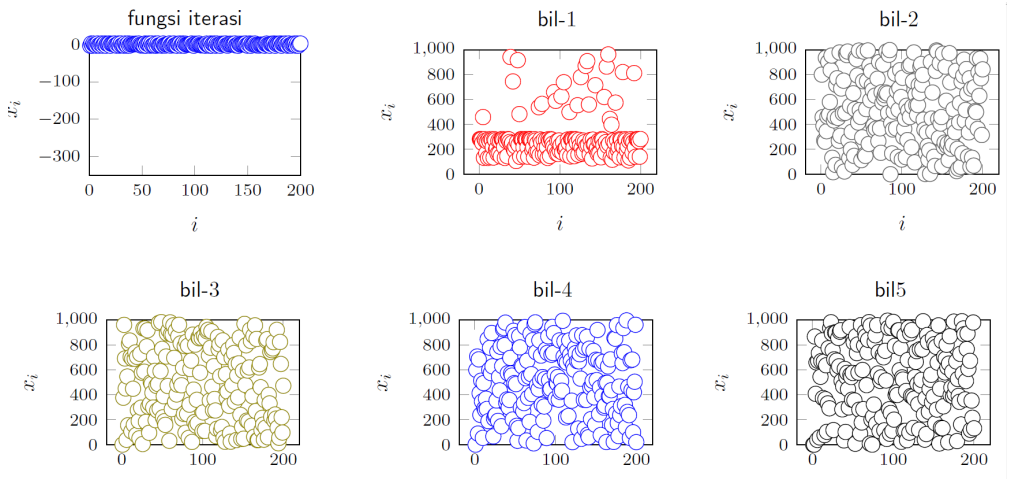
3.1.2 Iterasi Fungsi Untuk x_0 Terpilih

Nilai inisialisasi $x_0 = -1.4$ dijadikan sebagai sampel dalam pengujian ini. Pengujian awal menguji apakah suatu fungsi menggunakan metode Newton Raphson, yaitu $\frac{x-(x^2-4)}{2x}$ dengan inisialisasi $x_0 = -1.4$ dapat menghasilkan bilangan acak bersifat (CSPNRG) berbasis Chaos atau tidak. Sepuluh iterasi pertama dari proses ini ditunjukkan pada Tabel 3 dalam setiap iterasi, tiga digit pada bagian mantisa diambil dan masing-masing disebut sebagai bil-1, bil-2, bil-3, bil-4, dan bil-5.

Tabel 4. Hasil 10 Iterasi Pertama dengan $x_0 = -1.4$

i	xi	bil-1	bil-2	bil-3	bil-4	bil-5
1	-2.82800000000000	282	800	0	0	0
2	2.82458377600000	282	458	377	600	0
3	-2.79389962708868	279	389	962	708	868
4	2.52271692089408	252	271	692	89	408
5	-0.45926145168566	459	261	451	685	660
6	-1.32935039413038	132	935	39	413	38
7	-2.81345547239673	281	345	547	239	673
8	2.69463156514736	269	463	156	514	736
9	-1.69901811346058	169	901	811	346	58
10	-2.64480836054545	264	480	836	54	545

Kunci yang digunakan dalam kriptogra adalah bilangan bulat positif namun berdasarkan hasil dari 10 iterasi di atas menghasilkan bilangan negatif dan bilangan positif. Oleh karena itu, setiap bilangan negatif yang dihasilkan dari iterasi tersebut dapat diubah menjadi nilai absolut sehingga menghasilkan bilangan positif yang dapat digunakan sebagai kunci[14]. Gambar 6 menunjukkan hasil uji visualisasi menggunakan *Scatter Plot* untuk setiap angka.



