Percobaan Perancangan Fungsi Pembangkit Bilangan Acak Semu serta Analisisnya

Athia Saelan (13508029)[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*if18029@students.if.itb.ac.id*

***Abstrak*— Komputer, pada dasarnya, tidak dapat membangkitkan bilangan yang benar-benar acak. Untuk memenuhi kebutuhan bilangan acak dengan komputer, dibuatlah pembangkit bilangan acak semu atau Pseudo Random Number Generator (PRNG). Pembangkit tersebut pada dasarnya tidak menghasilkan bilangan yang acak, namun merupakan hasil dari fungsi tertentu. Pada makalah ini, akan dibahas sebuah percobaan perancangan fungsi pembangkit bilangan acak semu yang baru. Fungsi tersebut akan diujicobakan dan dibandingkan dengan fungsi yang sudah ada. Perbandingan yang dilakukan antara lain kemangkusan dan keacakan.**

***Kata Kunci*— bilangan acak, bilangan acak semu, pseudo random number generator**

I. PENDAHULUAN

Saat ini, komputer pada dasarnya tidak dapat membangkitkan bilangan acak. Namun, bilangan acak ini sering kali dibutuhkan dalam penggunaan komputer, misalnya untuk simulasi atau kriptografi. Untuk memenuhi kebutuhan tersebut, dibuatlah fungsi pembangkit bilangan acak semu atau *Pseudo Random Number Generator* (*PRNG*).

Pembangkit tersebut disebut semu karena bilangan yang dihasilkan sebenarnya tidak acak, namun merupakan hasil dari fungsi tertentu. Kemunculan bilangan acak tersebut dapat diulang urutannya jika menggunakan umpan yang sama. Bahkan beberapa algoritma dapat diprediksi kemunculannya.

Namun, untuk keperluan keamanan seperti kriptografi, dibutuhkan pembangkit bilangan acak yang sulit diprediksi. Oleh karena itu, dibuatlah pembangkit bilangan acak yang lebih aman yang disebut Cryptographically Secure Pseudo Random Number Generator (CSPRNG). Pembangkit jenis ini lebih aman karena kemunculan bilangan berikutnya lebih sulit diprediksi. Meskipun begitu, urutan kemunculan bilangan masih bisa diulang dengan menggunakan umpan yang sama.

II. DASAR TEORI

Bilangan acak adalah bilangan yang tidak dapat diprediksi kemunculannya. Bilangan acak ini banyak digunakan dalam kriptografi, misalnya untuk pembangkitan parameter kunci pada algoritma kunci publik, pembangkitan initial vector pada algoritma block chiper, dan sebagainya.

Tidak ada komputasi yang benar-benar menghasilkan deret bilangan acak. Bilangan acak yang dibangkitkan dengan rumus matematika adalah bilangan acak semu. Pembangkit ini disebut *Pseudo Random Number Generator* (*PRNG*).

Salah satu pembangkit bilangan acak yang digunakan adalah LCG (*linear congruential generator*). Fungsi dari pembangkit ini adalah sebagai berikut.

$$X_n = (aX_{n-1} + b) \bmod m$$

dengan kunci pembangkit adalah X_0 yang disebut *seed*.

Keunggulan LCG adalah kemangkusannya. Namun algoritma ini tidak dapat digunakan untuk kriptografi karena bilangan acaknya dapat diprediksi. Meskipun begitu, LCG tetap berguna untuk aplikasi non-kriptografi, karena LCG cukup mangkus dan keacakannya bagus.

III. METODE

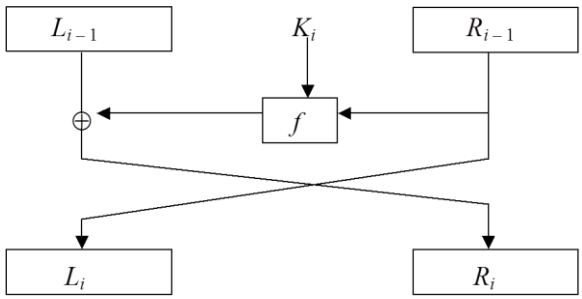
Metode yang digunakan pada percobaan ini adalah melakukan eksperimen terhadap beberapa fungsi yang diduga dapat membangkitkan bilangan acak. Fungsi tersebut dibuat kode programnya dalam bahasa C, kemudian diujikan dengan beberapa nilai parameter. Setelah itu dilihat, apakah hasilnya membentuk suatu barisan bilangan yang cukup acak atau hanya mengeluarkan beberapa bilangan yang terurut.

IV. PERCOBAAN

Eksperimen dilakukan terhadap beberapa fungsi. Namun beberapa fungsi yang dicobakan ternyata sama sekali tidak menghasilkan barisan bilangan yang acak. Fungsi yang akan dibahas pada makalah ini hanyalah yang hasilnya dirasa cukup acak.

A. Fungsi Berbasis Jaringan Feistel

Ide dasar dari fungsi ini adalah jaringan feistel yang sepertinya cukup mengacak dengan membolak-balikkan dua kumpulan bit. Fungsi ini dilakukan dengan membagi bilangan menjadi dua bagian, yaitu bagian kiri (L) dan bagian kanan (R). Pada percobaan yang dilakukan, bilangan yang digunakan terdiri dari 16 bit, kemudian dibagi dua menjadi masing-masing 8 bit.



Gambar 1 jaringan feistel

Fungsi f yang digunakan pada percobaan ini adalah sebagai berikut.

$$f(L,R) = (L \times R) \bmod n$$

Dalam percobaan ini, n yang digunakan adalah suatu bilangan prima. Secara umum, skema pembangkitan bilangan acak yang dilakukan adalah sebagai berikut.

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus \left((L_i \times R_i) \bmod n\right)$$

Algoritma yang dibuat untuk fungsi yang berbasis feistel adalah sebagai berikut.

```
input (x)  {seed}
L ← 8 bit paling kiri dari x
R ← 8 bit paling kanan dari x

while (not end) do
  temp ← R
  R ← L xor ((x*y) mod n)
  L ← temp
  X ← gabungan L dan R
```

B. Fungsi Perkalian

Fungsi selanjutnya yang dicobakan adalah fungsi yang berdasarkan pada perkalian. Perkalian yang dilakukan adalah antara dua angka sebelumnya. Fungsi yang dibuat adalah sebagai berikut.

$$x_n = (x_{n-1} \times x_{n-2}) \bmod n$$

Nilai n yang digunakan di sini juga merupakan bilangan prima. Karena perkalian yang dilakukan adalah antara dua angka sebelumnya, pada permulaan fungsi ini dibutuhkan dua buah seed.

Algoritma yang dibuat adalah sebagai berikut.

```
input (x1)  {seed1}
input (x2)  {seed2}

while (not end) do
  hasil ← (x2 * x1) mod n
  x2 ← x1
  x1 ← hasil
```

V. HASIL DAN PEMBAHASAN

Percobaan dilakukan dengan mengubah-ubah nilai n dan seed. Beberapa hasil yang didapatkan dari percobaan adalah sebagai berikut

A. *Fungsi Berbasis Jaringan Feistel*

Tabel di bawah ini menunjukkan hasil percobaan dengan menggunakan beberapa seed dan beberapa nilai n. Di sini hanya akan ditampilkan 10 nilai acak yang dihasilkan. Untuk lebih lengkapnya, 100 nilai acak yang dihasilkan dapat dilihat di lampiran.

n = 29

No.	seed		
	100	12345	23478
1	25600	14650	46680
2	100	14905	22718
3	25600	14650	48712
4	100	14905	18603
5	25600	14650	43864
6	100	14905	22705
7	25600	14650	45403
8	100	14905	23485
9	25600	14650	48473
10	100	14905	22972

n = 73

No.	seed		
	100	12345	23478
1	25600	14611	46619
2	100	4868	7073
3	25600	1040	41267
4	100	4164	13186
5	25600	17490	33295
6	100	21080	4022

7	25600	22636	46610
8	100	27734	4854
9	25600	22141	63010
10	100	32069	8924

n = 211

No.	seed		
	100	12345	23478
1	25600	14844	46643
2	100	64553	13159
3	25600	10544	26510
4	100	12396	36388
5	25600	27720	9406
6	100	18648	48764
7	25600	55517	31797
8	100	56810	13667
9	25600	60110	25474
10	100	52874	33457

Pada banyak kasus, fungsi ini tidak menghasilkan bilangan yang acak. Fungsi ini memiliki kelemahan, yaitu tidak boleh ada salah satu bagian yang bernilai nol. Jika ada yang bernilai nol, fungsi ini akan mengembalikan dua buah angka secara bergantian.

