Vide Noir Number

Adriano Milyardi - 13509010
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Codezero91@yahoo.com

***Abstrak*— Pembangkit bilangan random secara luas digunakan dibanyak bidang dalam kehidupan sehari-hari seperti bidang statistika, permainan, krtiptografi dan lain-lain. Pembentukan bilangan random sendiri merupakan tugas yang penting dan biasa dilakukan dalam pemrogaman. Beberapa bidang seperti kriptografi dan beberapa algoritma numerik memerlukan bilangan random dengan tingkat random yang benar-benar tinggi. Untuk membuat sebuah bilangan random, dapat digunakan sebuah algoritma komputasi yang menghasilkan nilai-nilai yang secara kasat mata merupakan sesuatu yang random yang dibentuk dengan memanfaatkan suatu nilai awal yang disebut dengan key atau seed. Biasanya hal ini disebut dengan pseudorandom number generators.**

Kata Kunci*—Bilangan Random, Kriptografi, *Seed*, *Pseudorandom

I. PENDAHULUAN

Bilangan random atau bilangan acak, adalah sebuah bilangan yang terpilih seakan-akan secara " kebetulan" dari beberapa distribusi yang spesifik sehingga sekumpulan set dari bilangan ini akan menghasilkan kembali distribusi yang mendasarinya. Hampir selalu, angka-angka tersebut juga harus saling bersifat independen, sehingga tidak menghasilkan sebuah korelasi antara angka yang berurutan. Angka random yang dihasilkan oleh komputasi dengan memanfaatkan komputer biasanya disebut dengan nama bilangan *Pseudorandom* , dimana kata "random" sendiri disimpan sebagai keluaran dari hasil proses fisik yang tidak terduga. Ketika digunakan tanpa sebuah kualifikasi terlebih dahulu, kata "random" itu sendiri dapat memiliki arti "acak dengan sebuah distribusi yang seragam". Meskipun begitu, tentu saja bentuk distribusi yang lain pun mungkin terjadi, sebagai contoh adalah *Box-Muller Transformation* yang memungkinkan pasangan-pasangan dari bilangan acak yang seragam untuk ditranformasikan menjadi bilangan acak yang bersesuaian, namun dengan memiliki distribusi normal dua dimensi.

Hampir mustahil untuk menghasilkan sebuah string yang sangat panjang dari suatu digit yang acak lalu kemudian membuktikan bahwa bilangan-bilangan tersebut adalah acak. Namun anehnya, adalah sangat sulit bagi manusia sendiri untuk menghasilkan sebuah string dari sekumpulan bilangan yang benar-benar acak, dan progam komputer yang dapat dibuat, secara umum dapat melakukan prediksi angka apa yang akan ditulis oleh manusia berdasarkan pada apa yang sudah ada sebelumnya.

Sebenarnya ada beberapa cara yang umum digunakan untuk menghasilkan bilangan-bilangan pseudorandom, yang paling sederhana adalah dengan memanfaatkan metode konkuren linear. Cara lain yang juga sederhana dan elegan adalah dengan menggunakan *Elementary Cellular Automaton* dan memanfaatkan aturan ke-30, dan biasanya digunakan sebagai penghasil bilangan acak untuk bilangan integer yang besar dalam matematika. Kebanyakan dari penghasil bilangan acak membutuhkan sebuah spesifikasi nilai awal yang dimanfaatkan sebagai titik awal, yang dikenal dengan nama "Seed". Kelebihan dari bilangan random yang dihasilkan oleh algoritma diatas adalah kemampuannya untuk dianalisis dengan melakukan pengecekan pada *noise sphere*.

Bilangan acak sendiri banyak digunakan diberbagai bidang, sebagai contoh dalam statistika. Satu dari beberapa kegunaan penting adalah dalam pemilihan dari sample secara random dari sebuah populasi yang terbatas. Anggap ada 80 siswa dalam sebuah kelas dan akan diambil delapan siswa secara acak. Para siswa tersebut dapat diberi nomor dari 1-80. Kemudian akan ditinjau sebuah table bilangan random. Jika ada angka manapun yang ada dalam jarak 1-80, maka dapat kita ambil untuk sample. Jika ada angka diatas 80 maka diabaikan karena tidak sesuai dengan populasi yang ada. Angka-angka yang diambil bisa dari posisi mana saja dari tabel.