Gambar 6. *Scatter Plot* 200 Iterasi Pertama dari $x_0 = -1.4$

Grafik Scatter Plot untuk bil-1 sampai bil-5 pada Gambar 6 menunjukkan hasil iterasi yang baik karena sebaran datanya tampak acak dan tidak membentuk pola tertentu. Untuk bil-1, pola sebarannya terlihat tidak seragam dibandingkan dengan bilangan lainnya. Penggunaan nilai inisialisasi $x_0 = -1.4$ berhasil menciptakan visualisasi grafik yang kacau. Hal ini menegaskan bahwa pendekatan dengan menggunakan metode pencarian bilangan yang berbeda dari sebelumnya memberikan hasil yang berbeda pula dapat membantu menunjukkan keberhasilan dalam pengujian visualisasi bilangan acak. Penting untuk melakukan uji keacakan statistik untuk menentukan apakah setiap angka yang dihasilkan dari inisialisasi $x_0 = -1.4$ dalam fungsi dapat dianggap sebagai angka acak berdasarkan CSPRNG *chaos* atau tidak.

3.2 Hubungan Setiap Bil

Hasil iterasi ditunjukkan pada Tabel 4 yang dibagi menjadi lima bagian untuk setiap kelompok. Ini sesuai dengan bil-i yang memiliki nilai dari 1 hingga 5. Semua iterasi berasal dari satu fungsi $\frac{x-(x^2-4)}{2}$ dengan inisialisasi $x_0 = -1.4$. Oleh karena itu, pengujian tambahan diperlukan untuk menentukan apakah setiap kumpulan bil berkorelasi secara statistik. Jika setiap kumpulan bil saling berkaitan erat, maka akan sulit untuk menggunakannya sebagai kunci pada kriptografi[6]. Untuk menguji hubungan pada setiap kumpulan bil, digunakan uji korelasi dengan melihat tingkat hubungan korelasi[16]. Apabila nilai korelasi yang dihasilka nmendekati 0 (nol), maka kedua variabel tersebut memiliki nilai kemiripan yang rendah (acak). Apabila nilai variabel yang dihasilkan mendekat 1 (satu), maka memiliki nilai kemiripan yang tinggi (tidak acak)[17]. Pengujian dilakukan pada 2 dari 5 kumpulan data yang dihasilkan, sehingga terdapat beberapa kombinasi uji ${}_2C_5 = 10$ yang harus dilaksanakan.

Tabel 5. Hasil Uji Korelasi

No	Relasi	Data	Hasil	Korelasi	Tingkat Hubungan
1	bil-1	–	bil-2	-0.11930312	sangat rendah
2	bil-1	–	bil-3	0.00226684	sangat rendah
3	bil-1	–	bil-4	-0.05144869	sangat rendah
4	bil-1	–	bil-5	-0.06381672	sangat rendah
5	bil-2	–	bil-3	0.05917566	sangat rendah
6	bil-2	–	bil-4	-0.01429950	sangat rendah
7	bil-2	–	bil-5	0.00616603	sangat rendah
8	bil-3	–	bil-4	-0.02487756	sangat rendah
9	bil-3	–	bil-5	0.07136208	sangat rendah
10	bil-4	–	bil-5	0.01417397	sangat rendah

Hasil uji korelasi ditunjukkan pada Tabel 5, secara keseluruhan nilai korelasi yang dihasilkan oleh setiap bil berada dalam rentang 0.00 - 0.199. Berdasarkan[16], range tersebut menunjukkan kekuatan hubun- gan yang sangat rendah. Oleh karena itu dengan menggunakan fungsi $\frac{x-(x^2-4)}{2x}$ dengan metode Newton Raphson dapat digunakan sebagai pembangkit kunci kriptografi, karena nilai yang diperoleh dari hasil perhitungan korelasi menunjukkan bahwa kumpulan bil secara statistika tidak terkait satu sama lain sehingga dapat mempersulit Kriptanalisis untuk menemukan kunci[18].

