Analisis Perbandingan Berbagai Teknik Pembangkit Bilangan *Pseudorandom* Berdasarkan Uji Kerandoman

Evan – 13506089
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16089@students.if.itb.ac.id

Abstrak

Bilangan random merupakan salah satu unsur penting dalam ilmu kriptografi. Bilangan random diperlukan untuk menimbulkan unsur ketidakpastian dalam beberapa metode penyandian pesan sehingga pesan menjadi lebih sulit diserang oleh kriptanalis. Pembangkit bilangan random merupakan alat yang biasa digunakan sebagai sumber dari bilangan random yang digunakan dalam aplikasi kriptografi.
Terdapat dua macam pembangkit bilangan random, pembangkit yang menghasilkan bilangan random sesungguhnya (*truly random*) dan pembangkit yang menghasilkan bilangan *pseudorandom*. Bilangan *pseudorandom* merupakan bilangan yang dihasilkan melalui proses komputasi, sehingga barisan bilangan yang dihasilkan oleh pembangkit bilangan *pseudorandom* sebenarnya dapat dihitung dari bilangan-bilangan sebelumnya.
Pembangkit bilangan *pseudorandom* yang baik dapat menghasilkan barisan bilangan random yang sulit dibedakan dengan bilangan random sesungguhnya. Kerandoman sebuah barisan bilangan random dapat menjadi tolak ukur yang menyatakan seberapa baik sebuah pembangkit bilangan *pseudorandom* merepresentasikan bilangan random sesungguhnya. Kerandoman ini dapat diuji dengan berbagai metode pengujian yang telah dikembangkan, seperti misalnya metode statistik, uji frekuensi, uji serial, uji poker, uji jarak, dan uji Diehard yang merupakan kumpulan dari 12 macam pengujian.
Makalah ini bertujuan untuk melakukan analisis perbandingan berbagai metode pembangkit bilangan *pseudorandom* yang telah dikembangkan, mulai dari metode *middle-square*, *Linear Congruential Generator*, *Mersenne twister*, dan CSPRNG (*Cryptographically Secure Pseudorandom Number Generator*). Perbandingan dilakukan berdasarkan hasil pengujian dengan uji kerandoman yang telah disebutkan di atas, termasuk uji Diehard. Hasil dari pengujian tersebut diharapkan dapat menjadi bahan pertimbangan dalam memilih metode pembangkit bilangan *pseudorandom* untuk aplikasi kriptografi.

Kata kunci: pembangkit bilangan random, *truly random*, *pseudorandom*, uji kerandoman, metode statistik, uji Diehard.

1 Pendahuluan

Dalam kriptografi, bilangan random seringkali digunakan dalam berbagai algoritma. Misalnya saja penggunaan pada cipher aliran, algoritma cipher block yang menggunakan Initial Value, dan algoritma pembangkitan kunci RSA. Untuk memakai bilangan random diperlukan sebuah pembangkit bilangan random. Pembangkit bilangan random yang ideal sebaiknya dapat menghasilkan bilangan-bilangan yang tidak dapat ditemukan polanya, sehingga menyulitkan kriptanalis untuk memecahkan kode. Tapi ada satu masalah dengan pembangkit bilangan random yang memanfaatkan komputasi matematis, atau biasa disebut pembangkit bilangan *pseudorandom*. Bilangan *pseudorandom* yang dihasilkan melalui proses komputasi selalu dapat ditemukan polanya apabila metode komputasi dan nilai awalnya sudah diketahui. Bahkan pola bilangan random yang dihasilkan dari metode komputasi yang lemah dapat dipecahkan oleh kriptanalis tanpa perlu mengetahui nilai awalnya. Karena itu dalam kriptografi diperlukan pembangkit bilangan *pseudorandom* yang kuat, tahan terhadap serangan-serangan dari kriptanalis.
Pengujian kekuatan sebuah pembangkit bilangan *pseudorandom* dapat dilakukan dengan uji kerandoman. Uji kerandoman dilakukan dengan menganalisis pola distribusi dari sebuah barisan bilangan. Semakin random sebuah barisan bilangan, pola distribusinya semakin tidak kelihatan. Dengan mendeteksi frekuensi kemunculan pola-pola tertentu pada barisan bilangan, kita dapat menentukan seberapa random barisan bilangan tersebut. Dalam makalah ini kita hanya membahas kerandoman dari barisan bilangan biner, yaitu barisan bilangan yang terdiri dari bit 0 dan 1 saja.