Dalam bidan kriptografi sendiri, bilangan acak memiliki peranan yang penting. Yang dibutuhkan dalam kriptografi adalah sebuah angka yang tidak dapat ditebak oleh pihak lawan selain dengan mencoba semua kemungkinan yang ada (metode *brute force*). Cara untuk memperolehnya sendiri ada banyak dan bervariasi satu samalainnya, tapi tak ada satupun yang dapat memastikan bahwa caranya benar-benar menghasilkan bilangan yang acak. Makadari itu, dapat dikatakan bahwa pembetukan bilangan acak adalah sebuah masalah seni dan asumsi-asumsi yang digunakan.

Beberapa manfaat dari bilangan acak yang diaplikasikan dalam kritptografi adalah sebagai berikut:

- Kebutuhan akan bit yang acak
  Seorang kriptografer terkadang membutuhkan bit-bit acak(atau beberapa nilai) untuk beberapa kebutuhan dalam kriptografi, tapi dua kegunaan paling umum adalah dalam menghasilkan kunci kriptografi (atau password)
- Ukuran untuk sebuah sumber acak
  Ada beberapa definisi dari sebuah sifat acak yang digunakan oleh para kriptografer, tapi secara umum hanya da satu ukuran dari sebuah sumber yang acak yakni musuh manapun yang sudah memiliki pengetahuan secara penuh atas software dan hardware, biaya yang dibutuhkan untuk membangun sebuah komputer yang dapat menyaingi dan melakukan pengujia, kemampuan untuk memasukan *bug* pada jaringan dan lain-lain, tidak boleh sampai mengetahui bit mana yang akan digunakan meskipun musuh tersebut mengetahui semua bit yang sudah digunakan sebelumnya.
- Sumber acak
  Sumber acak sendiri dapat dibedakan menjadi dua jenis, *true-random* dan *pseudo-random*. Yang terakhir adalah merupakan sebuah algoritma yang meniru yang pertama. Mekipun begitu, konsep dari sebuah keacakan sendiri masih sangat bersifat filosofis seperti ilmu fisika atau matematika dan masih sangat jauh untuk dipercahkan.
  Sumber yang *true-random* sendiri bisa dianggap tidak dapat di tebak dalam kondisi apapun, bahkan oleh seorang musuh dengan sumber daya komputasi yang tak terbatas sekalipun, sementara sumber *pseudo-random* hanya bermanfaat ketika menghadapi musuh dengan kemampuan komputasi terbatas.

II. PSEUDO RANDOM

A. Pseudorandomness

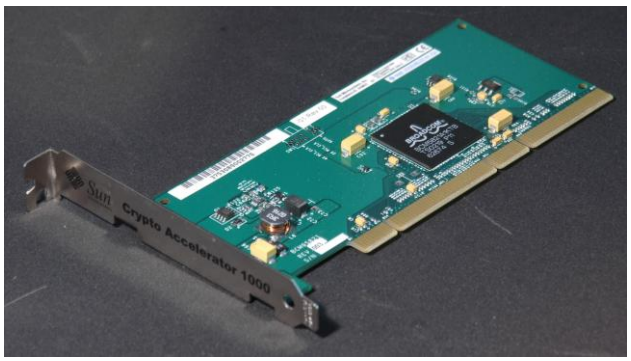
Sebuah proses *pseudorandom* adalah sebuah proses yang tampaknya secara sekilas bersifat acak namun sebernarnya tidak. Sebuah sekuens yang pseudorandom biasanya memperlihatkan *statistical randomness* ketika sedang dibuat oleh sebuah proses *causal deterministic*. Proses seperti itu pada kenyataannya jauh lebih mudah dibuat dibandingkan dengan yang benar-benar acak, serta memiliki kelebihan dimana proses tersebut dapat dilakukan berulang kali untuk menghasilkan nilai yang benar-benar sama-berguna untuk melakukan tes dan perbaikan pada perangkat lunak.