3.2.1 Pengujian Pengacakan

Terdapat beberapa metode pengujian secara statistika yang dapat digunakan berdasarkan[19] untuk keacakan dan bilangan acak semu pada pembangkitan bilangan acak yaitu Run Test, Mono Bit dan Block Bit. Uji keacakan bertujuan untuk memastiakan apakah ada hubungan antara dua variabel atau tidak[20]. Nilai uji keacakan yang digunakan adalah $\alpha = 1\%$, jika nilai *p-value* kurang dari $\alpha = 0.01$, maka dianggap tidak acak, dan sebaliknya dianggap acak[21]. Hasil pengujian dari lima bil, secara berurutan disajikan di Tabel 6 , Tabel 7 dan Tabel 8.

Tabel 6. Hasil Pengujian *Run-Test*

No	Data yang Diuji	p-value	Hasil Pengujian
1	bil-1	0.983479858619000	acak
2	bil-2	0.393179669139252	acak
3	bil-3	0.959171859760731	acak
4	bil-4	0.611624693609916	acak
5	bil-5	0.925459464452474	acak

Tabel 6 menunjukkan hasil uji setiap bil menggunakan metode *Run Test*. Secara keseluruhan, ketika semua nilai p-value lebih besar dari α, maka semua bil-i untuk i = 1,..., 5 dapat dikategorikan sebagai hasil acak. Dalam pengujian bil-1, nilai *p-value* adalah yang paling kecil, sehingga mendekati α dengan perbedaan yang kecil. Hasil ini juga sejalan dengan uji visualisasi sebelumnya. Meskipun termasuk dalam kategori acak, namun memiliki hasil yang tidak seoptimal dari yang lainnya.

Tabel 7. Hasil Pengujian *Mono Bit*

No	Data yang Diuji	p-value	Hasil Pengujian
1	bil-1	1.46482622396546	tidak acak
2	bil-2	0.54850623550015	acak
3	bil-3	0.31731050786291	acak
4	bil-4	1	acak
5	bil-5	0.42371079716679	acak

Hasil pada Tabel 7 dengan menggunakan metode pengujian *Mono Bit*, menunjukkan bahwa nilai *p-value* untuk bil-1 lebih kecil dari α = 1%. Dengan demikian dapat disimpulkan bahwa bilangan tersebut tidak memenuhi kriteria sebagai bilangan acak. Sementara itu, bilangan lainnya berhasil melewati pengujian acak dengan *p-value* yang signifikan karena menunjukkan perbedaan yang cukup besar.

Tabel 8. Hasil Pengujian *Block Bit*

No	Data yang Diuji	p-value	Hasil Pengujian
1	bil-1	1	acak
2	bil-2	1	acak
3	bil-3	1	acak
4	bil-4	1	acak
5	bil-5	1	acak

Tabel 8 menunjukkan hasil pengujian menggunakan metode *Block Bit*. Secara keseluruhan, masing-masing *p-value* yang diperoleh lebih besar dari α yang artinya setiap bilangan yang telah diuji termasuk dalam kategori yang acak dengan nilai *p-value* yang baik karena memiliki perbedaan yang signikan.

Hasil pengujian *Mono Bit* untuk kumpulan bil-1 tidak menunjukkan keacakan yang signikan, hal ini sesuai dengan hasil pengujian visualisasi dan *Run Test* yang menunjukkan data tersebut cenderung tidak acak. Penelitian ini telah menerapkan tiga metode uji keacakan. Dengan demikian, apabila setiap angka berhasil melewati kriteria keacakan dalam ketiga uji coba, maka angka tersebut dapat dianggap sebagai acak. Namun, bila salah satu atau ketiga ujiannya tidak berhasil, maka dianggap bukan acak. Sehingga, untuk inisialisasi $x_0 = -1.4$, bil-1 tidak berhasil dalam pengujian keacakan, dan oleh karena itu tidak disarankan untuk digunakan sebagai kunci.

