COMPARISON BETWEEN LINEAR CONGRUENTIAL GENERATORS, WELL EQUIDISTRIBUTED LONG-PERIOD LINEAR (WELL) , AND SIMD-ORIENTED FAST MERSENNE TWISTER (SFMT) ALGORITHM TO GENERATE PSEUDORANDOM NUMBER

Marvello Oni
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18031@students.if.itb.ac.id
marvello_oni@yahoo.com

Abstract— **Random number atau bilangan acak adalah sebuah bilangan yang dihasilkan dari sebuah proses, yang keluarannya tidak dapat diprediksi dan secara berurutan tidak bisa dihasilkan bilangan yang sama. Proses yang menghasilkan random number disebut random number generator. Random number generator ini umumnya digunakan untuk mendukung proses-proses kriptografi seperti pada steganografi ataupun *initialization vector (IV)* pada algoritma kunci-simetri. Walaupun kelihatannya cukup sederhana, dari definisinya, tetapi pada kenyataanya cukup sulit untuk menghasilkan bilangan yang benar-benar acak dikarenakan semua bilangan acak yang dihasilkan dengan perhitungan dapat dihasilkan ulang. Dikarenakan itu bilangan acak yang dihasilkan dengan perhitungan disebut dengan *pseudorandom*.**

Pada makalah ini akan dibahas mengenai bagaimana cara memproduksi bilangan *pseudorandom* menggunakan algoritma LCG, WELL, dan SFMT. Kemudian melakukan pengujian terhadap hasil dari algoritma tersebut dalam segi periode pengulangan, kecepatan pembuatan bilangan acak dan apakah algoritma tersebut cukup aman untuk digunakan untuk fungsi-fungsi pada kriptografi.

***Kata Kunci* — Pseudorandom, LCG, WELL, SFMT, Kriptografi.**

I. PENDAHULUAN

Di era modern ini kriptografi telah menjadi bidang ilmu yang dipergunakan dalam kehidupan sehari - hari, banyak sekali informasi yang beredar yang perlu dirahasiakan agar tidak diketahui oleh pihak lain. Apalagi semenjak berkembangnya internet yang merupakan lalu lintas informasi yang bersifat terbuka (dapat dilalui oleh semua orang), semakin dirasakan kebutuhan akan algoritma kriptografi yang dapat memberikan keamanan terhadap kerahasiaan suatu data.

Dalam ilmu kriptografi modern sekarang ini, bilangan acak (random number) seringkali menjadi faktor penting dalam algoritma – algoritma kriptografi yang digunakan. Pada salah satu algoritma kunci simetri, block cipher misalnya, bilangan acak digunakan sebagai seed / initialization vector yang memicu berjalannya algoritma enkripsi / dekripsi tersebut. Selain itu, pada algoritma kunci asimetri (kunci publik), bilangan acak juga diperlukan untuk menghasilkan kunci publik dan kunci privat yang akan digunakan, contohnya saja dalam algoritma RSA. Hasil enkripsi dan dekripsi dari algoritma – algortima kriptografi tersebut sungguh tergantung dari seed bilangan acak yang diberikan.

Pada makalah ini akan dibahas tiga buah algoritma pembangkit bilangan acak, yaitu algoritma Linear Congruential Generators (LCG), algoritma Well Equidistributed Long-period Linear (WELL) dan SIMD-oriented Fast Mersenne Twister (SFMT). Di sini akan dibahas masing – masing pengujian terhadap hasil dari algoritma tersebut dalam metode pembentukan bilangan acak semu, segi periode pengulangan, kecepatan pembuatan bilangan acak dan apakah algoritma tersebut cukup aman untuk digunakan untuk fungsi-fungsi pada kriptografi.