Dalam menghasilkan sebuah bilangan yang benar-benar acak membutuhkan ketelitian, keakuratan, dan pengukuran sistem beulang-ulang dari sebuah proses yang nondeterministic. Sebagai contoh, Linux menggunakan berbagai timing sistem (seperti ketikan keyboard dari user, I/O, atau pengukuran least-significant digit voltage) untuk menghasilkan sekumpulan angka-angka random. Linux berusaha untuk secara konstan mengisi ulang kumupan angka tersebut, bergantung pada level kepentignan, dan maka dari itu akan menghasilkan sebuah bilangan acak. Sistem ini adalah sebuah contoh, dan mirip dengan perangkat keras yang khusus untuk menghasilkan bilangan acak.

B. Pseudorandom Generator

Pada awalnya PRNG berbasis komputer diusulkan oleh John von Neumann pada tahun 1946, dikenal dengan nama metode middle-square. Algoritmanya adalah sebagai beritkut: “ambil angka acak, kuadratkan, ambil digit paling tengah dari hasil sebagai bilangan acak, dan gunakan sebagai seed pada iterasi selanjutnya. Masalah pada metode ini adalah kenyataan bahwa semua urutan mengulangi dirinya sendiri, beberapa dengan sangat cepat, misalkan saja “0000”. Von Neumann sdar dengan hal ini, namun dia menganggap bahwa pendekatan dari metode ini sudah tepat untuk kebutuhannya sendiri, dan khawatir bahwa perbaikan dari secara matematis akan menyembunyikan kesalahan yang mungkin terjadi dan bukan memperbaikinya.

Dikenal juga sebagai *Deterministic Random Bit Generator (DRBG)* , merupakan sebuah algoritma yang memiliki fungsi untuk menghasilkan sebuah urutan angka-angka yang mendekati sifat dari bilangan acak. Urutan ini sebenarnya tidaklah benar-benar acak dimana sebenarnya urutan ini ditentukan oleh sebuah set yang relative kecil yang digunakan sebagai nilai awal, disebut dengan nama PRNG state, yang termasuk juga atas sebuah seed yang benar-benar random. Meskipun urutan yang benar-benar dekat dengan acak sempurna dapat dihasilkan oleh perangkat keras khusus, bilangan *pseudorandom* penting dalam kegunaannya karena kecepatan dalam pembentukan angka dan kemampuannya untuk direproduksi ulang, dan pada akhirnya mereka menjadi bagian utama dalam simulasi, kriptografi, dan pembentukan procedural. Sifat-sifat statistical yang baik adalah bagian penting dalam hasil sebuah PRNG, dan beberapa algoritma yang biasa digunakan termasuk *linear congruential generators, lagged Fibonacci generators,* dan *linear feedback shift registers.*



Gambar 1. SSL Accelerator Computer Card sebuah hardware random number generator untuk menghasilkan kunci kriptografi untuk enkripsi data dan dikirim lewat jaringan komputer.

Sebuah PRNG diawalai dari sebuah state awal yang menggunakan seed state. Dengan state awal tersebut, maka sebuah urutan yang sama akan selalu dihasilkan. Periode dari sebuah PRNG didefinisikan sebagai keseluruhan state awal maksimum dari panjang prefix dari urutan yang tidak berulang. Periode tersebut terikat dengan ukuran dari state, dihitung dalam bits. Meskipun begitu, karena panjang dari periode mungkin untuk belipat ganda ketika setiap bit dari state ditambahkan, cukup mudah untuk membuat PRNG dengan periode yang cukup panjang untuk banyak aplikasi praktikal.

Pada kenyataannya sendiri, keluaran dari PRNG memiliki beberapa masalah antara lain:

- Periode yang lebih pendek dari yang diharapkan unuk beberapa seed states.
- Kekurangan distribusi keseragaman untuk angka yang dihasilkan dalam jumlah besar.
- Hubungan antara angka-angka yang berurutan.
- Distribusi dimensional yang lemah dari urutan hasil keluaran
- Jarak antara nilai yang pasti terdistribusi secara berbeda dari sebuah urutan distribusi acak.

