Studi dan Analisis Mengenai Pengujian Bilangan Acak: *Diehard Battery of Randomness Test*

Paul Gunawan Hariyanto (13504023)
Teknik Informatika ITB, Bandung 40132, e-mail: if14023@students.if.itb.ac.id

Abstract *– Konsep pembangkitan bilangan acak banyak sekali ditemukan dalam teori di bidang kriptografi, karena inti dari pembangkitan bilangan acak adalah memberikan sebuah deretan angka, dimana sangat sulit untuk memprediksi bagaimana deretan angka tersebut. Contoh abstraknya adalah terdapat suatu metode enkripsi yang membutuhkan sebuah nilai tertentu sebagai masukan untuk pertama kalinya, dan nilai ini haruslah seacak mungkin supaya kemanan enkripsi ini tidak dapat diserang oleh pihak ketiga.*

Salah satu cara untuk menguji kekuatan bilangan acak adalah dengan menganalisis pola dari persebaran kumpulan bilangan acak yang dihasilkan. Semakin tersebar polanya, maka semakin acak deretan tersebut. Profesor George Marsaglia melakukan riset dan menemukan terdapat dua belas buah pengujian terpisah yang dapat diterapkan pada suatu algoritma pembangkit bilangan acak, untuk mengukur kekuatan algoritma tersebut. Seluruh pengujian tersebut dinamakan dengan 'Diehard Battery of Randomness Test', atau 'Diehard Test'.

Kata Kunci*: pembangkitan bilangan acak, kriptografi, pengujian bilangan acak, Diehard Battery of Randomness Test, Diehard Test.*

1. PENDAHULUAN

Diehard Battery of Randomness Test, yang selanjutnya akan disebut dengan *Diehard Test*, dikembangkan oleh seorang profesor di Florida State University yang bernama George Marsaglia, melalui riset yang dibiayai oleh National Science Foundation [5]. *Diehard Test* dipublikasikan pada tahun 1995, yang mana merupakan sekumpulan tes statistik untuk menguji tingkat kualitas dari keacakan suatu deretan angka. Ada dua belas pengujian yang diterapkan, dan algoritma pembangkit bilangan acak yang berhasil lolos dari semua tes ini merupakan pembangkit bilangan acak yang aman secara kriptografi.

Pembangkit bilangan acak banyak terpisah, atau ada di hardware sendiri.

Berikut adalah dua belas pengujian tersebut, dan masing-masing akan dijelaskan pada bagian Analisis.

1. *Birthday spacing test*
2. *Overlapping permutations test*
3. *Rank of matrices test*
4. *Monkey tests*
5. *Count the 1s test*
6. *Parking lot test*
7. *Minimum distance test*
8. *Random sphere test*
9. *Squeeze test*
10. *Overlapping sums test*
11. *Runs test*
12. *The craps test*

2. ANALISIS

Diehard Test terdiri dari dua belas pengujian yang berbeda, yang terpisah satu dengan yang lain. Tiap-tiap pengujian akan menghasilkan sebuah nilai-p (*p-value)* dengan kisaran [0,1), yaitu $0 \leq p < 1$. Nilai-p didapat dari p=F(X), dimana nilai F adalah distribusi dari sample X. Pengujian dikatakan gagal apabila nilai-p dekat dengan 1, atau lebih besar dari 0,9999. Akan tetapi mungkin saja algoritma yang sebenarnya kuat di saat tertentu akan menghasilkan nilai-p yang mendekati 0,9999, karena nilai-p hanyalah sebuah nilai perkiraan secara asimtot. Maka, pengujian sebaiknya dilakukan berkali-kali untuk menghindari hal ini, dimana pengujian dikatakan gagal apabila jumlah nilai-p yang mendekati nilai 0,9999 ditemukan berkali-kali [6].

Keduabelas pengujian dalam Diehard Test adalah sebagai berikut [1]:

2.1. *Birthday Spacing Test*

Didasari oleh 'permasalahan ulang tahun' (*Birthday problem/paradox)*, sebuah permasalahan yang dikenal dalam dunia matematika, yang mengatakan "dapat dibuktikan secara matematika bahwa dalam 23 orang yang dipilih secara acak, akan ada 50% kemungkinan terdapat beberapa pasangan yang memiliki hari ulang tahun yang sama" [9].