Hasil uji keacakan Run Test, Mono Bit dan Block Bit menunjukkan bahwa kumpulan bilangan pada bil-2, bil-3, bil-4, dan bil-5 dapat dikategorikan sebagai acak. Hasil uji visualisasi sebelumnya juga menunjukkan

keberhasilan yang sama. Dengan begitu, ke-empat bil tersebut dapat diklasikasikan sebagai bilangan acak berbasis CSPRNG *chaos*.

3.3 Pengujian Skripsi

Setiap bil yang lulus uji akan dilakukan pengujian enkripsi dengan memilih beberapa bilangan sebagai kunci. Pengujian enkripsi bertujuan untuk menguji seberapa efektif algoritma enkripsi dalam melindungi data dari upaya dekripsi yang tidak sah[22]. Pengujian ini menggunakan *block cipher* dengan panjang kunci 256 bit (setara dengan 32 karakter) untuk mengenkripsi data dalam rentang ASCII, sehingga operasi dilakukan modulo 256. Proses enkripsi yang digunakan sederhana yaitu dengan menambahkan nilai *plaintext* (P) dan kunci (K) untuk menghasilkan *ciphertext* (C).

Pengujian enkripsi dilakukan dengan menggunakan tiga plainteks yang mewakili variasi yang biasanya digunakan oleh pengguna. Plainteks pertama berisi tulisan kombinasi huruf biasa yaitu nama lengkap saya adalah siska angelina.Biasanya dipanggil siska. Sementara plainteks yang kedua meliputi kombinasi huruf, simbol, dan angka yaitu NAm—A LeNGkaP 5ayA aD—a«LaĤ Si5kAA Ng3lina.B14«SaNya dIp A«N9gil 5iSka dan plainteks terakhir memiliki huruf yang sama semua kecuali huruf yang terakhir yaitu AAAA-AAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB.

Pengujian korelasi digunakan untuk menilai sejauh mana kunci yang digunakan mampu mengurangi atau menghilangkan korelasi antara plainteks dan cipherteks. Hal ini dilakukan untuk memastikan bahwa tidak ada informasi yang dapat ditarik atau dikenali dari cipherteks terkait dengan plainteks yang digunakan.

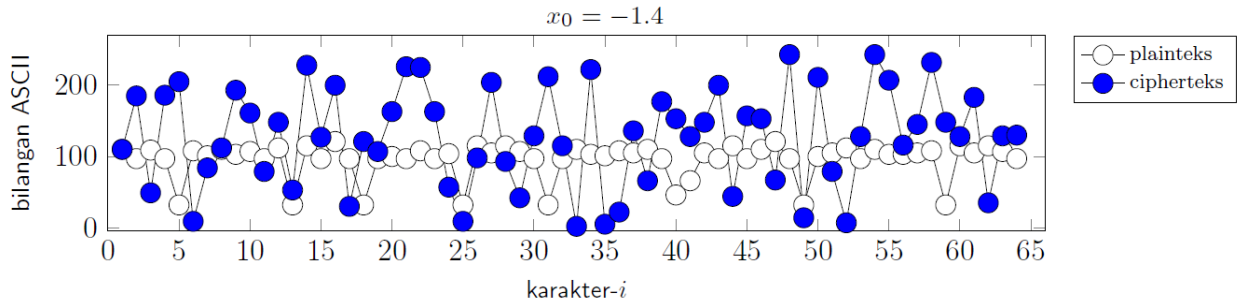
Tabel 9. Hasil Uji Korelasi

No	Kunci	Plainteks yang Diuji			Rata-Rata	Hasil Korelasi
		Plainteks 1	Plainteks 2	Plainteks 3		
1	bil-2	-0.17683578	0.18524564	-0.11891631	-0.03683548	Sangat Rendah
2	bil-3	-0.19371701	-0.17550337	-0.21661625	-0.19527887	Sangat Rendah
3	bil-4	0.02169153	-0.05553356	0.00541918	-0.00947428	Sangat Rendah
4	bil-5	0.07248945	-0.09767930	-0.10525845	-0.04348277	Sangat Rendah

Hasil pengujian tersebut kemudian disajikan dalam Tabel 9, yang memperlihatkan seberapa baik kunci enkripsi dapat mengamankan data dan menjaga kerahasiaan plainteks. Dengan demikian, menghilangkan hubungan antara keduanya adalah salah satu indikator keberhasilan proses enkripsi.