III. ALGORITMA DASAR

Linear Congruential Generator

Linear Congruential Generator merupakan satu dari algrotima penghasil bilangan acak yang paling tua dan paling dikenal. Alasan dari pembuatannya adalah kebutuhan atas sebuah algoritma pembangkit bilangan acak yang mudah dimengerti dan mudah diimplementasikan dan cepat.

Pembangkit bilangan acak tersebut didefinisikan dengan sebuah relasi rekurens:

$$X_{n+1} \equiv ({}_aX_n + c) \text{Mod } m$$

Dimana X_n adalah sebuah urutan dari bilangan pseudorandom serta:

- m dengan syarat 0<m, merupakan modulus
- a dengan syarat 0<a<m, adalah faktor pengali
- c dengan syarat 0≤c<m, adalah faktor penambah
- X_0, dengan syarat 0≤X_0<m, adalah seed

Periode dari sebuah LCG secara umum adalah m, dengan beberapa yang lebih kecil dari itu. Dengan diberikan sebuah nilai c yang bukan nol, sebuah LCG akan memiliki sebuah periode penuh untuk semua nilai seed jika dan hanya jika.

1. Nilai c dan m adalah relatif prima.
2. Nilai a-1 bisa dibagi oleh semua faktor prima dari m
3. Nilai a-1 merupakan hasil perkalian dari 4 jika m adalah kelipatan dari 4 juga.

Meskipun LCG sendiri memungkina untuk menhasilkan angka-angka pseudorandom yang baik, namun hal tersebut benar-benar tergantung pada pemilihan nilai parameter c, m dan a.

Dari apa yang sudah terjadi sebelumnya, pemilihan yang salah mengakibatkan implementasi LGC yang tidak efektif.

IV. VIDE NOIR NUMBER

Vide Noir Number adalah algoritma yang dibuat oleh penulis yang merupakan pengembangan dari *Linear Congruental Generator* dengan menggabungkan dua buah *Linear Congruential Generator*. Algoritma ini memiliki ide utama sebagai berikut:

- Secara garis besar masih menggunakan rumus pseudo random number generator yakni:

$$X_{n+1} = (aX_n + b) \bmod m$$

- Nilai b merupakan nilai minimal, a dan m merupakan masukan yang digunakan untuk membuat angka random.
- a merupakan suatu bilangan random dari X0-Xn.
- Jumlah pengulangan adalah n, merupakan masukan dari user.
- Seed untuk bilangan awal yang digunakan adalah selalu 0.

Tahap awal dari pembangkitan bilangan acak dengan *Vide Noir Number* ini adalah dengan memita masukan dahulu dari user, beruma nilai dari parameter a, b, n dan m.. Bilangan yang dirandom merupakan bilangan integer memiliki range dari 0 sampai m-1.

LGR pertama digunakan untuk mengenerate nilai a yang akan digunakan pada LGR kedua. LGR pertama akan menghasilkan sebuah urutan random dari bilangan mana yang akan digunakan sebagai a dengan melakukan modulusnya dengan nilai n pada LGR kedua, dengan range dari X_0-X_{n-1}. LGR kedua digunakan untuk menghasilkan urutan bilangan random akhir sebagai keluaran.

V. PERBANDINGAN KEDUA ALGORITMA

Agar dapat melihat perbedaan tingkat random antara kedua algoritma, dapat digunakan sebuah parameter yang sama dan membandingkan hasil dari kedua algoritma tersebut. Kemudian akan dilakukan analisis kemunculan angka-angka yang ada pada kedua hasil tersebut.

Parameter masukan:

nilai a= 2
nilai b= 3
nilai n= 100
nilai m = 97

A. Linear Congruential Generator biasa

Berikut dalah bilangan acak yang dihasilkan dengan menggunakan algortima LCG biasa, bilangan yang dihasilkan memiliki urutan dari kolom paling kiri lalu kebawah, kemudian bergeser kolom terus ke sebelah kanan.