Birthday spacing test adalah dengan memilih *m* hari ulang tahun pada rentang waktu *n* hari, kemudian mengurutkan jarak dari tiap-tiap ulang tahun, maka *j* yang merupakan nilai jarak yang muncul lebih dari sekali akan bernilai $m^3/4n$. Nilai ini adalah nilai rata-rata dari distribusi Poisson secara asimtot. Untuk pengujian, nilai *n* sebaiknya berjumlah besar, yaitu kira-kira bernilai minimal 2^{18}.

Sebagai contoh, apabila $m = 2^{10}$ dan $n = 2^{24}$, maka nilai rata-rata-nya harus berkisar di angka $2^{30}/2^{26} = 2^4 = 16$. Dengan mengambil nilai *sample* j sebanyak 200, dan menggunakan metode *chi-square* untuk

membandingkan distribusi nilainya, maka nilai-p akan dihasilkan.

2.2. *Overlapping Permutations Test*

Overlapping permutations test adalah dari sejumlah bilangan acak yang dianalisis, ambil *m* bilangan secara berurutan, maka *m*! pola yang dapat dibentuk dari *m* bilangan tadi haruslah memiliki kemungkinan yang sama untuk muncul secara statistik, dan sebisa mungkin tidak muncul lebih dari sekali (*overlapping*). Dan secara teori, nilai persebaran secara asimtot yang dihasilkan haruslah sebesar $(m!)^2$.

Sebagai contoh, apabila $m = 5$, maka akan terdapat 120 pola yang dianalisis. Tiap-tiap pola akan menjadi *state* yang diperiksa. Nilai persebaran yang dihasilkan haruslah berada di sekitar 120^2.

2.3. *Rank of Matrices Test*

Rank of matrices test adalah membentuk sebuah matrix biner dengan ukuran m x n ($m \leq n$), yang berisikan bit-bit dari bilangan acak yang diperlukan. Kemudian matrix ini ditentukan nilai *rank*-nya, dimana kisarannya adalah 0 hingga m. Semua nilai rank ini akan dikumpulkan hingga mencapai sejumlah nilai tertentu, kemudian dengan menggunakan *chi-square*, persebaran nilai *rank* tadi akan dihitung untuk menghasilkan nilai-p.

Berikut adalah variasi dari *rank of matrices test*:

2.3.1. Matriks 31 x 31

Matriks ini berukuran 31 x 31, dengan cara diambil 31 *bit* terkiri dari tiap bilangan acak yang terpilih. *Rank* yang bisa dibentuk berkisar antara 0 sampai 31, namun karena nilai *rank* yang lebih kecil dari 28 jarang ditemui, maka kesemuanya dimasukkan dalam kategori nilai *rank* 28. Jumlah matrix yang dikumpulkan adalah sekitar 40.000 matrix.

2.3.1. Matriks 32 x 32

Matriks ini berukuran 32 x 32, dengan cara diambil *bit*-nya untuk tiap bilangan acak yang terpilih. *Rank* yang bisa dibentuk berkisar antara 0 sampai 32, namun karena nilai *rank* yang lebih kecil dari 29 jarang ditemui, maka kesemuanya dimasukkan dalam kategori nilai *rank* 29. Jumlah matrix yang dikumpulkan adalah sekitar 40.000 matrix.

2.3.1. Matriks 31 x 31

Matriks ini berukuran 6 x 8, dengan cara dari tiap bilangan acak yang terpilih, diambil sebuah byte dari angka tersebut. *Byte* ini dipecah menjadi *bit* untuk menjadi komponen matriks. *Rank* yang bisa dibentuk berkisar antara 0 sampai 6, namun karena nilai *rank* yang lebih kecil dari 4 jarang ditemui, maka kesemuanya dimasukkan dalam kategori nilai *rank* 4. Jumlah matrix yang dikumpulkan adalah sekitar 100.000 matrix.

2.4. *Monkey Test*

Pengujian ini berdasarkan dari 'teorema monyet tak terbatas' (*infinite monkey theorem*), yang menyatakan bahwa seekor monyet yang menekan tombol pada *keyboard* komputer untuk waktu yang tak terbatas pada suatu saat akan menghasilkan tulisan tertentu, seperti sebuah karya novel tertentu.