Nilai nol bisa didapatkan sejak dari seednya atau didapatkan dari adanya salah satu sisi yang habis dibagi n. Nilai ini sering kali didapatkan dari nilai seed yang kecil, yaitu kurang dari 256, yang hanya akan mengisi bagian kanan dari jaringan feistel yang digunakan.

Contoh nilai nol yang didapatkan dari seed adalah ketika seed yang digunakan bernilai 100 (kurang dari 256). Sedangkan contoh nilai nol yang didapatkan dari sisi yang habis dibagi n adalah saat n = 29 dan seed = 12345. Ketika mencapai nilai 14650, saat dibagi dua, maka nilai L = 57 dan R = 58. Karena 58 merupakan kelipatan 29, maka nilai modulus yang dihasilkan adalah nol.

B. Fungsi Berdasarkan Perkalian

Tabel di bawah ini menunjukkan hasil percobaan dengan menggunakan beberapa seed dan beberapa nilai n. Sama seperti sebelumnya, di sini hanya akan ditampilkan 10 nilai acak yang dihasilkan dan 100 nilai acak yang dihasilkan dapat dilihat di lampiran.

n = 29

No.	seed			
	100	12345	12345	23478
	12345	100	23478	100
1	28	28	21	18
2	16	9	14	16
3	13	20	4	27
4	5	6	27	26
5	7	4	21	6
6	6	24	16	11
7	13	9	17	8
8	20	13	11	1
9	28	1	13	8
10	9	13	27	8

n = 73

No.	seed			
	100	12345	12345	23478
	12345	100	23478	100
1	70	70	68	47
2	65	49	33	71
3	24	72	54	52
4	27	24	30	42
5	64	49	14	67
6	49	8	55	40
7	70	27	40	52
8	72	70	10	36
9	3	65	35	47
10	70	24	58	13

n = 211

No.	seed			
	100	12345	12345	23478
	12345	100	23478	100
1	150	150	191	3
2	19	14	181	171
3	107	201	178	91
4	134	71	146	158
5	201	134	35	30
6	137	19	46	98
7	107	14	133	197
8	100	55	210	105
9	150	137	78	7
10	19	150	133	102

Pada fungsi ini, nilai yang dihasilkan akan selalu kurang dari n. Fungsi ini juga memiliki kelemahan yang mirip dengan feistel yang dibuat, yaitu tidak seed yang diberikan tidak boleh bernilai nol. Namun fungsi ini tidak mungkin menghasilkan nilai nol selama seed yang diberikan tidak nol dan tidak sama dengan n, serta n yang digunakan merupakan bilangan prima.

VI. ANALISIS

Berdasarkan hasil dari percobaan yang telah dibahas, akan dilakukan analisis keacakan hasil fungsi dan kemangkusan algoritma yang digunakan. Selain itu, fungsi yang dibuat juga akan dibandingkan dengan fungsi yang sudah ada, yaitu LCG (Linear Congruential Generator). Algoritma dari fungsi LCG kira-kira sebagai berikut.

```
input (x)  {seed}
while (not end) do
  x ← ((a*x) + b) mod n
```

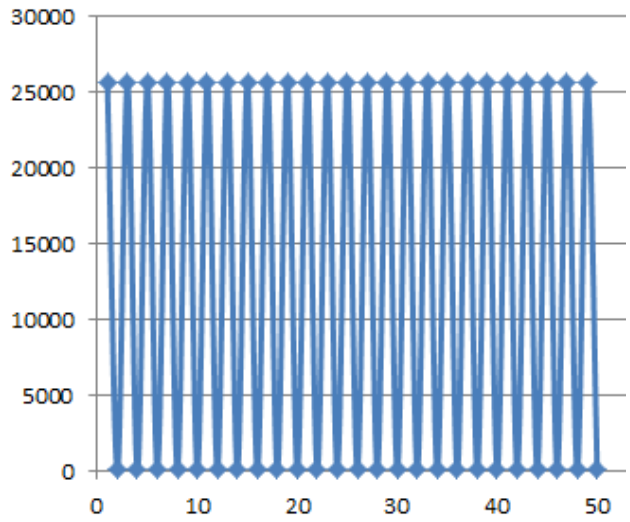
A. Keacakan

Analisis keacakan akan dilakukan dengan melihat barisan bilangan yang merupakan hasil dari fungsi yang dibuat. Barisan bilangan akan dibuat grafiknya dan dilihat keacakannya.

Pertama-tama akan dibahas mengenai keacakan fungsi yang berbasis jaringan feistel. Seperti yang sudah dibahas pada bagian sebelumnya, fungsi ini memiliki kecenderungan untuk menghasilkan dua bilangan secara bergantian pada kondisi tertentu. Meskipun begitu, ada juga beberapa kasus yang menyebabkannya bernilai cukup acak.

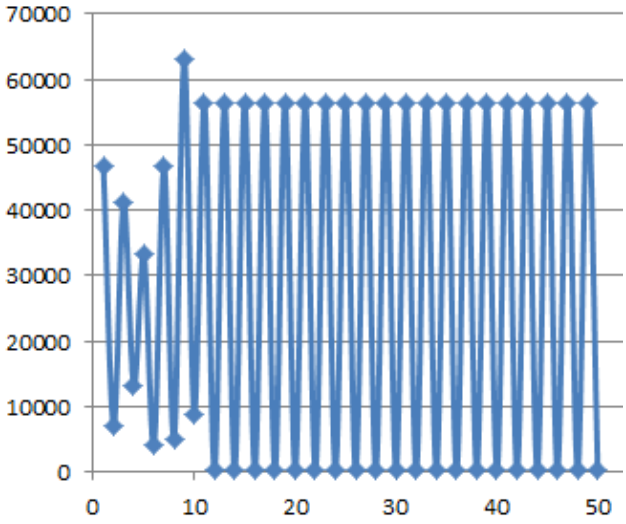
Berikut adalah beberapa grafik dari hasil pengacakan fungsi feistel yang dibuat. Grafik yang lebih lengkap dapat dilihat pada lampiran.

Salah satu hasil yang didapatkan, pada fungsi berbasis feistel, untuk seed=100, berapapun nilai n yang dicobakan menghasilkan hasil yang sama sebagai berikut.



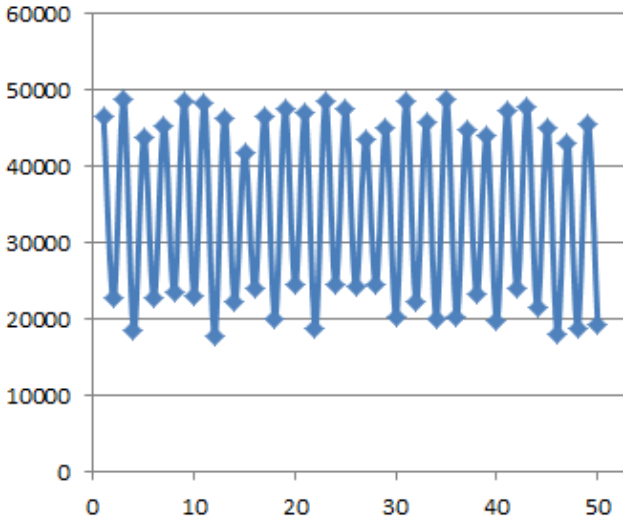
Feistel, seed=100

Masih banyak juga kasus yang menyebabkan deret yang dihasilkan sangat teratur seperti pada grafik di atas, antara lain saat n=29 dan seed=12345. Selain itu, ada juga yang awalnya agak acak namun akhirnya teratur seperti grafik berikut.



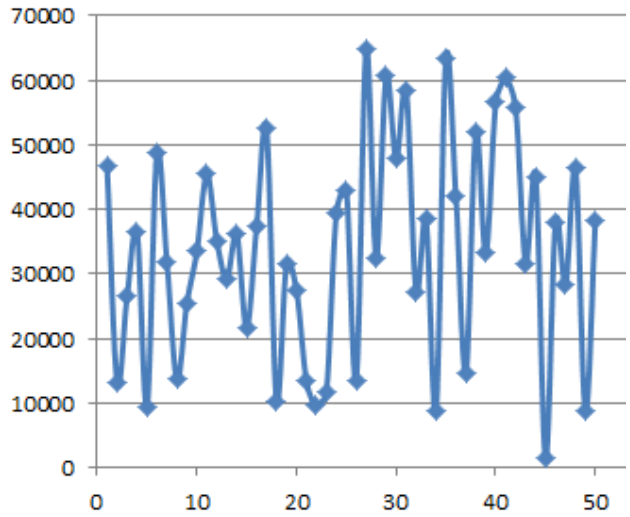
Feistel, n=73, seed=23478

Ada juga kasus yang grafiknya naik-turun secara teratur, namun angkanya tidak terlalu teratur seperti grafik berikut.