II. DASAR TEORI

1. Pembangkit Bilangan Acak Semu

Pseudorandom Number Generator (PNRG) atau dalam bahasa indonesia Pembangkit bilangan acak semu adalah sebuah algoritma yang membangkitkan sebuah deret bilangan yang tidak benar-benar acak. Keluaran dari pembangkit bilangan acak semu hanya mendekati beberapa dari sifat-sifat yang dimiliki bilangan acak. Walaupun bilangan yang benar-benar acak hanya dapat dibangkitkan oleh perangkat keras pembangkit bilangan acak, bukannya oleh perangkat lunak komputer, akan tetapi bilangan acak semu banyak digunakan dalam beberapa seperti untuk simulasi dalam ilmu fisika, matematik, biologi dan sebagainya, dan juga merupakan hal yang sangat penting dalam dunia kriptografi. Beberapa algoritma enkripsi baik yang simetris maupun nirsimetris memerlukan bilangan acak sebagai parameter masukannya seperti parameter kunci pada algoritma kunci publik dan pembangkitan initialization vector (IV) pada algoritma kunci-simetri. Walaupu terlihat sederhana untuk mendapatkan bilangan acak,tetapi diperlukan analisis matematika yang teliti untuk membangkitkan bilangan seacak mungkin. Robert R. Coveyoudari Oak Ridge National Laboratory menulis sebuah artikel berjudull "The generation of random numbers is too important to be left to chance" dan John von Neumann menulis sebuah ide yang lebih menarik "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin"

Berikut ini beberapa pembangkit bilangan acak semu :

1. linear congruential generators (LCG)
2. lagged Fibonacci generators
3. linear feedback shift registers
4. generalised feedback shift registers
5. Blum Blum Shub
6. Mersenne twister
7. Well Equidistributed Long-period Linear (WELL)
8. SIMD-oriented Fast Mersenne Twister (SFMT).

2. Pembangkit Bilangan Acak yang Aman untuk Kriptografi

Pembangkit bilangan acak yang cocok untuk kriptografi dinamakan *cryptographically secure pseudorandom generator* (*CSPRNG*).

Persyaratan *CSPRNG* adalah:

- Secara statistik ia mempunyai sifat-sifat yang bagus (yaitu lolos uji keacakan statistik).
- Tahan terhadap serangan (*attack*) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan.

3. Linear Congruential Generators (LCG)

Linear Congruential Generators (LCG) adalah alah satu pembangkit bilangan acak tertua dan sangat terkenal. LCG adalah algoritma yang sering diimplementasikan pada beberapa bahasa pemrograman untuk membangkitkan bilangan acak. LCG didefinisikan dalam relasi rekurens (Schneier, 1996):

$Xn°(aXn -1+ b)\bmod m$, yang dalam hal ini,
Xn = bilangan acak ke-n dari deretnya
$Xn - 1$ = bilangan acak sebelumnya
a = factor pengali
b = penambah (*increment*)
m = modulus
(a , b , dan m semuanya konstans)

Kunci pembangkit adalah X^0 yang disebut *seed (secret seed)*.

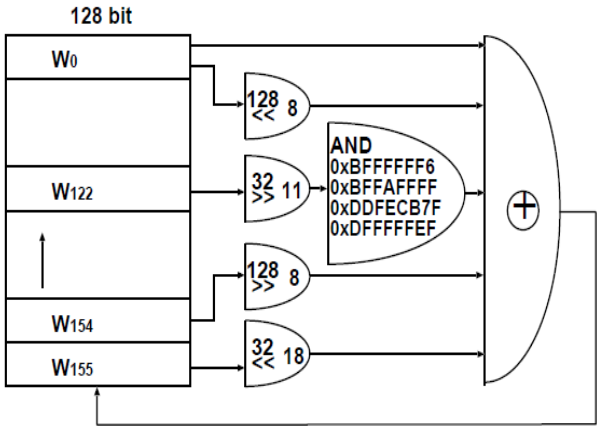
LCG mempunyai periode tidak lebih besar dari m, dan pada kebanyakan kasus periodenya kurang dari itu. LCG mempunyai periode penuh (m-1) jika memenuhi syarat berikut:

1. b relative prima terhadap m
2. a–1 dapat dibagi dengan semua faktor prima dari m
3. a–1 adalah kelipatan 4 jika m adalah kelipatan 4
4. m > maks (a, b, X0)
5. a > 0, b > 0

Meskipun LCG secara teoritis mampu menghasilkan bilangan acak yang lumayan, namun sangat *sensitive* terhadap pemilihan nilai-nilai a, b, dan m. pemilihan nilai-nilai yang tidak sesuai dapat mempengaruhi implementasi pada LCG.