Monkey test adalah dengan menghasilkan *m* kata yang terdiri dari *n*-huruf secara *overlapping*, maka hitunglah nilai *j* yang merupakan jumlah dari kata *n*-huruf tersebut yang tidak pernah muncul dalam deretan. Tiap huruf ditentukan dari x bit, sehingga banyaknya huruf yang dapat dihasilkan adalah 2^x. Jumlah ini haruslah terdistribusi normal dengan nilai rata-rata *r*, nilai standar deviasi *d*, sehingga nilai standar normal yang dihasilkan yaitu sebesar $(j - r)/d$.

Terdapat 4 buah test yang menjadi variasi dari *monkey test*, yaitu:

2.4.1 *The OPSO Test*

Pengujian *OPSO*, atau *Overlapping Pairs Sparse Occupancy*, adalah menghasilkan 2^{21} buah kata yang terdiri dari 2-huruf. Tiap huruf ditentukan dari 10 bit, sehingga terdapat 1024 huruf yang dapat dihasilkan. Dari jumlah ini, nilai yang seharusnya dihasilkan adalah $r = 141.909$, $d = 290$, maka nilai standar normalnya adalah $(j - 141.909) / 290$.

2.4.2 *The OQSO Test*

OQSO test, atau *Overlapping Quadruples Sparse Occupancy test*, melakukan pengujian yang mirip dengan *OPSO test*, namun kali ini kata yang dihasilkan terdiri dari 4-huruf, dan tiap huruf ditentukan dari 5 bit, sehingga terdapat 32 huruf yang dapat dihasilkan. Jumlah kata yang dihasilkan juga sebanyak 2^{21}. Nilai yang seharusnya dihasilkan adalah $r = 141.909$, $d = 295$, maka nilai standar normalnya adalah $(j - 141.909) / 295$.

2.4.3 *The DNA Test*

DNA test melakukan pengujian dengan menghasilkan 2^{21} kata, yang terdiri dari 10 huruf. Tiap hurufnya terdiri dari 2 bit, sehingga banyaknya huruf yang dapat dihasilkan adalah 4 buah, yaitu C, G, A, dan T; dimana juga merupakan komponen dari DNA manusia. Nilai yang seharusnya dihasilkan adalah $r = 141.909$, $d = 339$, maka nilai standar normalnya adalah $(j - 141.909) / 339$.

2.4.4 *The Bit Stream Test*

Bit stream test menghasilkan 2^{21} buah kata yang terdiri dari 20 huruf, dan tiap hurufnya terdiri dari 20 bit, sehingga banyaknya huruf yang dapat dihasilkan adalah 1.048.576. Nilai yang seharusnya dihasilkan adalah $r = 141.909$, $d = 428$, maka nilai standar normalnya adalah $(j - 141.909) / 428$.

2.5. *Count The 1's Test*

Count the 1's test mengambil 5 buah byte, baik dengan cara berurutan (menganggap deretan angka yang dianalisis sebagai *stream of byte*) maupun spesifik (mengambil sebuah byte dari tiap-tiap angka). Tiap *byte* ini merepresentasikan sebuah huruf, yang tergantung dari berapa banyak bit 1 yang terkandung dalam *byte* tersebut. Misal digunakan huruf A, B, C, D, dan E, maka untuk jumlah 0, 1, dan 2 akan menghasilkan huruf A, untuk jumlah 3 adalah huruf B, untuk jumlah 4 adalah huruf C, untuk jumlah 5 adalah huruf D, dan untuk huruf 6, 7, dan 8 adalah huruf E. Jumlah bit 1 sebanyak 0, 1, 2 dan 6, 7, 8 masing-masing dijadikan 1 karena probabilitas munculnya jumlah tersebut adalah kecil dibandingkan jumlah lainnya.

Terdapat total 3.125 pola kata yang dapat dibentuk. Andaikan tiap kata terdiri dari hanya 5 huruf tadi, maka dengan menghitung seberapa banyak jenis masing-masing kata yang dapat dibentuk, persebaran nilai-p akan dapat dihitung dengan metode *chi-square*.

2.6. *Parking Lot Test*

Parking lot test adalah melakukan *n* usaha untuk memarkir *c* buah mobil (untuk memudahkan persoalan, dapat juga digunakan helikopter sebagai pengganti mobil) ke dalam suatu tempat yang acak pada tempat parkir (*parking lot*) yang berukuran *m* x *m*. Jika mobil itu menabrak mobil lain yang sudah parkir, maka ambil tempat acak lain untuk memarkir mobil tersebut. Kemungkinan berhasil penempatan parkir ini akan berkurang seiring dengan banyaknya mobil yang telah parkir.