Feistel, n=29, seed=23478

Meskipun begitu, untuk kasus lain yang dicobakan, hasilnya cukup acak, misalnya saat n=73 dan seed=12345, saat n=211 dan seed=12345, atau saat n=211 dan seed=23478. Salah satu grafiknya adalah sebagai berikut.

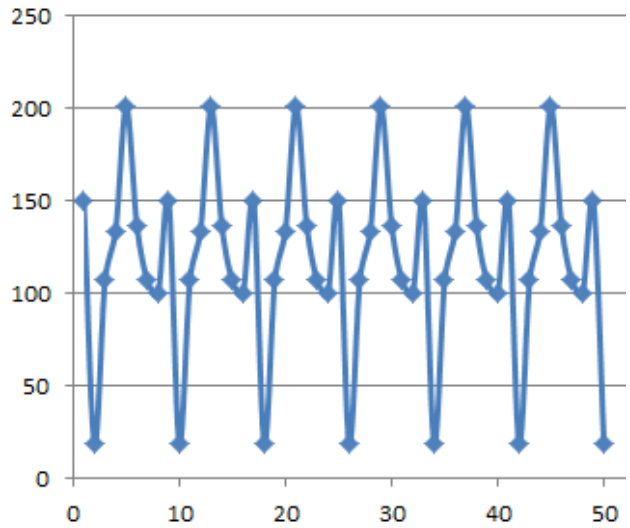


Feistel, n=211, seed=23478

Dari hasil tersebut, hanya 3 dari 9 kasus uji yang menghasilkan barisan bilangan yang cukup acak. 2 dari 3 yang cukup acak tersebut terjadi ketika n=211, yang merupakan n terbesar yang diujicobakan. Dengan demikian, dapat diduga bahwa n yang lebih besar akan lebih baik. Meskipun begitu, butuh penelitian lebih lanjut untuk membuktikan dugaan tersebut.

Selanjutnya, akan dibahas fungsi pembangkit bilangan acak yang berdasarkan perkalian.

Pada fungsi berdasarkan perkalian ini, nilai yang dihasilkan tidak akan lebih dari n. Selain itu, fungsi ini juga akan menghasilkan periode, yaitu kemunculannya akan berulang setelah waktu tertentu. Salah satu grafik yang menunjukkan periode dari fungsi ini yang cukup singkat adalah grafik di bawah ini.

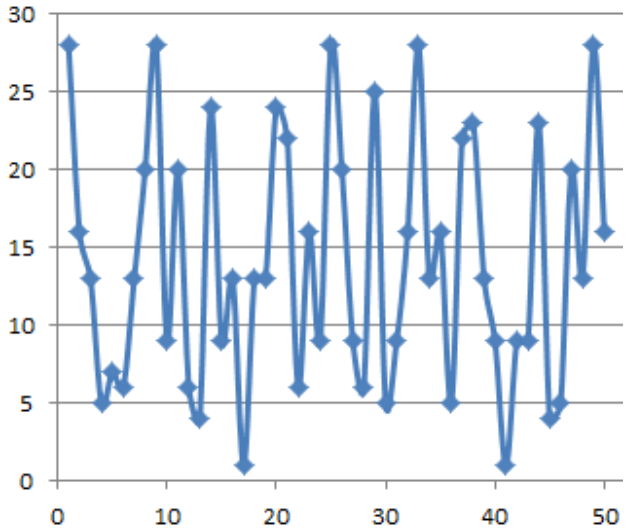


Perkalian, n=211, seed1=100, seed2=12345

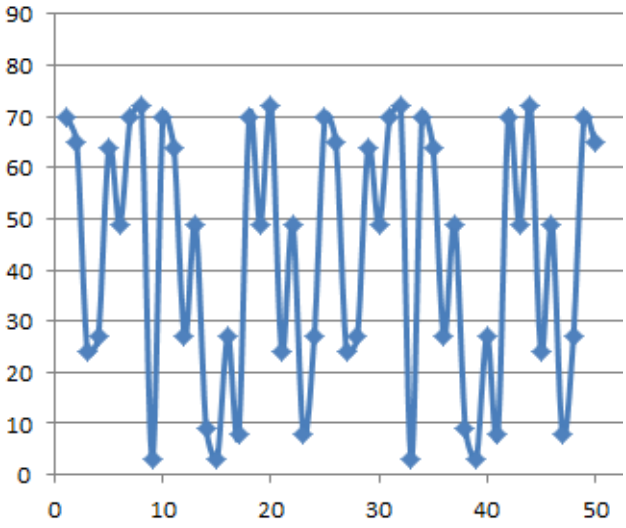
Pada kasus tersebut, periode yang terjadi adalah 8, yaitu nilai keluaran dari fungsi tersebut akan berulang setiap 8 kali pengulangan. Namun yang menyebabkan nilai keluaran berulang secepat itu hanya satu kasus dari 12 kasus yang diujikan. Secara umum, periode yang dihasilkan ketika n=29 adalah 48, jadi nilai keluaran pertama akan berulang pada pengulangan ke-49.

Sedangkan untuk n=73, pengulangan terjadi pada keluaran ke 25. Hal ini lebih buruk daripada saat n=29. Sedangkan untuk n=211, selain saat seed=100, nilai tidak berulang setelah 100 kali pengulangan. Oleh karena itu, belum bisa disimpulkan bagaimana menentukan nilai n agar hasilnya lebih baik. D iperlukan penelitian yang lebih lanjut untuk mengetahui parameter-parameter yang terbaik untuk menghasilkan bilangan acak.

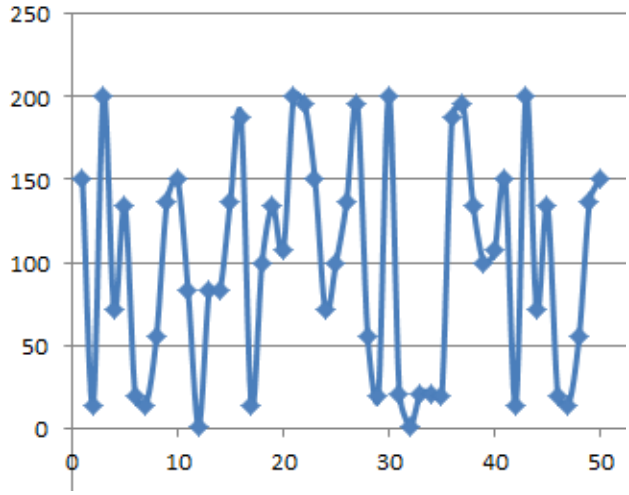
Di bawah ini ditampilkan beberapa grafik lain yang dihasilkan pada pembangkit bilangan acak dengan fungsi berdasarkan perkalian. Untuk grafik yang lebih lengkap dapat dilihat pada lampiran.