4. SIMD-oriented Fast Mersenne Twister (SFMT)

SIMD-oriented Fast Mersenne Twister (SFMT) adalah sebuah varian baru dari Mersenne Twister yang dikenalkan oleh Mutsuo Saito dan Makoto Mastsumoto pada tahun 2006. SFMT adalah sebuah generator Linear Feedbacked Shift Register (LFSR) yang menghasilkan sebuah bilangan integer acak semu 128 bit. Algoritma ini dirancang dengan model CPU modern yang bersifat pararel dan instuksi SIMD. Algoritma ini jauh lebih cepat dibandingkan dengan Mersenne Twister, pada sebagian besar bahasa pemrogramman. Algoritma dari SFMT ini dapat digambarkan seperti gambar berikut :



Gambar 1 Algoritma SFMT

5. Well Equidistributed Long-period Linear (WELL)

Well Equidistributed Long-period Linear (WELL) adalah generator bilangan pseudorandom yang dikembangkan pada tahun 2006 oleh F. Panneton, P. L'Ecuyer, dan M. Matsumoto yang didasarkan pada rekurensi linier modulo 2 pada field biner terbatas F_2. Berikut adalah algoritma dari WELL :

```
z0 =  rotp(vTi,r-2, vTi,r-1)T;
z1 =  T0vi,0 _ T1vi,m1 ;
z2 =  T2vi,m2 _ T3vi,m3 ;
z3 =  z1 _ z2;
z4 =  T4z0 _ T5z1 _ T6z2 _ T7z3;
vi+1,r-1 =    vi,r-2 & mp;
for j = r - 2, . . . , 2, do
vi+1,j =  vi,j-1;
vi+1,1 =  z3;
vi+1,0 =  z4;
output yi = vi,1 or yi = vi,0.
```

III. Analisis

Untuk melakukan analisi dilakukan pembuatan angka secara random menggunakan algoritma LCG, SFMT, dan WELL sebanyak 10^4 bilangan integer dan kemudian dicetak dilayar. Percobaan ini dilakukan dengan menggunakan komputer Intel Pentium 4 CPU 2.00 GHz 2.00 GHz, 1.50 GB RAM dengan menggunakan bahasa C++.

A. Metode penghasilan Bilangan Acak

Pada LCG, penghasilan bilangan acak dilakukan dengan melakukan perhitungan sederhana dengan menggunakna modulus. Pada algoritma ini digunakan 3 buat parameter yang bersifat konstan. Inisialisasi ketiga parameter mempengaruhi tingkat keamanan dan kecepatan dari algoritma ini. Algoritma ini juga memanfaatkan bilangan acak yang telah dihasilkan oleh perhitungan sebelumnya. Pada algoritma ini *seed* dimasukan sebagai bilangan acak ke-0 dimana bilangan acak ke-1 adalah hasil perhitungan yang pertama.

Pada SFMT menggunakan Linear Feedbacked Shift Register (LFSR) menggunakan 156 buah bilangan 128 bit. Yang dalam pemrosesannya dapat dibagi menjadi 4 buah bilangan 32 bit maupun diproses secara utuh. Pada setiap penghasilan bilangan acak maka bilangan ke 2 sampai dengan 156 akan digeser menjadi bilangan ke 1 sampai dengan 155 dan kemudian bilangan ke 156 akan diinisialisasi dengan bilangan acak yang dibentuk sebelumnya. Kemudian semua bilangan

tersebut akan dioperasikan dengan operasi XOR dengan urutan tertentu. Pada algoritma ini *seed* dimasukan sebagai bilangan acak ke-0 dimana bilangan acak ke-1 adalah hasil perhitungan yang pertama.

Pada WELL, bilangan acak dihasilkan dengan menggunakan perhitungan XOR dan juga pengulangan. Dimana digunakan paremeter-paremeter yang sudah diinisialisasi sebelumnya kemudian dilakukan perhitungan XOR yang dilanjutkan dengan perulangan sebanyak panjang bit – 3. Pada algoritma ini *seed* dimasukan sebagai bilangan acak ke-0 dimana bilangan acak ke-1 adalah hasil perhitungan yang pertama.