Setelah dilakukan *n* kali, akan didapat *k* buah mobil yang berhasil parkir. Kurva persebaran dari $k(n)$ seharusnya akan mendekati pembangkit bilangan acak yang sempurna. Namun karena pembangkit bilangan acak yang sempurna belum ditemukan, diambil sebuah karakter yang dianggap demikian melalui simulasi yang berulang-ulang. Jumlah usaha yang dilakukan (n) adalah 12.000, dan persebaran nilai k bernilai 3523 dan standar deviasi 21,9. Nilai standar normal yang dihasilkan sebesar (k - 3523) / 21,9.

2.7. *Minimum Distance Test*

Minimum distance test adalah dari 8.000 buah titik yang dipilih secara acak dalam kotak berukuran 10.000 x 10.000, carilah nilai *d* (jarak terpendek antara $(n^2-n)/2$ pasangan titik). Persebaran yang bagus akan menghasilkan nilai *d*, dimana d^2 akan terdistribusi secara exponen dengan suatu nilai rata-rata *r*. Nilai $1-\exp(-d^2/0{,}995)$, akan menghasilkan kisaran angka [0,1), dan nilai inilah yang merupakan nilai-p.

Berdasarkan simulasi, nilai r yang menjadi standar adalah 0,995. Maka, nilai-p yang dihasilkan adalah $1-\exp(-d^2/0{,}995)$.

2.8. *Random Sphere Test*

Random sphere test adalah dari 4.000 buah titik yang dipilih secara acak dari kubus berukuran 1.000 x 1.000 x 1.000, tempatkan sebuah bola (*sphere*) pada titik yang mana merupakan jarak terpendek dari titik sebelumnya. Volume terkecil yang dibentuk dari penempatan bola-bola ini akan terdistribusi secara exponen dengan suatu nilai rata-rata *r*. Nilai $1-\exp(-d^2/0{,}995)$, akan menghasilkan kisaran angka [0,1), dan nilai inilah yang merupakan nilai-p.

Berdasarkan simulasi, nilai r yang menjadi standar adalah 30. Maka, nilai-p yang dihasilkan adalah $1-\exp(-r^3/30)$.

2.9. *Squeeze Test*

Squeeze test adalah melakukan perkalian sebanyak *j* kali antara 2^{31} dengan bilangan acak *real* yang dihasilkan berkisar [0,1), guna mencapai 1. Hasil perkalian untuk tiap iterasi akan dibulatkan ke bawah. Jumlah j yang dibutuhkan kira-kira sebesar 100.000 kali, dan angka-angka yang didapatkan saat $j \leq 6$, $j = 7$, $j = 8$, ..., $j = 46$, $j = 47$, dan $j \geq 48$, akan dipakai dalam metode *chi-square* untuk mencari persebarannya, dan menghasilkan nilai-p.

2.10. *Overlapping Sums Test*

Overlapping sums test adalah mengambil dan menambahkan 100 buah bilangan acak - yang kemudian diubah menjadi bilangan *real* dengan kisaran [0,1) - secara *overlapping*, dimana hasil penambahan bilangan tersebut seharusnya akan terdistribusi secara normal dan mengikuti karakteristik tertentu. Persebaran ini akan menjadi suatu nilai-p, dan nilai ini akan menjadi perbandingan untuk nilai-p berikutnya.

2.11. *Runs Test*

Runs test adalah melakukan penghitungan dari *n* bilangan acak *real* dengan kisaran [0,1), kapan terdapat deretan bilangan yang menaik, dan kapan menurun. Pola naik atau turun ini seharusnya akan mengikuti suatu pola persebaran tertentu, dan pola ini akan menghasilkan nilai-p dengan metode chi-square.

2.12. *The Craps Test*

Craps merupakan suatu permainan yang melibatkan dua buah dadu, dimana dilakukan oleh seorang pemain atau lebih. Dimainkan dalam babak (*round*), dimana babak akan berakhir (dan akan dilakukan pergantian pemain) apabila jumlah dadu yang dilempar adalah 7, 11, 2, 3, atau 12. Didapatnya angka 7 dan 11 adalah menang; sedangkan angka 2, 3, dan 12 disebut dengan *craps*.