Perkalian n=29, seed1=100, seed2=12345

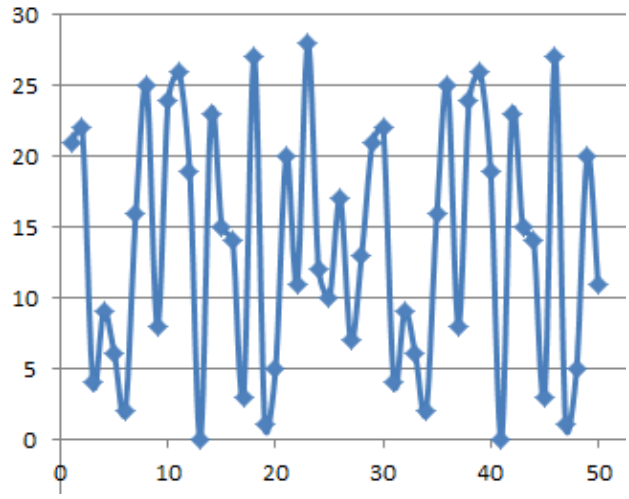


Perkalian, n=73, seed1=100, seed=12345

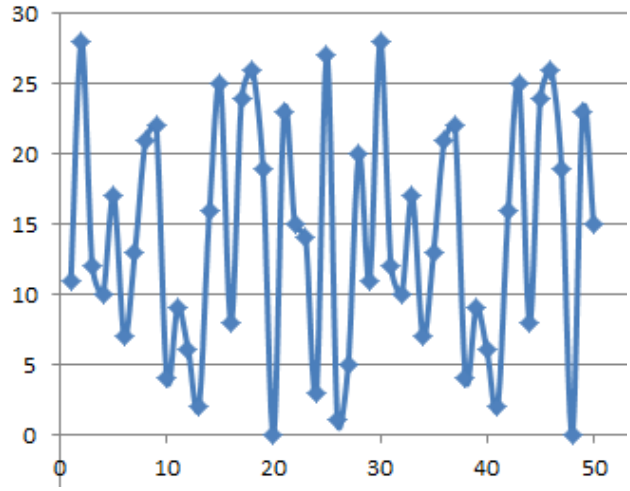


Perkalian, n=211, seed1=12345, seed2=100

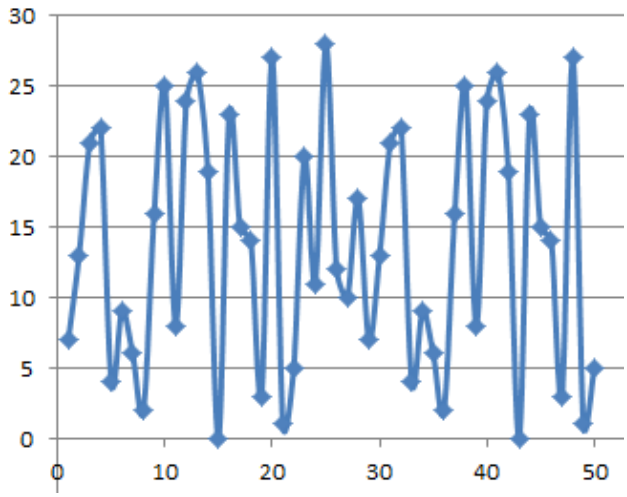
Sedangkan untuk LCG, dengan a=11, b=23, dan n=29, hasil grafiknya adalah sebagai berikut.



LCG, seed=100



LCG, seed=12345



LCG, seed=23478

Sama seperti pada fungsi perkalian, LCG juga memiliki periode. Periode pada LCG adalah kurang dari n, sedangkan pada pembangkit berdasarkan perkalian, periodenga bisa lebih dari n. Hal tersebut terjadi karena pada fungsi perkalian, parameter untuk membangkitkan suatu nilai keluaran adalah nilai keluaran sebelumnya dan dua nilai sebelumnya, sedangkan pada LCG hanya satu nilai sebelumnya. Hal tersebut menyebabkan nilai yang dihasilkan pada suatu saat tidak sepenuhnya mempengaruhi nilai yang berikutnya.

Namun, pada fungsi yang berdasarkan perkalian, akan ada beberapa kasus yang menyebabkan periodenya sangat kecil. Begitu juga dengan LCG. Namun, karena LCG sudah lebih banyak digunakan, LCG sudah memiliki aturan-aturan yang menyebabkan periodenya bisa maksimum.

B. Kemangkusan

Kemangkusan fungsi yang dibuat akan dilihat dari banyaknya jumlah operasi yang dilakukan. Semakin sedikit jumlah operasi, maka fungsi tersebut semakin mangkus.

Pertama, fungsi dengan jaringan feistel. Fungsi tersebut melakukan empat buah assignment pada setiap iterasinya. Operasi yang dilakukan antara lain perkalian, mod, dan penggabungan.

Selanjutnya fungsi dengan perkalian, melakukan tiga assignment pada setiap iterasinya. Operasi yang dilakukan adalah perkalian dan mod.

Sedangkan LCG, hanya melakukan satu buah assignment pada setiap iterasinya, dengan operasi berupa perkalian, penjumlahan, dan mod.

Dari perbandingan tersebut, dapat dilihat bahwa algoritma yang sudah ada sebelumnya, yaitu LCG, lebih mangkus daripada dua algoritma yang dicoba pada makalah ini.

VII. SIMPULAN

Dari eksperimen yang telah dibahas pada makalah ini, dapat diambil kesimpulan sebagai berikut.

1. Fungsi berbasis jaringan feistel yang dibuat belum baik sebagai pembangkit bilangan acak.
2. Fungsi perkalian yang dibuat sudah cukup dapat membangkitkan bilangan acak.
3. Fungsi pembangkit bilangan acak yang sudah ada sudah baik.
4. Pembangkit bilangan acak yang dibuat belum aman untuk kriptografi.

Saran untuk pengembangan selanjutnya:

1. Mencoba menggunakan jaringan feistel dengan fungsi yang lain untuk membangkitkan bilangan acak.
2. Menggunakan pengujian keacakan yang lebih baik.

REFERENSI

[1] Slide kuliah kriptografi, bagian pembangkit bilangan acak dan algoritma kriptografi modern.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Mei 2011

Athia Saelan
13508029

LAMPIRAN

Hasil pengacakan dengan fungsi berbasis jaringan feistel

n	29			73			211		
seed	100	12345	23478	100	12345	23478	100	12345	23478
1	25600	14650	46680	25600	14611	46619	25600	14844	46643
2	100	14905	22718	100	4868	7073	100	64553	13159
3	25600	14650	48712	25600	1040	41267	25600	10544	26510
4	100	14905	18603	100	4164	13186	100	12396	36388
5	25600	14650	43864	25600	17490	33295	25600	27720	9406
6	100	14905	22705	100	21080	4022	100	18648	48764
7	25600	14650	45403	25600	22636	46610	25600	55517	31797
8	100	14905	23485	100	27734	4854	100	56810	13667
9	25600	14650	48473	25600	22141	63010	25600	60110	25474
10	100	14905	22972	100	32069	8924	100	52874	33457
11	25600	14650	48197	25600	17782	56320	25600	35412	45449
12	100	14905	17845	100	30306	220	100	21580	35186
13	25600	14650	46422	25600	25192	56320	25600	19554	29325
14	100	14905	22179	100	26703	220	100	25203	36180
15	25600	14650	41821	25600	20288	56320	25600	29493	21649
16	100	14905	23990	100	16476	220	100	13768	37325
17	25600	14650	46670	25600	23664	56320	25600	51207	52520
18	100	14905	20153	100	28759	220	100	1870	10363
19	25600	14650	47455	25600	22355	56320	25600	20091	31595
20	100	14905	24504	100	21268	220	100	31533	27444
21	25600	14650	47177	25600	5221	56320	25600	11594	13349
22	100	14905	18877	100	25893	220	100	19080	9517
23	25600	14650	48479	25600	9579	56320	25600	35033	11673
24	100	14905	24505	100	27444	220	100	55615	39336
25	25600	14650	47454	25600	13435	56320	25600	16254	43060
26	100	14905	24234	100	31513	220	100	32444	13565
27	25600	14650	43615	25600	6514	56320	25600	48198	64894
28	100	14905	24496	100	29210	220	100	18162	32492
29	25600	14650	45135	25600	6750	56320	25600	62074	60602
30	100	14905	20413	100	24121	220	100	31281	47844
31	25600	14650	48470	25600	14659	56320	25600	12604	58474
32	100	14905	22195	100	17198	220	100	15604	27286
33	25600	14650	45902	25600	11859	56320	25600	62573	38433
34	100	14905	20158	100	21304	220	100	28158	8695
35	25600	14650	48719	25600	14434	56320	25600	65088	63396
36	100	14905	20399	100	25141	220	100	16631	42040
37	25600	14650	44890	25600	13673	56320	25600	63362	14539
38	100	14905	23212	100	26916	220	100	33489	52097
39	25600	14650	44109	25600	9296	56320	25600	53536	33244
40	100	14905	19896	100	20485	220	100	8258	56555