2. Uji Kerandoman

Terdapat berbagai metode untuk menguji kerandoman sebuah barisan bilangan biner, di antaranya metode statistik, metode transformasi, dan metode kompleksitas. Metode statistik menggunakan metode-metode ilmu statistik seperti perhitungan frekuensi, distribusi bilangan, peluang, dll. Sebuah barisan bilangan dikatakan random secara statistik jika barisan tersebut tidak memiliki pola berulang tertentu yang dapat dikenali. Pengujian random dengan metode statistik pada dasarnya menggunakan perhitungan distribusi peluang untuk sebuah kejadian random, dan membandingkannya dengan barisan bilangan yang diperiksa. Beberapa parameter statistik yang dapat menjadi patokan kerandoman sebuah barisan bilangan adalah:

a. Entropi: ukuran kepadatan informasi yang terkandung dalam sebuah barisan bilangan. Barisan bilangan random memiliki kepadatan informasi yang tinggi, jadi entropinya juga tinggi. Barisan bilangan yang memiliki entropi rendah kemungkinan besar tidak random.

b. Uji *chi-square*: pengujian yang paling sering digunakan dalam menguji kerandoman sebuah barisan bilangan. Hasil dari uji *chi-square* direpresentasikan dalam sebuah angka persentase. Jika persentase di atas 99% atau di bawah 1%, bisa dipastikan barisan tidak random. Jika persentase di antara 95%-99% atau 1%-4%, barisan tersebut diduga tidak random. Semakin random sebuah barisan, hasil uji *chi-square*nya akan semakin mendekati angka 50%.

c. Rata-rata (uji frekuensi). Jika pada barisan random diambil setiap subbarisan n digit, rata-rata dari subbarisan-subbarisan tersebut akan mendekati bilangan tertentu. Contohnya untuk n=1 rata-ratanya mendekati 0,5 (frekuensi kemunculan 0 = frekuensi kemunculan 1). Untuk n=4 rata-ratanya mendekati 7,5 (karena setiap bilangan dari 0 sampai 15 memiliki probabilitas kemunculan yang sama, jumlahkan semuanya lalu bagi 16 akan didapat angka 7,5).

d. Uji π Monte Carlo. Diberikan persegi dengan sisi tertentu dan sebuah lingkaran di tengahnya yang diameternya sama dengan sisi persegi. Barisan random dipecah menjadi blok-blok, setiap blok dipecah 2 menjadi koordinat x dan y. Koordinat (x, y) tersebut kemudian diuji apakah berada di dalam atau di luar lingkaran. Dari hasil pengujian tersebut kemudian dapat dihitung perkiraan nilai π. Semakin random barisan bilangan, semakin dekat hasil perhitungan ke nilai π yang sebenarnya.

e. SCC (*Serial Correlation Coefficient*). Bilangan ini menyatakan kebergantungan sebuah bilangan dalam barisan terhadap bilangan-bilangan sebelumnya. Semakin random sebuah barisan, nilainya akan mendekati 0.

Beberapa metode pengujian lain yang termasuk uji kerandoman secara statistik adalah:

a. Uji poker: memeriksa subbarisan 5 digit seperti pada permainan poker

b. Uji jarak: memeriksa jarak antara bit 0, misalnya 00 berjarak 0, 010 berjarak 1, 01110 berjarak 3, dst.

Selain itu ada sebuah kumpulan pengujian yang dibuat oleh George Marsaglia dan diberi nama uji Diehard. Uji Diehard terdiri dari 12 buah pengujian statistik sebagai berikut:

a. *Birthday spacings*: pilih titik-titik random dalam interval yang sangat besar. Jarak antartitik seharusnya berdistribusi Poisson.

b. *Overlapping permutations*: seperti uji serial namun menggunakan subbarisan 5 digit, bukan 2 digit. Frekuensi kemunculan semua kemungkinan bilangan (00000, 00001, 00010, dst) seharusnya hampir sama.

c. *Ranks of matrices*: bentuk sebuah matriks biner dari beberapa bit random, dan tentukan *rank* dari matriks tersebut. Hitung distribusi *rank* yang dihasilkan.

d. *Monkey tests*: ambil sejumlah subbarisan bit sebagai sebuah kata, hitung jumlah kata yang tidak muncul dalam barisan. Jumlah tersebut seharusnya mengikuti distribusi tertentu.

e. *Count the 1s*: hitung jumlah kemunculan bit 1, konversikan ke sebuah angka. Lalu hitung jumlah kemunculan 5 buah angka.

f. *Parking lot test*: pada sebuah kotak berukuran 100 x 100, taruh lingkaran-lingkaran berjari-jari 1 secara random sehingga lingkaran-lingkaran tersebut tidak saling beririsan. Setelah 12.000 kali percobaan, jumlah lingkaran yang berhasil ditaruh seharusnya berdistribusi normal.

g. *Minimum distance test*: pada sebuah kotak berukuran 10.000 x 10.000, taruh 8.000 titik random, lalu cari jarak terdekat antara 2 titik. Kuadrat jarak tersebut seharusnya berdistribusi eksponensial dengan rata-rata tertentu.