The Craps test adalah melakukan permainan *craps* sebanyak n (minimal 200.000 kali), kemudian dihitung jumlah kemenangan dan jumlah lemparan dadu yang dilakukan pada tiap permainan. Tiap-tiap dari angka ini seharusnya akan membentuk persebaran

pola yang mendekati pola tertentu. Angka dari dadu dalam permainan dihasilkan melalui angka acak dalam deretan diubah menjadi bilangan real [0,1), yang dikalikan dengan 6.

Nilai rata-rata yang seharusnya didapat adalah *np*, dengan nilai *variance* sebesar np (1-p), dimana p = 244/495.

3. HASIL DAN PEMBAHASAN

Berikut adalah salah satu contoh dari hasil Diehard Test yang dilakukan dalam menguji algoritma pembangkit bilangan acak pada aplikasi Stata, yaitu sebuah perangkat lunak untuk Data Analysis dan Statistical, atau Data Mining. Hasil berikut berupa kumpulan semua nilai-p yang dihasilkan [10].

Tabel 1. Contoh hasil *Diehard Test*

Birthday Spacing	Parking Test	Overlapping Sum	Runs Test
0.496600	0.753306	0.687281	0.698717
0.685700	0.205562	0.239832	0.628001
0.320627	0.323972	0.780777	0.304695
0.339908	0.907282	0.686973	0.254565
0.682586	0.261324	0.658549	
0.790704	0.006836	0.914380	
0.252479	0.853193	0.266535	
0.058541	0.642555	0.836885	
0.121231	0.276387	0.603915	
	0.518210	0.289208	

OPSO	OQSO	Matrice Test 6x8	Count 1's Bytes
0.6201	0.3330	0.492591	0.610413
0.8671	0.1551	0.140335	0.440286
0.1438	0.9847	0.552921	0.874250
0.8054	0.5960	0.363156	0.299979
0.8110	0.4177	0.635619	0.122604
0.1856	0.6297	0.848542	0.868545
0.3118	0.0042	0.150218	0.512628
0.3069	0.5023	0.648319	0.989226
0.1122	0.7868	0.483047	0.553270
0.8611	0.3823	0.552624	0.168868
0.5188	0.4296	0.864940	0.561769
0.9447	0.1028	0.025812	0.258610
0.4968	0.4216	0.432879	0.936716
0.4216	0.3404	0.850342	0.815773
0.3094	0.9997	0.774656	0.999685
0.5869	0.9853	0.060820	0.191903
0.2320	0.5360	0.830375	0.030241
0.4284	0.5023	0.268389	0.649521
0.4136	0.3784	0.410072	0.491447
0.5325	0.8411	0.964345	0.275806
0.8618	0.1625	0.995751	0.955717
0.6689	0.3733	0.791425	0.655291
0.4748	0.1264	0.886268	0.817905
	0.1358	0.006625	0.159428
	0.0493	0.706425	0.667430
	0.2957		
	0.1797		
	0.5602		

Minimum Distance	Matrice Test 31x31	Squeeze	Matrice Test 32x32
0.690770	0.322191	0.843286	0.329856

Craps Win	Craps Throws	Overlap Permut	Count 1's Stream
0.050885	0.116953	0.043421	0.034771
		0.381748	0.110043

Spheres	BitStream Test	DNA	
0.00667	0.12262	0.0985	
0.49696	0.85059	0.5126	
0.98401	0.49969	0.9932	
0.24728	0.22710	0.5837	
0.99245	0.13540	0.3150	
0.07170	0.17541	0.7091	
0.80567	0.13540	0.7131	
0.53561	0.15791	0.9574	
0.42710	0.96375	0.6235	
0.55683	0.70100	0.1902	
0.97005	0.20787	0.3139	
0.89231	0.08944	0.5906	
0.07843	0.51181	0.7121	
0.71144	0.90725	0.2204	
0.53868	0.37948	0.8003	
0.57628	0.65755	0.3129	
0.59247	0.47641	0.1831	
0.85392	0.74402	0.0761	
0.52476	0.03344	0.2475	
0.66157	0.58935	0.3646	
		0.2186	
		0.2569	
		0.9503	
		0.6235	
		0.1815	
		0.0237	
		0.7040	
		0.4305	
		0.2041	
		0.5466	
		0.8187	

Hasil dari tabel diatas terangkum sebagai berikut.