41	25600	14650	47197	25600	1395	56320	25600	16930	60377
42	100	14905	23994	100	29509	220	100	8900	55675
43	25600	14650	47699	25600	17728	56320	25600	50265	31664
44	100	14905	21424	100	16481	220	100	22858	45061
45	25600	14650	45126	25600	24899	56320	25600	19060	1428
46	100	14905	18088	100	17251	220	100	29914	37998
47	25600	14650	43081	25600	25468	56320	25600	56007	28341
48	100	14905	18866	100	31855	220	100	51109	46370
49	25600	14650	45643	25600	28500	56320	25600	42309	8854
50	100	14905	19384	100	21594	220	100	17775	38406
51	25600	14650	47186	25600	23165	56320	25600	28538	1710
52	100	14905	21168	100	32082	220	100	31305	44750
53	25600	14650	45121	25600	21091	56320	25600	18774	52759
54	100	14905	16830	100	25437	220	100	22230	6062
55	25600	14650	48728	25600	23914	56320	25600	54905	44763
56	100	14905	22702	100	27230	220	100	31054	56272
57	25600	14650	44632	25600	24142	56320	25600	20195	53344
58	100	14905	22702	100	20094	220	100	58255	24662
59	25600	14650	44632	25600	32352	56320	25600	36689	22139
60	100	14905	22702	100	24653	220	100	20786	31562
61	25600	14650	44632	25600	19827	56320	25600	12920	19046
62	100	14905	22702	100	29531	220	100	30830	26345
63	25600	14650	44632	25600	23401	56320	25600	28174	59872
64	100	14905	22702	100	26906	220	100	3665	57506
65	25600	14650	44632	25600	6772	56320	25600	20801	41519
66	100	14905	22702	100	29709	220	100	16792	12208
67	25600	14650	44632	25600	3396	56320	25600	39151	45060
68	100	14905	22702	100	17413	220	100	61372	1271
69	25600	14650	44632	25600	1396	56320	25600	48167	63380
70	100	14905	22702	100	29760	220	100	10018	38082
71	25600	14650	44632	25600	16455	56320	25600	8731	49796
72	100	14905	22702	100	18258	220	100	7016	33935
73	25600	14650	44632	25600	21104	56320	25600	26714	36837
74	100	14905	22702	100	28777	220	100	23076	58789
75	25600	14650	44632	25600	26999	56320	25600	9233	42485
76	100	14905	22702	100	30565	220	100	4506	62937
77	25600	14650	44632	25600	25944	56320	25600	39495	55609
78	100	14905	22702	100	22610	220	100	18231	14682
79	25600	14650	44632	25600	21094	56320	25600	14124	23163
80	100	14905	22702	100	26232	220	100	11348	31544
81	25600	14650	44632	25600	30807	56320	25600	21569	14579
82	100	14905	22702	100	22393	220	100	16877	62288
83	25600	14650	44632	25600	31064	56320	25600	60739	20719
84	100	14905	22702	100	22598	220	100	17371	61394
85	25600	14650	44632	25600	17988	56320	25600	56113	53848

86	100	14905	22702	100	17481	220	100	12654	22697
87	25600	14650	44632	25600	18756	56320	25600	28226	43326
88	100	14905	22702	100	17481	220	100	16952	15906
89	25600	14650	44632	25600	18756	56320	25600	14383	8943
90	100	14905	22702	100	17481	220	100	12124	61262
91	25600	14650	44632	25600	18756	56320	25600	23623	20133
92	100	14905	22702	100	17481	220	100	18326	42396
93	25600	14650	44632	25600	18756	56320	25600	38435	40052
94	100	14905	22702	100	17481	220	100	9004	29757
95	25600	14650	44632	25600	18756	56320	25600	11292	15621
96	100	14905	22702	100	17481	220	100	7325	1379
97	25600	14650	44632	25600	18756	56320	25600	40364	25420
98	100	14905	22702	100	17481	220	100	44114	19688
99	25600	14650	44632	25600	18756	56320	25600	21022	59451
100	100	14905	22702	100	17481	220	100	7897	15184

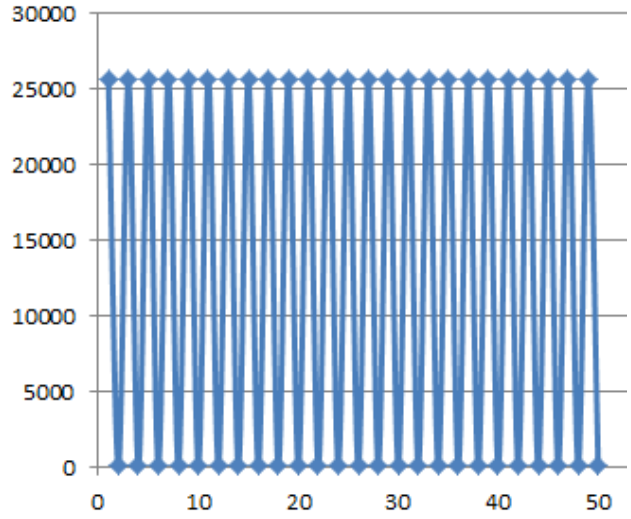
Hasil pengacakan dengan fungsi berdasarkan perkalian

n	29				73				211			
s1	100	12345	12345	23478	100	12345	12345	23478	100	12345	12345	23478
S2	12345	100	23478	100	12345	100	23478	100	12345	100	23478	100
1	28	28	21	18	70	70	68	47	150	150	191	3
2	16	9	14	16	65	49	33	71	19	14	181	171
3	13	20	4	27	24	72	54	52	107	201	178	91
4	5	6	27	26	27	24	30	42	134	71	146	158
5	7	4	21	6	64	49	14	67	201	134	35	30
6	6	24	16	11	49	8	55	40	137	19	46	98
7	13	9	17	8	70	27	40	52	107	14	133	197
8	20	13	11	1	72	70	10	36	100	55	210	105
9	28	1	13	8	3	65	35	47	150	137	78	7
10	9	13	27	8	70	24	58	13	19	150	133	102
11	20	13	3	6	64	27	59	27	107	83	35	81
12	6	24	23	19	27	64	64	59	134	1	13	33
13	4	22	11	27	49	49	53	60	201	83	33	141
14	24	6	21	20	9	70	34	36	137	83	7	11
15	9	16	28	18	3	72	50	43	107	137	20	74
16	13	9	8	12	27	3	21	15	100	188	140	181
17	1	28	21	13	8	70	28	61	150	14	57	101
18	13	20	23	11	70	64	4	39	19	100	173	135
19	13	9	19	27	49	27	39	43	107	134	155	131
20	24	6	2	7	72	49	10	71	134	107	18	172
21	22	25	9	15	24	9	25	60	201	201	47	166
22	6	5	18	18	49	3	31	26	137	196	2	67
23	16	9	17	9	8	27	45	27	107	150	94	150
24	9	16	16	17	27	8	8	45	100	71	188	133

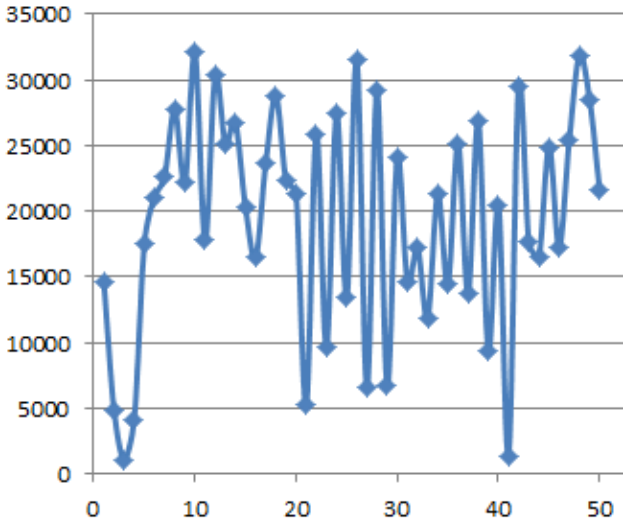
25	28	28	11	8	70	70	68	47	150	100	159	116
26	20	13	2	20	65	49	33	71	19	137	141	25
27	9	16	22	15	24	72	54	52	107	196	53	157
28	6	5	15	10	27	24	30	42	134	55	88	127
29	25	22	11	5	64	49	14	67	201	19	22	105
30	5	23	20	21	49	8	55	40	137	201	37	42
31	9	13	17	18	70	27	40	52	107	21	181	190
32	16	9	21	1	72	70	10	36	100	1	156	173
33	28	1	9	18	3	65	35	47	150	21	173	165
34	13	9	15	18	70	24	58	13	19	21	191	60
35	16	9	19	5	64	27	59	27	107	19	127	194
36	5	23	24	3	27	64	64	59	134	188	203	35
37	22	4	21	15	49	49	53	60	201	196	39	38
38	23	5	11	16	9	70	34	36	137	134	110	64
39	13	20	28	8	3	72	50	43	107	100	70	111
40	9	13	18	12	27	3	21	15	100	107	104	141
41	1	28	11	9	8	70	28	61	150	150	106	37
42	9	16	24	21	70	64	4	39	19	14	52	153
43	9	13	3	15	49	27	39	43	107	201	26	175
44	23	5	14	25	72	49	10	71	134	71	86	189
45	4	7	13	27	24	9	25	60	201	134	126	159
46	5	6	8	8	49	3	31	26	137	19	75	89
47	20	13	17	13	8	27	45	27	107	14	166	14
48	13	20	20	17	27	8	8	45	100	55	1	191
49	28	28	21	18	70	70	68	47	150	137	166	142
50	16	9	14	16	65	49	33	71	19	150	166	114
51	13	20	4	27	24	72	54	52	107	83	126	152
52	5	6	27	26	27	24	30	42	134	1	27	26
53	7	4	21	6	64	49	14	67	201	83	26	154
54	6	24	16	11	49	8	55	40	137	83	69	206
55	13	9	17	8	70	27	40	52	107	137	106	74
56	20	13	11	1	72	70	10	36	100	188	140	52
57	28	1	13	8	3	65	35	47	150	14	70	50
58	9	13	27	8	70	24	58	13	19	100	94	68
59	20	13	3	6	64	27	59	27	107	134	39	24
60	6	24	23	19	27	64	64	59	134	107	79	155
61	4	22	11	27	49	49	53	60	201	201	127	133
62	24	6	21	20	9	70	34	36	137	196	116	148
63	9	16	28	18	3	72	50	43	107	150	173	61
64	13	9	8	12	27	3	21	15	100	71	23	166
65	1	28	21	13	8	70	28	61	150	100	181	209
66	13	20	23	11	70	64	4	39	19	137	154	90
67	13	9	19	27	49	27	39	43	107	196	22	31
68	24	6	2	7	72	49	10	71	134	55	12	47
69	22	25	9	15	24	9	25	60	201	19	53	191