3	82	21	46
9	70	45	95
21	46	93	96
45	95	92	1
93	96	90	5
92	1	86	13
90	5	78	29
86	13	62	61
78	29	30	28
62	61	63	59
30	28	32	24
63	59	67	51
32	24	40	8
67	51	83	19
40	8	72	41
83	19	50	85
72	41	6	76
50	85	15	58
6	76	33	22
15	58	69	47
33	22	44	0
69	47	91	3
44	0	88	9
91	3	82	21
88	9	70	45

jumlah pseudo 0 ada 2	jumlah pseudo 50 ada 2
jumlah pseudo 1 ada 2	jumlah pseudo 51 ada 2
jumlah pseudo 3 ada 3	jumlah pseudo 58 ada 2
jumlah pseudo 5 ada 2	jumlah pseudo 59 ada 2
jumlah pseudo 6 ada 2	jumlah pseudo 61 ada 2
jumlah pseudo 8 ada 2	jumlah pseudo 62 ada 2
jumlah pseudo 9 ada 3	jumlah pseudo 63 ada 2
jumlah pseudo 13 ada 2	jumlah pseudo 67 ada 2
jumlah pseudo 15 ada 2	jumlah pseudo 69 ada 2
jumlah pseudo 19 ada 2	jumlah pseudo 70 ada 2
jumlah pseudo 21 ada 3	jumlah pseudo 72 ada 2
jumlah pseudo 22 ada 2	jumlah pseudo 76 ada 2
jumlah pseudo 24 ada 2	jumlah pseudo 78 ada 2
jumlah pseudo 28 ada 2	jumlah pseudo 82 ada 2
jumlah pseudo 29 ada 2	jumlah pseudo 83 ada 2
jumlah pseudo 30 ada 2	jumlah pseudo 85 ada 2
jumlah pseudo 32 ada 2	jumlah pseudo 86 ada 2
jumlah pseudo 33 ada 2	jumlah pseudo 88 ada 2
jumlah pseudo 40 ada 2	jumlah pseudo 90 ada 2
jumlah pseudo 41 ada 2	jumlah pseudo 91 ada 2
jumlah pseudo 44 ada 2	jumlah pseudo 92 ada 2
jumlah pseudo 45 ada 3	jumlah pseudo 93 ada 2
jumlah pseudo 46 ada 2	jumlah pseudo 95 ada 2
jumlah pseudo 47 ada 2	jumlah pseudo 96 ada 2

B. Vide Noir Number

Berikut dalah bilangan acak yang dihasilkan dengan menggunakan algoritma dari Vide Noir Number, bilangan yang dihasilkan memiliki urutan dari kolom paling kiri lalu kebawah, kemudian bergeser kolom terus ke sebelah kanan.

3	92	42	23
6	85	44	54
21	64	5	37
24	69	44	78
72	68	66	17
16	17	82	66
49	25	60	23
65	20	30	7
8	87	14	54
22	46	27	89
49	49	46	77
79	69	14	67
80	23	94	20
96	54	92	60

5	54	13	30
27	89	4	4
33	77	29	30
83	67	48	40
95	20	4	21
23	60	63	91
54	82	73	84
37	51	35	29
1	79	9	50
38	39	38	37
56	6	56	37