h. *Random spheres test*: pada sebuah kubus dengan panjang sisi 1.000, taruh 4.000 titik random, lalu buat sebuah bola yang berpusat di tiap titik dan jari-jarinya adalah jarak terdekat titik tersebut ke titik lain. Volume bola terkecil seharusnya berdistribusi eksponensial.

i. *The squeeze test*: pangkatkan bilangan 2^{31} dengan bilangan real random pada selang [0,1] sampai mencapai angka 1, ulangi sebanyak 100.000 kali. Jumlah bilangna real yang dibutuhkan seharusnya mengikuti distribusi tertentu.

j. *Overlapping sums test*: anggap barisan biner sebagai barisan bilangan real. Jumlahkan

setiap 100 bilangan real. Jumlah tersebut seharusnya berdistribusi normal

k. *Runs test*: anggap barisan biner sebagai barisan bilangan real. Hitung panjang subbarisan yang menaik dan menurun. Panjang tersebut seharusnya mengikuti distribusi tertentu.

l. *The craps test*: mainkan 200.000 kali permainan craps (semacam permainan dadu), hitung jumlah kemenangan dan jumlah lemparan dadu pada setiap babak. Perhitungan tersebut seharusnya mengikuti distribusi tertentu.

3. Metode Pembangkitan Bilangan *Pseudorandom*

Berbagai metode pembangkitan bilangan *pseudorandom* telah dikembangkan oleh para ahli matematika dan komputasi. Beberapa di antaranya yang akan diuji dalam makalah ini adalah sebagai berikut:

a. *Middle-square*. Metode ini digunakan untuk membangkitkan bilangan random sepanjang n digit. Untuk membangkitkan bilangan berikutnya, bilangan yang sekarang dikuadratkan, kemudian dari hasil kuadrat tersebut diambil n digit yang terletak di tengah. Kelemahan cara ini adalah jika mencapai bilangan 0, bilangan-bilangan berikutnya akan terus 0.

b. LCG (*Linear Congruential Generator*). Metode ini menggunakan persamaan kongruensial $x_n \quad ax_{n\ 1} \quad c \bmod n$, dengan mengambil nilai a, c, dan n konstan.

c. MWC (*Multiply With Carry)*. Metode ini merupakan pengembangan dari LCG, berbentuk persamaan $x_n \quad ax_{n\ 1} \quad c_{n\ 1} \bmod n$ di mana c_n merupakan carry yang diambil dari upper bit x_n, yaitu $c_n \quad \frac{ax_{n\ 1} \quad c_{n\ 1}}{n}$.

d. Mersenne *twister*, merupakan metode pembangkitan bilangan *pseudorandom* yang cepat dan bagus. Mersenne *twister* memiliki periode perulangan sebesar sebuah bilangan prima Mersenne. Tapi algoritma ini dalam bentuk aslinya kurang cocok untuk aplikasi kriptografi karena setelah mengamati sejumlah barisan bilangan tertentu, bilangan-bilangan berikutnya dapat ditentukan.

e. CSPRNG (*Cryptographically Secure Pseudorandom Number Generator*). Pembangkit bilangan yang masuk ke dalam kategori CSPRNG adalah pembangkit bilangan yang telah berhasil melalui uji kerandoman dan juga tahan terhadap serangan. Beberapa metode yang termasuk CSPRNG di antaranya pembangkit bilangan *pseudorandom* berbasis cipher blok dan berbasis hash. Pada percobaan nanti penulis menggunakan library CSPRNG yang disediakan oleh Microsoft .NET.

f. Fungsi rand() pada bahasa pemrograman C di GNU C Compiler.

4. Hasil Pengujian dan Analisis

Pengujian terhadap pembangkit bilangan *pseudorandom* dilakukan dengan 2 set pengujian. Set pertama adalah perhitungan parameter statistik yang meliputi entropi, *chi-square*, rata-rata, Monte Carlo, dan *Serial Correlation Coefficient*. Program yang digunakan untuk pengujian didapatkan penulis dari situs http://www.fourmilab.ch/random/. Set kedua menggunakan uji Diehard yang meliputi 12 uji seperti telah dijelaskan di atas. Program uji Diehard dapat diunduh di http://stat.fsu.edu/pub/diehard/.

Untuk melakukan pengujian, pertama-tama penulis menggunakan pembangkit bilangan *pseudorandom* yang akan diuji untuk menulis file biner sepanjang 11.468.000 byte. Panjang tersebut adalah standar panjang untuk uji Diehard. Khusus untuk metode *middle-square*, pengujian Diehard tidak dilakukan karena algoritma *middle-square* tidak dapat membangkitkan barisan sepanjang itu tanpa mengalami konvergensi ke angka 0.

Di bawah ini adalah hasil pengujian yang telah dilakukan penulis terhadap pembangkit-pembangkit bilangan *pseudorandom* yang telah disebutkan pada bab sebelumnya.