70	6	5	18	18	49	3	31	26	137	201	3	115
71	16	9	17	9	8	27	45	27	107	21	159	21
72	9	16	16	17	27	8	8	45	100	1	55	94
73	28	28	11	8	70	70	68	47	150	21	94	75
74	20	13	2	20	65	49	33	71	19	21	106	87
75	9	16	22	15	24	72	54	52	107	19	47	195
76	6	5	15	10	27	24	30	42	134	188	129	85
77	25	22	11	5	64	49	14	67	201	196	155	117
78	5	23	20	21	49	8	55	40	137	134	161	28
79	9	13	17	18	70	27	40	52	107	100	57	111
80	16	9	21	1	72	70	10	36	100	107	104	154
81	28	1	9	18	3	65	35	47	150	150	20	3
82	13	9	15	18	70	24	58	13	19	14	181	40
83	16	9	19	5	64	27	59	27	107	201	33	120
84	5	23	24	3	27	64	64	59	134	71	65	158
85	22	4	21	15	49	49	53	60	201	134	35	181
86	23	5	11	16	9	70	34	36	137	19	165	113
87	13	20	28	8	3	72	50	43	107	14	78	197
88	9	13	18	12	27	3	21	15	100	55	210	106
89	1	28	11	9	8	70	28	61	150	137	133	204
90	9	16	24	21	70	64	4	39	19	150	78	102
91	9	13	3	15	49	27	39	43	107	83	35	130
92	23	5	14	25	72	49	10	71	134	1	198	178
93	4	7	13	27	24	9	25	60	201	83	178	141
94	5	6	8	8	49	3	31	26	137	83	7	200
95	20	13	17	13	8	27	45	27	107	137	191	137
96	13	20	20	17	27	8	8	45	100	188	71	181
97	28	28	21	18	70	70	68	47	150	14	57	110
98	16	9	14	16	65	49	33	71	19	100	38	76
99	13	20	4	27	24	72	54	52	107	134	56	131
100	5	6	27	26	27	24	30	42	134	107	18	39