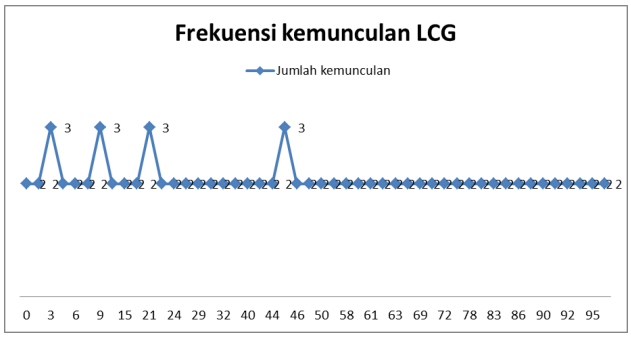
jumlah noir 1 ada 1	jumlah noir 49 ada 3
jumlah noir 3 ada 1	jumlah noir 50 ada 1
jumlah noir 4 ada 3	jumlah noir 51 ada 1
jumlah noir 5 ada 2	jumlah noir 54 ada 5
jumlah noir 6 ada 2	jumlah noir 56 ada 2
jumlah noir 7 ada 1	jumlah noir 60 ada 3
jumlah noir 8 ada 1	jumlah noir 63 ada 1
jumlah noir 9 ada 1	jumlah noir 64 ada 1
jumlah noir 13 ada 1	jumlah noir 65 ada 1
jumlah noir 14 ada 2	jumlah noir 66 ada 2
jumlah noir 16 ada 1	jumlah noir 67 ada 2
jumlah noir 17 ada 2	jumlah noir 68 ada 1
jumlah noir 20 ada 3	jumlah noir 69 ada 2
jumlah noir 21 ada 2	jumlah noir 72 ada 1
jumlah noir 22 ada 1	jumlah noir 73 ada 1
jumlah noir 23 ada 4	jumlah noir 77 ada 2
jumlah noir 24 ada 1	jumlah noir 78 ada 1
jumlah noir 25 ada 1	jumlah noir 79 ada 2
jumlah noir 27 ada 2	jumlah noir 80 ada 1
jumlah noir 29 ada 2	jumlah noir 82 ada 2
jumlah noir 30 ada 3	jumlah noir 83 ada 1
jumlah noir 33 ada 1	jumlah noir 84 ada 1
jumlah noir 35 ada 1	jumlah noir 85 ada 1
jumlah noir 37 ada 4	jumlah noir 87 ada 1
jumlah noir 38 ada 2	jumlah noir 89 ada 2
jumlah noir 39 ada 1	jumlah noir 91 ada 1
jumlah noir 40 ada 1	jumlah noir 92 ada 2
jumlah noir 42 ada 1	jumlah noir 94 ada 1
jumlah noir 44 ada 2	jumlah noir 95 ada 1
jumlah noir 46 ada 2	jumlah noir 96 ada 1
jumlah noir 48 ada 1	

C. Analisis pada hasil

Pertama-tama mari ditinjau telebih dahulu hasil dari LCG asli. Dari hasil tersebut dapat diperoleh beberapa fakta:

1. Hasil kumpulan bilangan acak yang dihasilkan memiliki pengulangan, dengan periode pengulangan setiap 45 bilangan yang dihasilkan.
2. Hanya terdapat 48 jenis bilangan yang dihasilkan oleh algoritma ini.
3. Mayoritas kemunculan adalah dua kali kemunculan dengan kemunculan tertinggi adalah tiga kali, masing-masing untuk bilangan 3, 9, 21 dan 45.

Berikut adalah grafik yang menunjukan frekuensi kemunculan dari bilangan yang dihasilkan. Untuk hasil yang lebih jelas dapat dilihat pada bagian lampiran.

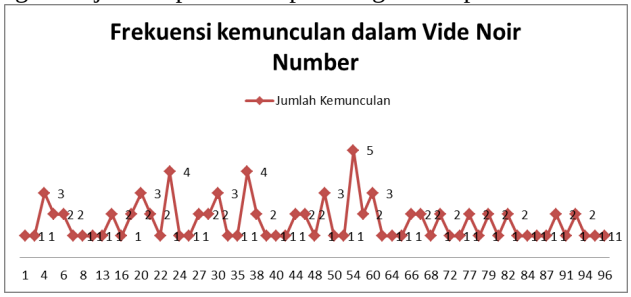


Gambar 2. Grafik kemunculan bilangan hasil LCG

Kemudian dilihat hasil dari Vide Noir Number. Dari hasil tersebut dapat diperoleh beberapa fakta, antar lain:
Pertama-tama mari ditinjau telebih dahulu hasil dari LCG asli. Dari hasil tersebut dapat kita peroleh beberapa fakta:

1. Hasil kumpulan bilangan acak yang dihasilkan ternyata tidak memiliki pengulangan, meskipun ada angka yang muncul lebih dari sekali namun tidak mengikuti pola berulang seperti yang ada pada LCG.
2. Terdapat 61 jenis bilangan yang dihasilkan.
3. Mayoritas kemunculan adalah satu kali kemunculan dengan kemunculan tertinggi adalah empat kali, untuk bilangan 23 dan 37.