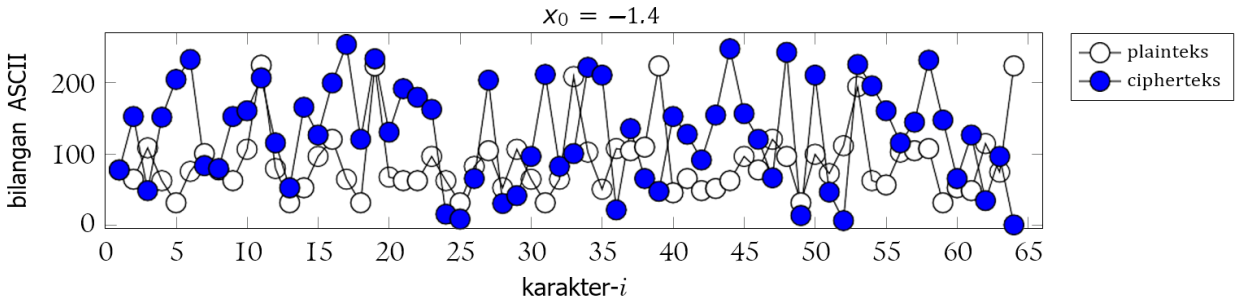
Pengujian enkripsi tidak melibatkan bil-1, karena hasilnya tidaklah acak dan tidak memenuhi standar keacakan dari CSPRNG *chaos* sehingga dianggap tidak berhasil dalam pengujian keacakan. Secara rata-rata menunjukkan bahwa memiliki tingkat keterkaitan yang sangat rendah. Hal ini menunjukkan bahwa semua bilangan yang digunakan sebagai kunci berhasil membuat plainteks tidak memiliki hubungan secara statistik dengan cipherteks. Pengujian ini menunjukkan bahwa kekuatan kunci mampu menyulitkan upaya kriptanalisis dalam mencari hubungan antara plainteks dan cipherteks. Dengan demikian, persamaan $\frac{x-(x^2-4)}{2x}$ dan inisialisasi $x_0 = -1.4$, berhasil digunakan sebagai pembangkit bilangan acak.

Pengujian selanjutnya visualisasi plainteks dan cipherteks menggunakan *butterfly effect*. Berdasarkan hasil uji korelasi pada setiap bil yang ditunjukan pada Tabel 9, bil-4 memiliki nilai rata-rata yang paling kecil sehingga bil-4 menjadi kunci yang terpilih. Untuk membuat buttery eect menggunakan bil-4 sebagai kunci dan menggunakan tiga plainteks yang telah digunakan sebelumnya. Pengujian ini dapat menunjukkan apakah kunci yang digunakan dapat menghasilkan cipherteks yang tidak memiliki pola yang dapat diprediksi dari plainteks. Jika hasil visualisasi menunjukkan perbedaan bentuk antara plainteks dan cipherteks, maka dapat dikatakah bahwa alagoritma enkripsi (kunci) telah berhasil menciptakan efek difusi dan konfusi pada proses enkripsi.