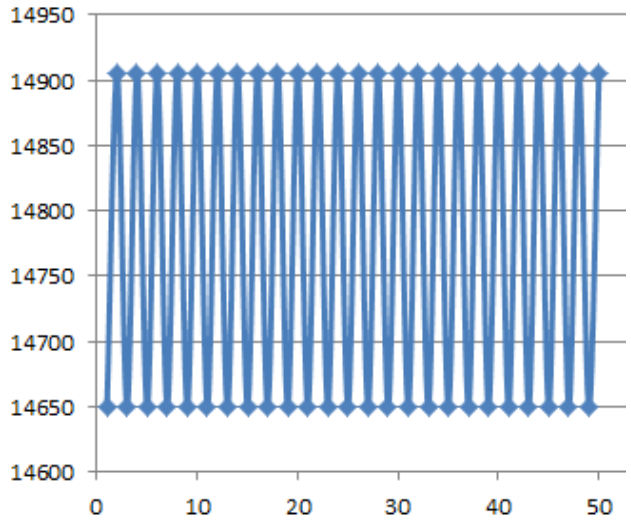
Grafik untuk berbagai kasus



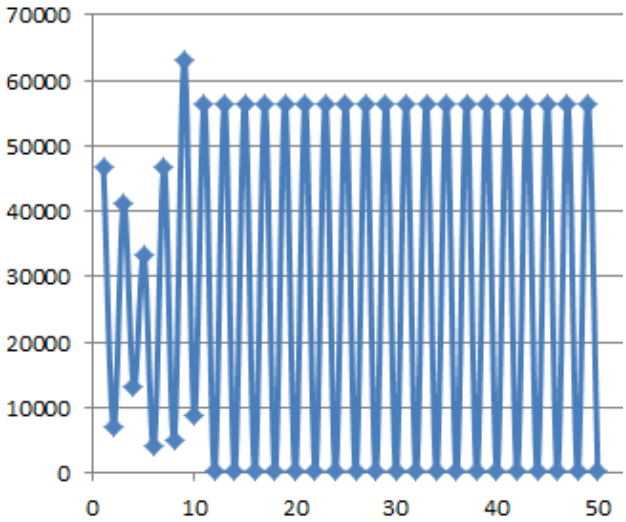
Feistel, seed=100



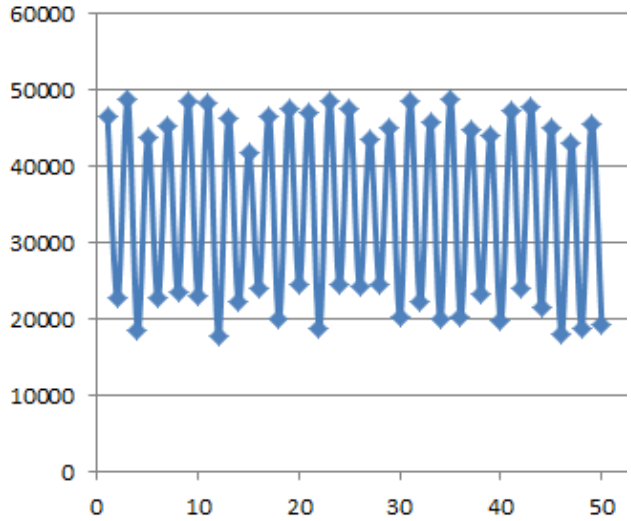
Feistel, n=73, seed=12345



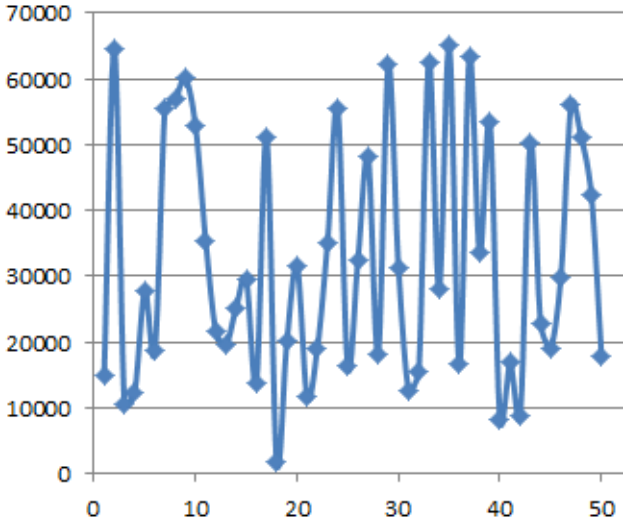
Feistel, n=29, seed=12345



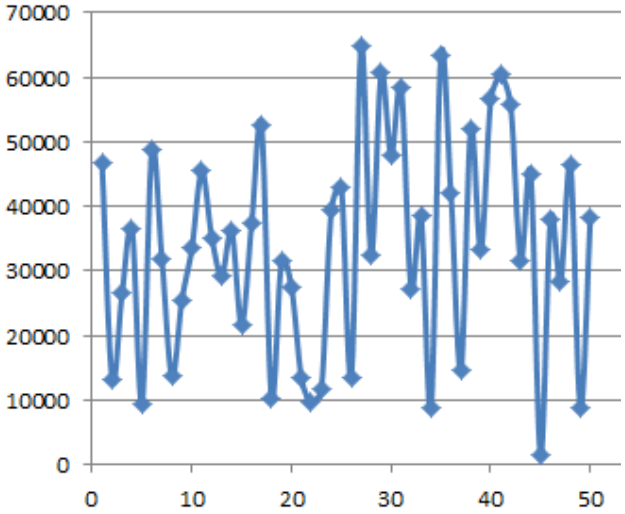
Feistel, n=73, seed=23478



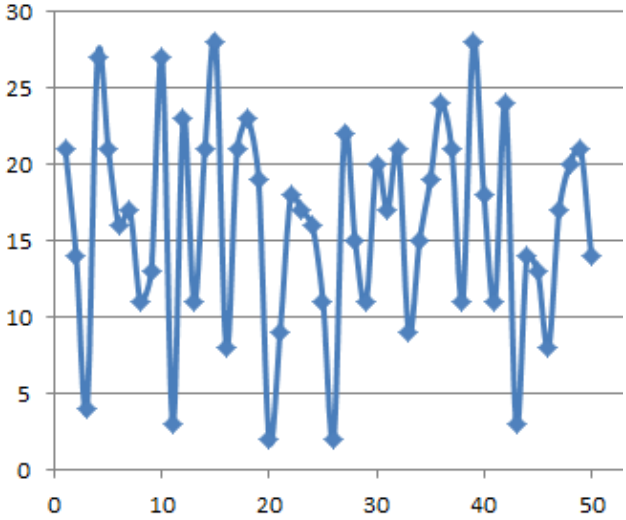
Feistel, n=29, seed=23478



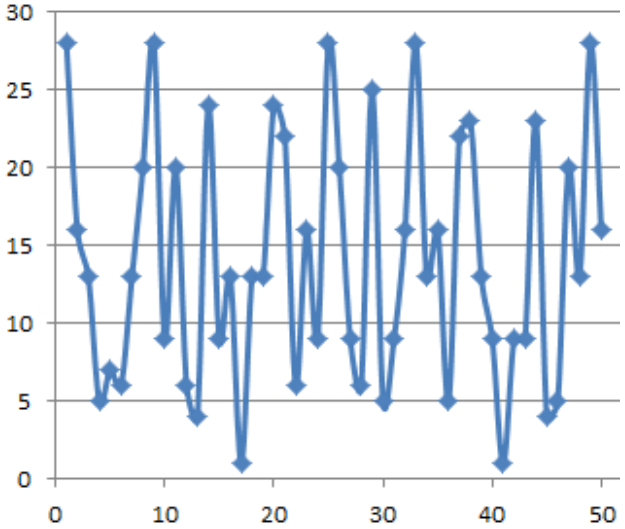
Feistel, n=211, seed=12345



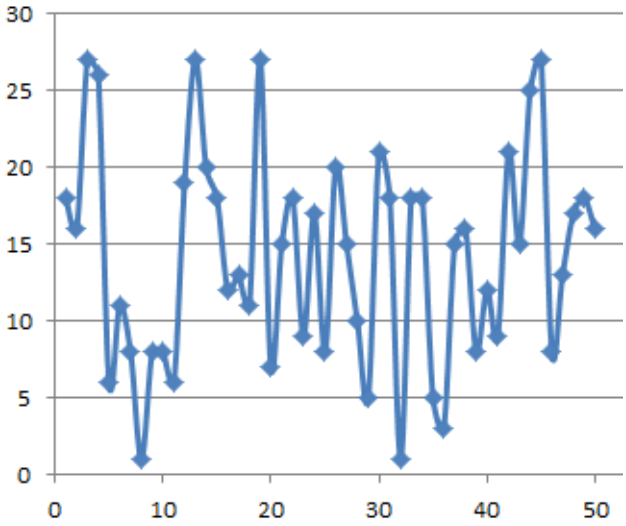
Feistel, n=211, seed=23478



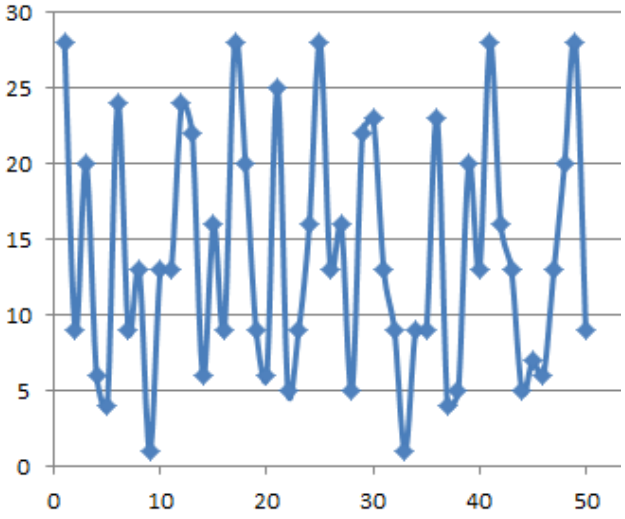
Perkalian, n=29, seed1=12345, seed2=23478