Tabel 1. Uji statistik

	middle-square	**LCG**	**MWC**	**Mersenne *twister***	**.NET CSPRNG**	**C rand()**
entropi (bit per byte)	7,998267	6,615474	7,999984	7,999983	7,999985	7,999995
***chi-square* (%)**	45,03	0,01	45,59	19,10	71,09	72,28
rata-rata	127,0417	95,2065	127,4794	127,4929	127,5139	127,4891
Monte-Carlo (% error)	0,09	13,97	0,05	0,01	0,01	0,01
SCC	0,001754	-0,138725	-0,000138	0,000227	0,000177	0,000042

Tabel 2. Uji Diehard

	LCG	MWC	Mersenne *twister*	CSPRNG	C rand()
Birthday spacings	FAIL	OK	OK	OK	FAIL
Overlapping permutations	OK	OK	OK	OK	OK
Ranks of matrices	FAIL	OK	OK	OK	OK
Monkey	FAIL	OK	OK	OK	OK
Count the 1s	FAIL	OK	OK	OK	OK
Parking lot	OK	OK	OK	OK	FAIL
Minimum distance	OK	OK	OK	OK	FAIL
Random spheres	OK	OK	OK	OK	OK
Squeeze	OK	OK	OK	OK	OK
Overlapping sums	OK	OK	OK	OK	OK
Runs	OK	OK	OK	OK	OK
Craps	OK	OK	OK	OK	OK

Dari hasil pengujian di atas, penulis menyimpulkan bahwa MWC, Mersenne *twister*, dan CSPRNG adalah metode pembangkit bilangan *pseudorandom* yang bagus, ketiganya memiliki nilai statistik yang tinggi dan lolos semua uji Diehard. Sedangkan metode LCG sebaiknya tidak dipakai lagi karena memiliki nilai statistik yang buruk dan tidak lolos pada beberapa uji Diehard. Hasil yang menarik adalah pengujian statistik pada metode *middle-square*, yang ternyata menghasilkan nilai yang sangat baik, hampir sama dengan MWC, Mersenne *twister*, dan CSPRNG. Namun sayangnya metode *middle-square* tidak dapat digunakan karena memiliki konvergensi ke angka 0, kecuali jika dibuat pembangkit bilangan *pseudorandom* yang menangani kasus tersebut. Yang terakhir, fungsi rand() pada GCC memiliki nilai statistik yang tinggi, namun tidak lolos uji Diehard. Karena itu untuk aplikasi kriptografi yang dikembangkan di atas GCC sebaiknya digunakan *library* pembangkit bilangan pseudorandom yang lebih baik.

5. Kesimpulan

Berbagai metode pembangkit bilangan *pseudorandom* yang telah dikembangkan perlu diuji terlebih dahulu untuk mengetahui keamanannya terutama jika dipakai dalam aplikasi kriptografi. Beberapa metode memiliki performa yang buruk dalam pengujian, seperti misalnya *Linear Congruential Generator*, sehingga tidak layak lagi dipakai dalam aplikasi kriptografi. Namun metode-metode yang relatif baru memiliki performa yang bagus. Meskipun demikian tidak semua metode yang lolos pengujian dapat dipakai begitu saja dalam aplikasi kriptografi, seperti misalnya metode Mersenne *twister* yang dapat diterka. Beberapa metode perlu dimodifikasi dahulu supaya aman digunakan. Dengan semakin bertambahnya metode pembangkit bilangan *pseudorandom* yang aman, keamanan aplikasi kriptografi juga akan semakin bertambah. Tapi itu semua kembali kepada ketelitian para pengembang aplikasi kriptografi untuk memilih pembangkit bilangan *pseudorandom* yang tepat.

Daftar Pustaka

Munir, Rinaldi. 2006. *Diktat Kuliah IF5054 Kriptografi*. Bandung: Informatika ITB.

Schinldler, Werner. 1999. *Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators*. Federal Office for Information Security.

Viega, John. 2003. *Practical Random Number Generation in Software*. Annual Computer Security Applications Conference, Proc. 19th Dec. 2003.

http://www.bedaux.net/mtrand/. *Freeware C++ Mersenne Twister pseudo-random number generator*, diakses Mei 2009.

http://www.cs.auckland.ac.nz/CDMTCS/chaitin/sciamer.html. *Randomness and Mathematical Proof*, diakses Mei 2009.

http://www.fourmilab.ch/random/. *Pseudorandom Sequence Test Program*, diakses Mei 2009.

http://stat.fsu.edu/pub/diehard/. *The Marsaglia Random Number CDROM Including the Diehard Battery of Tests*, diakses Mei 2009.

http://www.phy.duke.edu/~rgb/General/rand_rate.php. *DieHarder: A Random Number Test Suite*, diakses Mei 2009.