Berikut adalah grafik yang menunjukan frekuensi kemunculan dari bilangan yang dihasilkan. Untuk hasil yang lebih jelas dapat dilihat pada bagian lampiran.



Gambar 3. Grafik kemunculan bilangan hasil Vide Noir Number

Pada percobaan kedua dengan memanfaatkan parameter yang sama, LCG dan Vide Noir Number sama-sama menghasilkan keluaran yang sama persis dengan pengujian yang pertama. Sama-sama memenuhi sifat dasar dari LCG dimana dengan masukan yang sama maka akan menghasilkan keluaran yang sama pula.

VI. KESIMPULAN

Melihat berbagai perbandingan dari hasil percobaan diatas, penulis dapat menganalisis dan menyimpulkan:

- Tingkat keacakan dari *Vide Noir Number* ini lebih baik jika dibanding *Linear Congurential Generator* biasa karena terbukti memiliki variasi bilangan yang lebih banyak dan pola pengulangan yang lebih tidak telihat. Hal ini disebabkan karena algoritma ini menggunakan dua buah *Linear Congurential Generator* sehingga dapat mengurangi adanya pola bilangan yang berulang dan bilangan mana yang akan dihasilkan selanjutnya akan lebih sulit untuk ditebak.
- Hasil dari Vide Noir Number lebih tersebar, karena menghasilkan lebih banyak angka acak jika dibandingkan dengan algoritma LCG biasa.
- Ketika dibandingkan dengan algoritma LCG biasa sebenarnya dapat disimpulkan bahwa Void Noir Number tidak memiliki kelemahan secara langsung, namun mungkin keterbatasan karena seed yang digunakan hanya angka 0 saja.

VII. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih terutama kepada Tuhan Yang Maha Esa karena anugerah yang diberikan-Nya makalah ini dapat diselesaikan. Penulis juga mengucapkan terima kasih kepada Bapak Ir. Rinaldi Munir, M.T. sebagai dosen pengajar kuliah IF3058 Kriptografi karena berkat kuliah dan referensi yang diberikan oleh beliau makalah ini dapat disempurnakan.

REFERENCES

[1] http://www.std.com/~cme/P1363/ranno.html.
[2] http://mathworld.wolfram.com/RandomNumber.html
[3] http://www.emathzone.com/tutorials/basic-statistics/application-of-random-numbers.html
[4] http://en.wikipedia.org/wiki/Pseudorandomness
[5] http://en.wikipedia.org/wiki/Linear_congruential_generator

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, 13 Mei 2012

ttd

Adriano Milyardi - 13509010

LAMPIRAN

A. ALGORITMA Vide Noir Number

```
   /*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package randomnumber;

import java.util.ArrayList;

/**
 *
 * @author asus
 */
public class Random {
   ArrayList<Integer> Randomlist1;
   ArrayList<Integer> Randomlist2;
   ArrayList<Integer> Randomlist3;
   int[] numlist1;
   int[] numlist2;

   public Random(){

   }

   public void pseudocalculate(int a,int b,int seed,int n){
   int temp = 0;
   Randomlist1 = new ArrayList<Integer>(n);

   for (int i= 0;i<n;i++){
      if (i == 0){
         Randomlist1.add(b%seed);
      } else{
         temp = (a*Randomlist1.get(i-1)+ b)%seed;
         Randomlist1.add(temp);
      }
   }
   }

   public void noircalculate(int a,int b,int seed,int n){
   int temp = 0;
   int atemp = 0;
   Randomlist2 = new ArrayList<Integer>(n);
   Randomlist3 = new ArrayList<Integer>(n);

   this.Randomlist2 = this.Randomlist1;
   this.pseudocalculate(a, 1, n , n);

   for (int i= 0;i<n;i++){
      if (i == 0){
         Randomlist3.add(b%seed);
      } else{
         atemp = this.Randomlist1.get(i);
         temp = (Randomlist1.get(atemp%i) *Randomlist3.get(i-1)+ b)%seed;
         Randomlist3.add(temp);
      }
   }
   }
```