Bila dilihat dari metode penghasilan bilangan acaknya maka dapat dilihat bahwa SFMT dan WELL mempunyai metode yang cukup sama yaitu menggunakan opereasi XOR dan bekerja pada bentuk bit bukan pada bentuk desimal seperti LCG. Perhitungan yang dlakukan oleh LCG sangat sederhana dibandingkan dengan kedua algoritma lainnya yang memiliki banyak parameter dan menggukan lebih banyak *resource* dari komputer.

B. Kecepatan

Dari hasil uji coba diperoleh hasil sebagai berikut :

LCG	SFMT	WELL
0.527	1.017	1.970

Table 1 Waktu Penghasilan angka

Dari data diatas dapat dilihat bahwa rata-rata selisih waktu dari ketiga algoritma tesebut adalah 0.962. LCG mempunyai waktu yang lebih cepat dibandingkan dengan kedua algoritma yang lain. Kemudian diikuti oleh SFMT dan terakhir adalah WELL.. Hal ini bukanlah sesuatu yang membingungkan mengingat perhitungan yang dilakukan oleh LCG adalah yang paling sederhana sehingga waktu pemrosesan untuk menghasilkan bilangan acak sangat cepat. Dan bila dilihat SFTM memiliki waktu lebih cepat dibandingkan dengan WELL meskipun perhitungan pada algoritma WELL dapat dibilang lebih sederhana dibandingkan SFMT. Hal ini dapat terjadi dikarenakan implementasi dari SFMT yang memanfaatkan arsitektur dari komputer seperti adanya pararelism sehingga pemrosesan dari SFMT dapat lebih cepat dibandingkan dengan WELL.

CPU	mrg	rand48	LCG	SFMT	WELL
Pentium M	3.277	1.417	0.527	1.017	1.970

Table 2 Waktu Penghasilan angka Algoritma Lain

Bila dibandingkan dengan waktu rata-rata dari algoritma lain waktu dari algoritma LCG, SFMT maupun WELL tergolong cukup cepat.

C. Pengulangan

Dari segi pengulangan, pada kasus ini LCG menggunakan parameter sebagai berikut :

$M = 2^{32}$

A = 1103515245

B = 12345

Seperti yang diketahui bahwa periode dari LCG paling banyak adalah M. Sehingga pada kasus ini LCG akan berulang setelah dilakukan pembentukan bilangan acak sebanyak 2^{32} kali atau sebanyak 4294967296 kali. Dikarenakan periode ini bergantung pada nilai M sehingga pada implementasi tertentu periode dari LCG akan lebih panjang. Tetapi bila nilai M ini terlalu besar maka akan memepengaruhi kecepatan dari penghasilan angka itu sendiri.

Sedangkan untuk SFMT mempunyai periode sebesar $2^{19937} – 1$[2]. Hal ini dikarenakan oleh SFMT mempunyai sifat yang rekursif dan pengiinisialisasian parameter. Dimana diambil nilai p =13c9e684 00000000 00000000 00000001(dalam bentuk hexadesimal) adalah sebuah vektor sertifikat periode untuk SFMT. Maka dengan menggunakan initial state *s* = (w0,w1, . . .

,w$N-1$) memenuhi w0 :p = 1, maka periode dari algortima SFMT adalah $2^{19937} - 1$.

Pada algoritma WELL diperoleh bahwa periode dari algoritma ini berkisar ($2^{1024}-1$ to $2^{44497} - 1$) bergantung pada nilai inisialisasi awalnya.

D. Keamanan

Segi keamanan bergantung pada periode dari sebuah algoritma. Dari hasil analisis sebelumnya kita dapat menarik asumsi bahwa algoritma WELL lebih aman dibandingkan dengan SFMT ataupun LCG. Hal ini dikarenakan oleh periodenya yang cukup panjang yaitu 2^{44497}-1 atau setara dengan 8,54509824303633803193300705318 4e+13394. Selain itu dikarenakan WELL menggunakan bilangan yang kecil dengan tingkat ketelitian yang tinggi, membut angka yang dihasilkan oleh algoritma ini tidak mempunyai sebuah pola tertentu. Ini membuat algoritma WELL lebih cocok digunakan untuk fungsi-fungsi kriptografi.