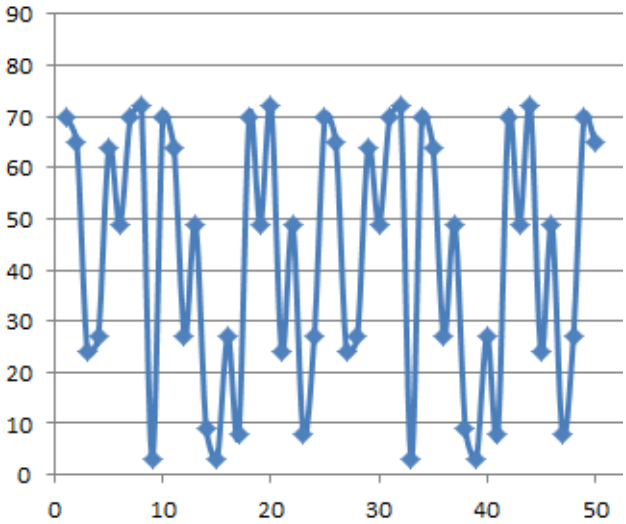
Perkalian n=29, seed1=100, seed2=12345



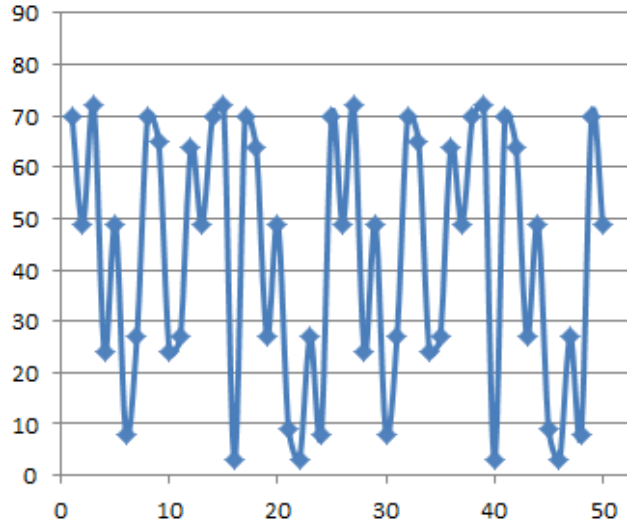
Perkalian, n=29, seed1=23478, seed2=100



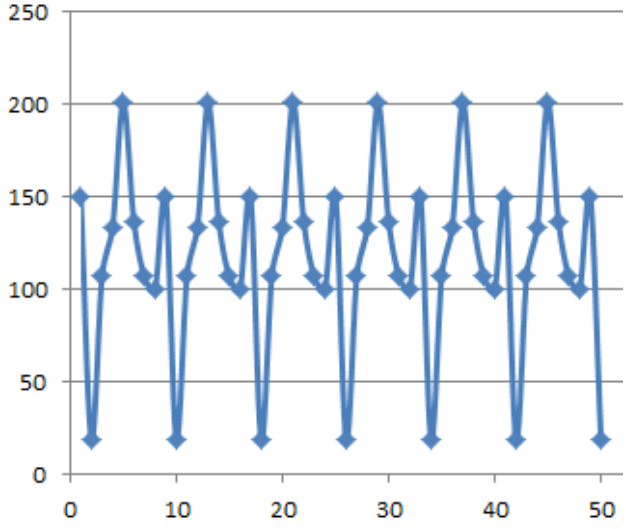
Perkalian, n=29, seed1=12345, seed=100



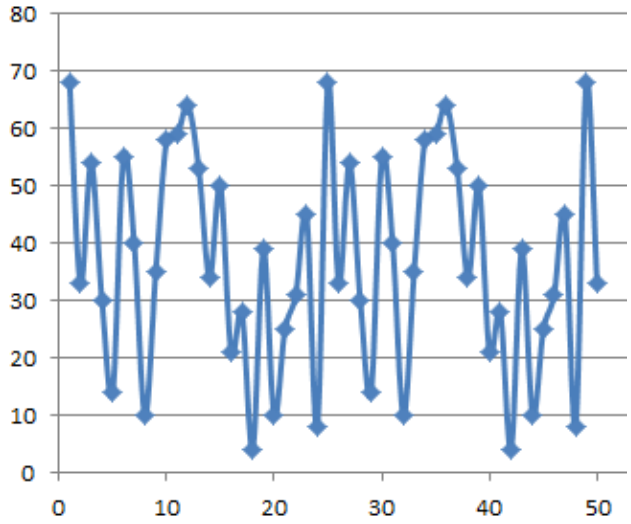
Perkalian, n=73, seed1=100, seed=12345



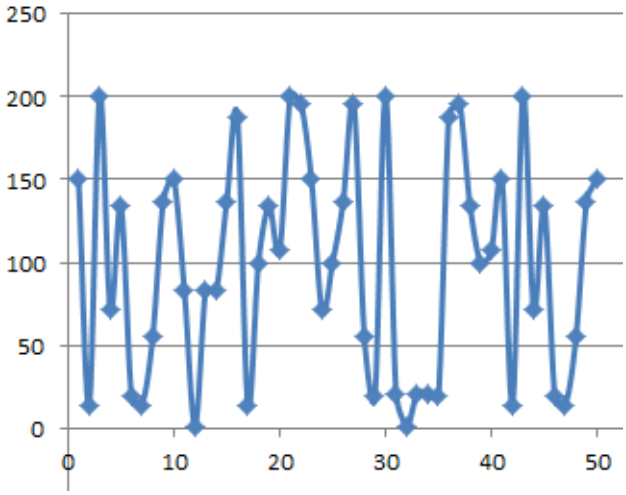
Perkalian, n=73, seed1=12345, seed2=100



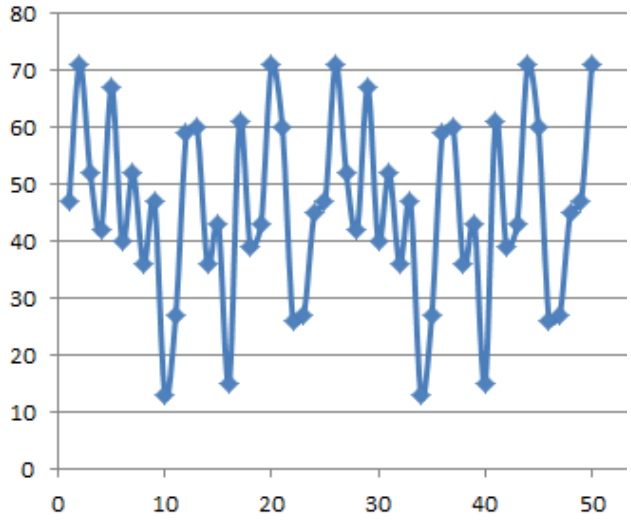
Perkalian, n=211, seed1=100, seed2=12345



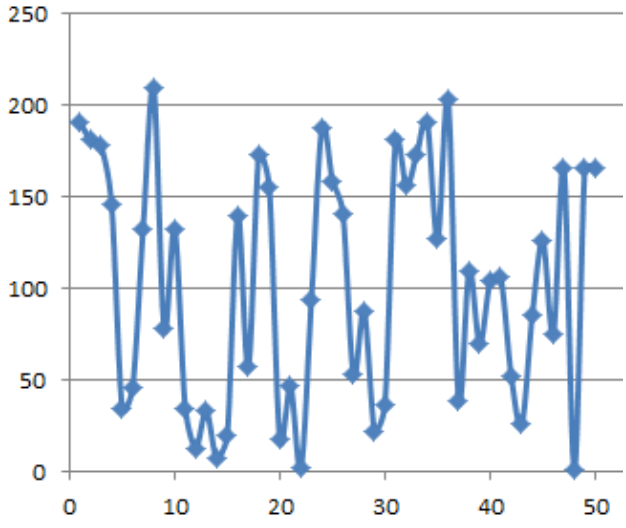
Perkalian, n=73, seed1=12345, seed2=23478



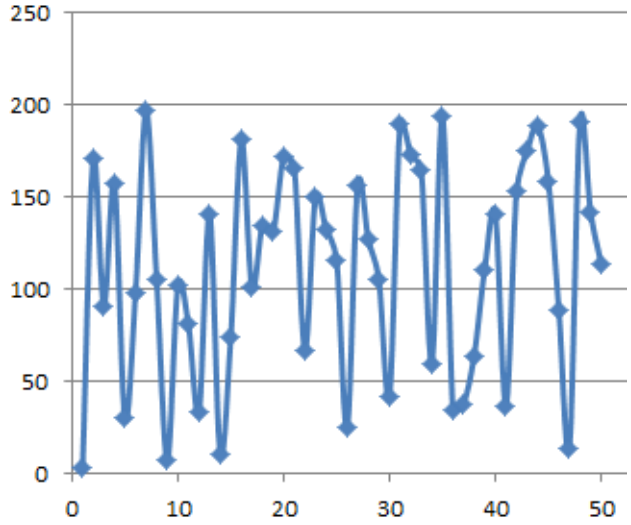
Perkalian, n=211, seed1=12345, seed2=100



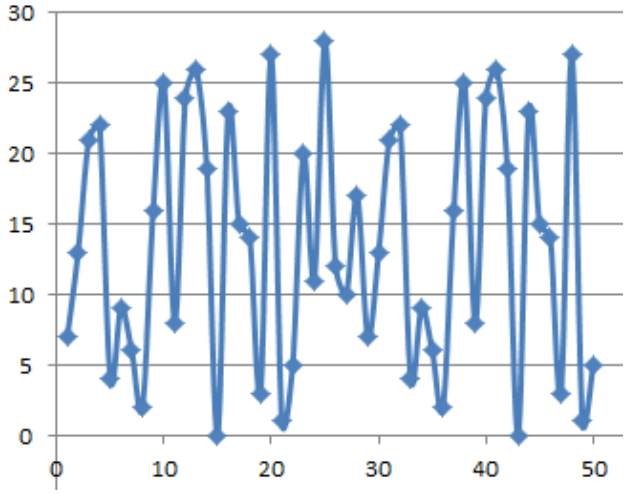
Perkalian, n=73, seed1=23478, seed2=100



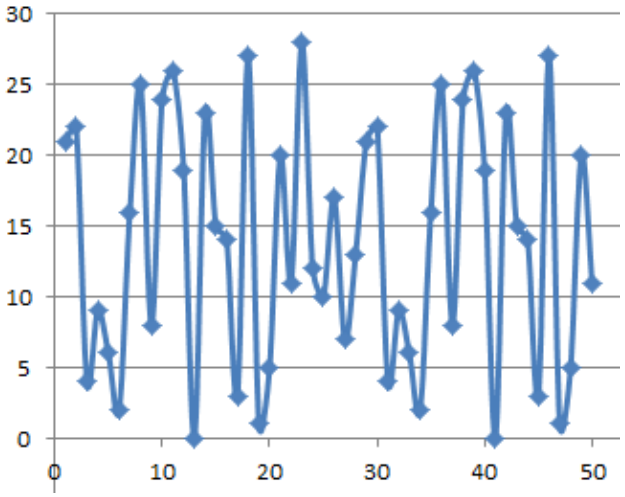
Perkalian, n=211, seed1=12345, seed2=23478



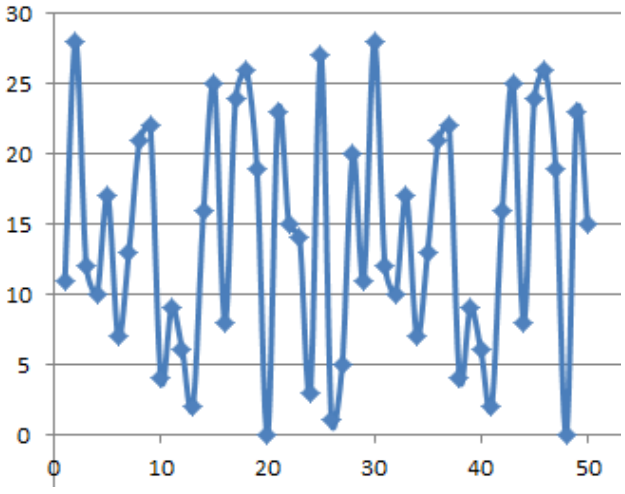
Perkalian, n=211, seed1=23478, seed2=100



LCG, seed=23478



LCG, seed=100



LCG, seed=12345