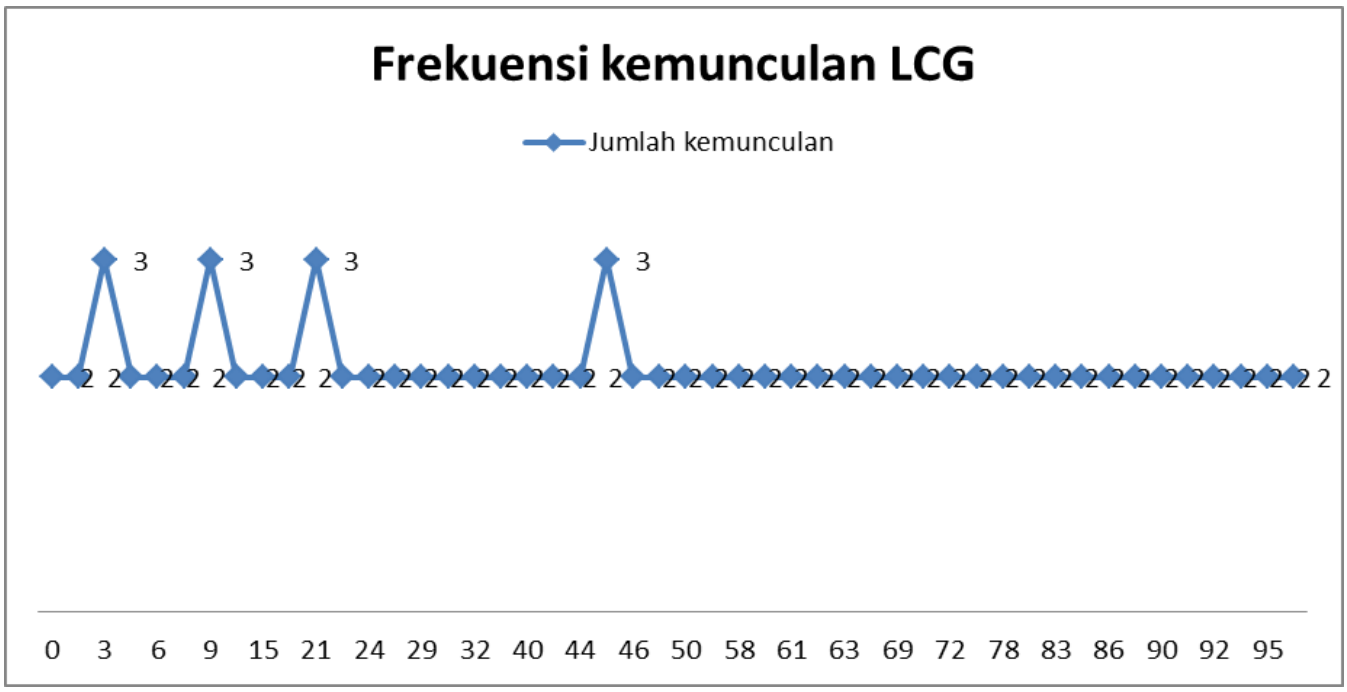
```
    public void makenumlist(int seed){
        numlist1 = new int[seed];
        numlist2 = new int[seed];
        int temp = 0;


            for (int i = 0; i < numlist1.length;i++){
                for (int n = 0; n < this.Randomlist2.size();n++){
                    if (i == this.Randomlist2.get(n)){
                        temp++;
                    }
                }
                numlist1[i] = temp;
                temp = 0;
                if (numlist1[i] != 0){
                System.out.println("jumlah pseudo "+i+" ada "+numlist1[i]);}
            }

            for (int i = 0; i < numlist2.length;i++){
                for (int n = 0; n < this.Randomlist3.size();n++){
                    if (i == this.Randomlist3.get(n)){
                        temp++;
                    }
                }
                numlist2[i] = temp;
                temp = 0;
                if (numlist2[i] != 0){
                System.out.println("jumlah noir "+i+" ada "+numlist2[i]);}
            }
        //System.out.println(Randomlist2.size());

    }
}
```

B. Grafik Kemunculan Hasil LCG



C. Grafik Kemunculan Hasil Vide Noir Number