Algoritma SFMT pun mempunyai keamanan yang cukup baik dengan periode sebesar 4294967296. Ini lebih baik dibandingkan dengan algoritma pendahulunya yaitu Mersenne Twister yang sekarang banyak digunakan. Hal ini disebabkan karena SFMT menggunakan Linear Feedbacked Shift Register (LFSR) dan juga operasi XOR dalam bit sehingga kemungkinan sebuah angka untuk muncul berulang kali menjadi lebih sedikit.

Sedangkan LCG mempunyai tingkat keamaanan paling rendah diatara ketiga algoritma tersebut karena sangat bergantung pada nilai inisialisasinya. Tetapi dengan membuat nilai inisialisasi tersebut menjadi cukup tinggi sehingga tingkat keamanan dari LCG dapat terjamin menyebabkan biaya untuk melakukan perhitungan menjadi lebih besar.

IV. KESEIMPULAN

Dari analisis diatas kita dapat melihat bahwa algoritma Linear Congruential Generators (LCG), algoritma Well Equidistributed Long-period Linear (WELL) dan SIMD-oriented Fast Mersenne Twister (SFMT) dapat digunakan untuk menghasilkan bilangan acak semu (pseudorandom).

Dari hasil analisis diperoleh bahwa dari segi kecepatan LCG membutuhkan waktu yang paling pencek dalam menghasilkan bilangan acak. Sedangkan dari sisi periode WELL mempunyai periode yang paling panjang hal ini membuat WELL mempunyai tingkat keamanan paling tinggi.

Meningkatnya tingkat keamanan akan berbanding terbalik dengan kecepatan. Sehingga bila diingiinkan keamanan yang lebih tinggi maka kecepatan akan berkurang dan bila menginginkan kecepatan yang tinggi maka keamanan akan berkurang.

Karena mempunyai tingkat keamanan yang cukup tinggi dan kecepatan yang cukup cepat, memungkinkan Algoritma Well Equidistributed Long-period Linear (WELL) dan SIMD-oriented Fast Mersenne Twister (SFMT) untuk digunakan dalam berbagai algoritma untuk fungsi-fungsi kriptografi. Selain itu berdasarkan [2] dan [3] algoritma Algoritma Well Equidistributed Long-period Linear (WELL) dan SIMD-oriented Fast Mersenne Twister (SFMT) tergolong *cryptographically secure pseudorandom generator* (*CSPRNG*).

Hanya saja pemilihan algoritma ini bergantung pada tingkat keamanan dan kecepatan yang digunakan pada implementasi dimana bila diinginkan hasil yang lebih cepat maka dapat digunakan algoritma SIMD-oriented Fast Mersenne Twister (SFMT) dan bila diinginkan keamanan yang lebih baik maka dapat digunakan algoritma Well Equidistributed Long-period Linear (WELL).

REFERENCES

[1] L'Ecuyer, Pierre; Panneton, François; Matsumoto, Makoto (2006), "*Improved Long-Period Generators Based on Linear Recurrences Modulo 2*".

[2] Mutsuo Saito and Makoto Matsumoto, "SIMD-oriented Fast Mersenne Twister: a 128-bit Pseudorandom Number Generator", Monte Carlo and Quasi-Monte Carlo Methods 2006, Springer, 2008.

[3] Mutsuo Saito and Makoto Matsumoto, " An Application of Finite Field: Design and Implementation of 128-bit Instruction-Based Fast Pseudorandom Number Generator".

[4] Bolte, joe. "Linear Congruential Generators", Wolfram Demonstrations Project.

[5] http://en.wikipedia.org/wiki/Linear_congruential_generator

[6] http://en.wikipedia.org/wiki/Mersenne_twister

[7] http://en.wikipedia.org/wiki/Well_Equidistributed_Long-period_Linear

[8] SIMD from wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/SIMD.

[9] http://www.iro.umontreal.ca/~panneton/WELLRNG.html

[10] http://www.math.sci.hiroshimau.ac.jp/~mmat/MT/SFMT/index.html

[11] G. Marsaglia. Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6,2003.

[12] Ir. Rinaldi Munir, M.T., Diktat Kuliah IF3058 Kriptografi, Departemen Teknik Informatika ITB, 2005.

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, 09 Mei 2011

ttd

Marvello Oni

13508031