Tabel 2. Contoh hasil *Diehard Test*

Nilai-p	Persentase hasil	Persentase yang diharapkan
0.0 -- 0.1	10	10
0.1 -- 0.2	12	10
0.2 -- 0.3	10	10
0.3 -- 0.4	10	10
0.4 -- 0.5	9	10
0.5 -- 0.6	12	10
0.6 -- 0.7	11	10
0.7 -- 0.8	7	10
0.8 -- 0.9	11	10
0.9 -- 1.0	9	10

Terlihat pada tabel 2 bahwa persentase hasil terdistribusi merata pada setiap nilai, oleh karena itu algoritma pembangkit bilangan acak pada aplikasi Stata, lolos dari *Diehard Test*.

4. PENGUJIAN PEMBANGKIT BILANGAN ACAK LAINNYA

Selain Diehard Test, masih banyak jenis pengujian lain yang dapat diterapkan pada pembangkit bilangan acak. Berikut adalah daftar dari beberapa jenis pengujian tersebut [7].

4.1. *Bob Jenkins' RNG Tests*

Di [2] terdapat kode dalam C yang diklaim telah dibuat sebelum *Diehard Test*, yang bertujuan menghitung bagaimana tingkat persebaran frekuensi, dan celah yang dihasilkan dari algoritma pembangkit bilangan acak. Bob Jenkins membuat test ini untuk menguji ketahanan algoritma pembangkit bilangan acak yang dibuatnya sendiri, yang bernama ISAAC. Algoritma ISAAC ini sudah lolos dari pengujiannya sendiri, dan juga *Diehard Test* [3].

4.2. *ENT Pseudorandom Sequence Test Program*

Program ini akan mengaplikasikan sejumlah tes dari sekumpulan *byte* pada file masukan, dan melaporkan file tersebut. Tes ini meliputi penghitungan entropy (kepadatan *byte*), *chi-square test*, *aritmatic mean* (penjumlahan semua *byte*), dan *serial correlation coefficient* (menghitung korelasi antar *byte*) [4].

Program ini berguna untuk mengevaluasi pembangkit bilangan acak semu yang digunakan untuk enkripsi, aplikasi sampling dalam bidang statistika, algoritma compresi, maupun aplikasi lainnya.

4.2. *NIST 800-22 Test Suite*

Pengujian ini adalah kumpulan dari 16 buah tes yang dikembangkan untuk menguji keacakan dari deretan biner yang relatif panjang, yang diproduksi baik oleh suatu perangkat lunak maupun perangkat keras khusus. Semua jenis pengujian ini berfokus pada macam-macam variasi ketidak-acakan yang dapat muncul pada deretan *byte* tersebut [8].

5. KESIMPULAN

Diehard Test merupakan salah satu cara untuk menguji tingkat keacakan dari deretan bilangan acak. Pengujian yang dilakukan dalam *Diehard Test* dibuat melalui riset selama bertahun-tahun, sehingga algoritma pembangkit bilangan acak yang lolos dari *Diehard Test* dapat dikatakan aman secara kriptografi.

Diehard Test dapat digunakan dalam banyak hal. Salah satunya adalah dalam pembuatan aplikasi kriptografi yang serius, dimana keamanan menjadi prioritas paling tinggi, algoritma pembangkit bilangan acak yang digunakan haruslah berhasil lolos dari pengujian statistik seperti *Diehard Test.*

Contoh lainnya adalah dalam pengembangan algoritma pembangkit bilangan acak yang baru; apabila terdapat hasil dari salah satu pengujian pada *Diehard Test* yang menunjukkan performa yang kurang baik, maka algoritma tersebut dapat diketahui celah ketidak-amanannya dan dapat diperbaiki sebelum dipublikasikan.

.

DAFTAR REFERENSI

[1] B. Narasimhan, "JDiehard: An implementation of Diehard in Java", Workshop on Distributed Statistical Computing, Vienna, Austria, 2001.

[2] http://burtleburtle.net/bob/rand/testsfor.html

[3] http://burtleburtle.net/bob/rand/isaacafa.html

[4] http://ftp.fourmilab.ch/random/

[5] http://i.cs.hku.hk/~diehard/cdrom/, diakses pada tanggal 13 Januari 2008.

[6] http://random.com.hr/products/random/manual/index.html, diakses pada tanggal 13 Januari 2008.

[7] http://randomnumber.org/links.htm

[8] http://www.cs.fsu.edu/~mascagni

[9] http://www.damninteresting.com/?p=402

[10] http://www.stata.com/support/cert/diehard/, diakses pada tanggal 13 Januari 2008.