Gambar 7. Perbandingan Plainteks-1 dan Cipherteks

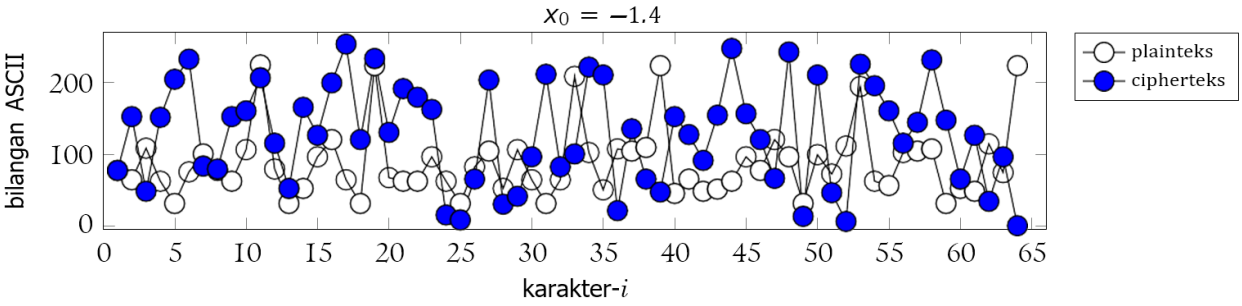
Hasil pengujian untuk plainteks 1, nama lengkap saya adalah siska angelina.Biasanya dipanggil siska ditampilkan dalam Gambar 7. Diagram garis untuk plainteks dan cipherteks memiliki perbedaan yang signikan, keduanya tidak menunjukkan pola yang serupa. Plainteks hanya ada di antara data 32 hingga 121, sedangkan cipherteks memiliki rentang yang lebih luas antara 2 hingga 243. Oleh sebab itu, penggunaan kunci ini efektif dalam menciptakan bentuk geometri yang bervariasi. Dengan demikian, akan lebih sulit bagi kripotanalisis untuk menebak dengan menggunakan pola dan grafik.



Gambar 8. Perbandingan Plainteks-2 dan Cipherteks

Hasil pengujian untuk plainteks 1, nama lengkap saya adalah siska angelina.Biasanya dipanggil siska ditampilkan dalam Gambar 7. Diagram garis untuk plainteks dan cipherteks memiliki perbedaan yang signikan, keduanya tidak menunjukkan pola yang serupa. Plainteks hanya ada di antara data 32 hingga 121, sedangkan cipherteks memiliki rentang yang lebih luas antara 2 hingga 243. Oleh sebab itu, penggunaan kunci ini efektif dalam menciptakan bentuk geometri yang bervariasi. Dengan demikian, akan lebih sulit bagi kriptanalisis untuk menebak dengan menggunakan pola dan grak.

NAm—A LeNGkaP 5ayA aD—a«LaĤ Si5kAA Ng3lina.B14«SaNya dIp A«N9gil 5iSka sebagai penggunaaan plainteks 2 ditunjukkan pada Gambar 8. Ruang plainteks terlihat lebih besar daripada plainteks 1, dengan rentang 32 hingga 225, dan memiliki sebuah diagram garis yang kurang stabil. Cipherteks yang didapat memiliki rentang yang luas, yaitu antara 1 hingga 254, dan juga menunjukkan ketidakstabilan pada data diagram garis.



Gambar 9. Perbandingan Plainteks-3 dan Cipherteks

Pada Plainteks 3 dilakukan pengujian ekstrim dengan menggunakan karakter yang serupa. Plainteks AAA-AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAB. Plainteks tersebut membentuk satu garis lurus yang bisa mengakibatkan perubahan besar pada cipherteks. Hasil dari rentang plainteks adalah antara 65 hingga 66, sementara rentang cipherteks berada dalam rentang 0 hingga 254.

4. Pembahasan

Pada pengujian enkripsi, setiap bilangan yang digunakan sebagai kunci memiliki tingkat keterkaitan yang sangat rendah karena nilai rata-rata yang dihasilkan mendekati 0 (nol). Selain itu, berdasakan butterfly effect pada cipherteks, terlihat bahwa kunci mampu mengurangi atau bahkan menghilangkan korelasi antara plainteks dan cipherteks karena memiliki peran yang sangat efektif dalam melakukan proses enkripsi. Hasil ini menunjukkan bahwa kunci yang diperoleh melalui pembangkitan bilangan acak menggunakan fungsi $\frac{x-(x^2-4)}{2x}$ dapat mengamankan data dan menjaga kerahasiaan plainteks, mampu menghasilkan cipherteks yang tetap stabil dalam rentang yang luas, yaitu antara 0 hingga 254 saat dibandingkan dengan plainteks.

Dampaknya adalah lebih sulit untuk menebak plainteks maupun cipherteks, karena kunci dapat menghasilkan berbagai kemungkinan cipher. Meskipun demikian, kriptanalis mungkin berhasil dalam mendekripsi pesan, namun algoritma enkripsi hanya melibatkan operasi penjumlahan, menggunakan metode Newton Raphson sebagai pembangkit kunci kriptogra pasti akan membutuhkan waktu yang lebih lama.

Penelitian ini membuat suatu pembangkit bilangan acak yang menghasilkan bilangan unik dan sulit diprediksi sehingga dapat diimplementasikan dalam berbagai bidang seperti pada pembuatan permainan untuk men- ciptakan variasi dalam level, karakter, item, peristiwa dalam permainan, meningkatkan keunikan dan tan- tangan setiap kali permainan dimainkan misalnya dalam permainan pada puzzle angka, tebak kata, ludo, sudoku dan ular tangga. Pada sistem keamanan sebagai pembangkit kunci, token otentikasi, generator kata sandi, mengamankan data citra digital, mengamankan le video dan angka acak lainnya untuk melindungi data dan sistem.

5. Simpulan

Dari pengujian yang telah dilakukan dapat disimpulkan bahwa penelitian ini berhasil menemukan fungsi polinomial yang dapat dijadikan sebagi pembangkit bilangan acak berbasis CSPNRG chaos. Melalui kombinasi dari metode iterasi yang belum ditemukan dari penelitian sebelumnya yaitu dengan metode *Fixed Point Iteration* (FPI) dan metode Newton Raphson menghasilkan hasil yang baik.

Inisialisasi $x_0 = -1.4$ adalah nilai yang optimal untuk menghasilkan hasil iterasi yang beragam. Analisis visual dengan menggunakan *Scatter Plot* menghasilkan grafik yang tidak membentuk sebuah pola untuk setiap kumpulan bil. Pengujian keacakan dengan *run test*, *mono bit*, dan *block bit* juga menunjukkan bahwa setiap kumpulan bit yang dihasilkan bersifat acak, dengan nilai *p-value* yang kurang dari $\alpha = 1\%$. Hasil uji enkripsi menunjukkan bahwa korelasi antara *ciphertext* dan kekuatan relatifnya merupakan "sangat rendah". Semua kunci acak yang digunakan dalam proses enkripsi berhasil menyembunyikan hubungan antara plaintext dan ciphertext dari sudut pandang statistik.

Secara keseluruhan, hasil dari metode Newton Raphson untuk fungsi $\frac{x-(x^2-4)}{2x}$ dengan nilai inisialisasi $x_0 = -1.4$ dapat menyebabkan kemunculan bilangan secara acak berdasarkan *chaos* CSPRNG. Fungsi $\frac{x-(x^2-4)}{2x}$ dapat dipergunakan untuk menciptakan sebuah kunci dalam cipher *block cipher*.

Pustaka

[1] F. I. P. S. Publication, "Advanced encryption standard (des)," U.S. Department of Commerce: National Institute of Standards and Technology (NIST), Tech. Rep., 2001.

[2] A. Biryukov and C. D. Cannière, "Data encryption standard (des)," in *Encyclopedia of Cryptography and Security*. Boston, MA: Springer US, 2011, pp. 295–301.

[3] H. Solis-Sanchez and E. G. Barrantes, "Using the logistic coupled map for public key cryptography under a distributed dynamics encryption scheme," *Information*, vol. 9, no. 7, 2018.

[4] D. Arius, *Pengantar Ilmu Kriptografi: Teori Analisis Dan Implementasi*. Yogyakarta: Andi, 2008.

[5] D. A. D. Paraditasari and Wowor, "Desain pembangkit kunci block cipher berbasis csprng chaos menggunakan fungsi trigonometri," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)*, vol. 6, pp. 400–405, Nov. 2021.

[6] O. N. E. Choesni, "Implementasi 30x3−19x−6 sebagai fungsi pembangkit bilangan acak menggunakan metode newton raphson," 2022, accessed: 2024-08-07. [Online]. Available: https://repository.uksw.edu/handle/123456789/26431

[7] E. T. Ndruru and Zebua, "Pembangkitan kunci beaufort cipher dengan teknik blum-blum shub pada pengamanan citra digital," *Bulletin of Information Technology (BIT)*, vol. 3, no. 2, pp. 149–154, Jun. 2022.

[8] T. S. A. D. Nahading and Wowor, "Desain pembangkit kunci lfsr dengan skema a5/1 menggunakan empat blok bit fungsi xor," *Jurnal Penerapan Sistem Informasi (Komputer dan Manajemen)*, vol. 4, no. 2, pp. 409–419, Apr. 2023.

[9] H. Simanjuntak, "Implementasi algoritma lucifer dengan pembangkit kunci multiply with carry generator untuk mengamankan data citra digital," *Jurnal Media Informatika (JUMIN)*, vol. 5, no. 1, pp. 22–31, Dec. 2023.

[10] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed. Brooks and Cole, 2011.

[11] I. Riadi *et al.*, "Analisis perbandingan detection trac anomaly dengan metode naive bayes dan support vector machine (svm)," *ILKOM Jurnal Ilmiah*, vol. 11, no. 1, pp. 17–24, 2019.

[12] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 6th ed. New York: McGraw-Hill, 2010.

[13] N. Onasie and S. Sulaiman, "Perancangan sendok makan parkinson dengan metode pid berbasis arduino," *Techné: Jurnal Ilmiah Elektroteknika*, vol. 22, no. 1, pp. 33–48, 2023.

[14] R. J. Belora, "Implementasi 3x3 + 5x2 + 7x−20 sebagai fungsi generator menggunakan metode newton raphson," 2021, accessed: 2024-08-07. [Online]. Available: https://repository.uksw.edu/bitstream/123456789/27295/2/T1_672016064_Isi.pdf

[15] R. S. Witte and J. S. Witte, *Statistics*. John Wiley & Sons, 2017.

[16] D. C. Montgomery and G. C. Runger, *Applied Statistics and Probability for Engineers*, 6th ed. New Jersey: John Wiley & Sons, 2014.

[17] G. I. Taopan *et al.*, "Pengamanan portable document format (pdf) menggunakan algoritma kriptografi kurva eliptik," *J-Icon: Jurnal Komputer dan Informatika*, vol. 10, no. 1, pp. 47–54, 2022.

[18] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 3rd ed. Pearson Education, Inc., 2003.

[19] A. Rukhin *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," National Institute of Standards and Technology, Gaithersburg, MD, Special Publication (NIST SP), 2001.

[20] J. F. Hair *et al.*, *Multivariate Data Analysis*, 8th ed. Cengage Learning, 2019.

[21] A. A. D. Ramadhani and Wowor, "Implementasi variasi fungsi xor dalam pembangkitan kunci lfsr pada skema a5/1 dengan tiga blok bit," *JIKO (Jurnal Informatika dan Komputer)*, vol. 8, no. 1, pp. 161–173, 2024.

[22] I. Riadi *et al.*, "Pengamanan citra digital berbasis kriptografi menggunakan algoritma vigenere cipher," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 7, no. 1, pp. 33–45, 2022.