Pembangkitan dan Pengujian Bilangan Acak Berbasis Bilangan Irrational

Ismail Sunni 13508064
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
ismailsunni@yahoo.co.id

bilangan acak merupakan salah satu bagian penting dari dunia kriptografi. Penggunaan nya banyak sekali, mulai dari pembangkitna kunci, initial vector, dan masih banyak lagi.
Di makalah ini, penulis, mencoba untuk mengajukan suatu metode pembangkitan bilangan acak dengan memanfaatkan sifat ketidakperiodikan suatu bilangan irrational.
Di akhir pembangkitan, akan dilakukan pula uji tetrhadap kerandoman yang telah dibangkitkan.
Namun, hasil yang didapatkan tidak terlalu memuaskan.

***Index Terms*—irrational numbers, PRNG, Uji Random**

I. LATAR BELAKANG

Dalam usaha pengamanan sistem dengan metode kriptografi, ada beberapa metode yang tingkat keamanannya bergantung pada suatu pembangkitan nilai yang tidak terprediksi, atau dengan lain tingkat keacakannya. Beberapa metode yang mempergunakan keacakan sebagai bagian dari tekniknya.

1. Keystream pada onetime pad
2. Kunci rahasia pada algoritma DES
3. Bilangan prima p dan q pada RSA
4. Bilangan prima pada mekanisme tanda tangan digital RSA
5. Kunci privat pada DSA

Kesemua contoh di atas menggunakan suatu nilai acak dengan ukuran tertentu. Acak disini dalan artian acak secara probabilitas. Selain acak, biasanya juga menggunakan ukuran yang cukup besar, misalnya 512 bit. Ukuran yang besar dan keacakan ini membuat kriptanalis tidak akan berfikir untuk melakukan brute-force. Karena, akan memakan waktu yang sangat lama, bisa mencapai ratusan atau ribuan tahun bahkan dengan super komputer paling hebat.
Penggunaan bilangan acak tidak hanya di wilayah kriptografi saja. Wilaya lain misalnya Permainan Komputer, Indutri Judi (misal untuk mengkocok kartu), password-generator, dan masih banyak lainnya. Jadi, bilangan acak memegang peranan yang cukup penting.

Bilangan acak sendiri tidak serta merta turun dari langit, namun harus dibangkitkan. Ada banyak metode pembangkitan bilangan acak, beberapa diantaranya :

1. Linear congruential generator (LCG)
2. Blum Blum Shut (BBS)
3. Pembangkitan berbasis RSA
4. Henon Map
5. Arnold's cat map

II. DASAR TEORI

Ada beberapa dan definisi yang akan digunakan dalama makalah ini

Bilangan Irrasional

[D] *bilangan irrational* adalah bilangan yang tidak dapat direpresentasikan sebagai bentuk desimal berhingga atau sebagai pecahan $\frac{m}{n}$ dimana, $GCD(m,n) = 1;\ m,n \in Z$.
Seperti definisi di atas, bilangan seperti *pi* $\pi = 3{,}14\ldots$ termasuk bilangan irrasional karena memiliki banyak digit tak hingga yang tidak memiliki periode. Namun, bilangan seperti $0.3333\ldots$ tidak termasuk bilangan-bilangan irrational karena memiliki digit tak hingga yang periodi, selain bisa direpresentasikan ke dalam pecahan, yakni $\frac{1}{3}$.

Ke-acak-an (*Randomness*)

[D] *Randomness* atau keacakan adalah suatu hal yang dibangkitakan oleh suatu proses, yang hasilnya tak terprediksi (*unpredictable*) dan tidak dapat diproduksi ulang walau dengan kondisi yang sama (*cannot be subsequentiallya reliably reproduced*)
[D] *unpredictable* dalam kasus barisan bit, tidak dapat diprediksi dengan mudah secara komputasional bit acak apa yang akan muncul selanjutnya walaupun diberikan 3 hal berikut ini :
1. penjelasan yang lengkap mengenai algoritma pembangkitannya
2. perangkat keras untuk membangkitkan barisannya
3. semua bit yang telah dibangkitkan sebelumnya.

[D] *unrepeatable sequence* jika diberikan suatu kondisi awal yang sama, hasil pembangkitan barisan tidak sama.

Jenis-jenis pembangkit bilangan acak :

1. PRNG : Pseudo Random Number Generator (Pembangkit Bilangan Acak Semu)

   Seperti namanya, *pseudo*, PRNG tidak menghasilkan random yang sebenarnya seperti ketika melempar dadu atau koin. Pada dasarnya, PRNG dibangkitkan oleh formula matematika atau bisa juga dari sebuah table yang telah diisi sebelumnya. Walaupun tidak menghasilkan bilangan random yang sebenarnya, namun beberapa algoritma bilangan random mampu membangkitkan bilangan random yang terliaht seperti sangat acak.

   Dibalik ke-kurang-acak-annya dibanding TRNG, PRNG memiliki beberapa kelebihan, yakni :

   a. Mampu menghasilkan banyak bilangan acak dalam waktu relatif singkat
   b. Deterministik atau bisa membangkitkan lagi bilangan acak dengan memasukan seed yang sama

   Namun, ada juga kelemahanny, yakni memiliki periode. Semua PRNG memiliki periode, misal saja memiliki periode, seperti pada LCG.

   Namun, makin hari, keperiodikan ini makin tidak terlihat, dengan makin berkembangnya algoritma untuk PRNG.
2. TRNG : True Random Number Generator (Pemabangkit Bilangan Acak Sejati)

   Tidak seperti PRNG, TRNG menghasilkan bilangan acak yang benar-benar acak, layaknya kita melempar dadu. TRNG bisa dibangkitkan dari beberapa kejadian alam, seperti :

   a. Peluruhan bahan radioaktif, proses mekanik pada level quantum
   b. Noise atau gangguan pada suhu atmofir dan sejenisnya, seperti yang digunakan di Random.org
   c. Suara dari mikrofon atau video dari sebuah kamera

   Walaupun demikian, TRNG memiliki beberapa kekurangan, yakni tidak efisien, dan tidak memungkinkan kita untuk membangkitkan bilangan yang serupa.

   Tapi, ada kelebihan yang tidak dimiliki PRNG pada umumnya, yakni tidak adanya periode dalam TRNG. Hal ini sangat membantu untuk kasus-kasus yang membutuhkan bilangan yang benar-benar acak.

PRBG Pseudo Random Bit Generator

PRBG merupakan saudara dari PRNG. Serupa, kecuali dalam domainnya. PRNG memiliki domain number, atau bilanagan. Sedangkan PRBG lebih ke arah bit.

[D] PRBG – Pseudo Random Bit Generator adalaha sebuah algoritma deterministik yang menghasilkan barisan bit yang terlihat seperti rando. Input dari PRBG disebut seed, sedangkan hasil keluaran dari PRBG disebut *pseudorandom bit sequences* atau barisan bit acak semu.

III. PEMBANGKITAN BILANGAN ACAK

Ketidakperiodikan bilangan irratioanl

Seperti yang sudah diketahui sebelumnya, bahwa PRNG memiliki kelemahan pada aspek keperiodikan, yang merupakan kelebihan dari TRNG. Tapi, sebaliknya, TRNG memiliki kekurangan dibanding PRNG yakni, tidak bisa membangkitkan bilangan yang sama. Walaupun, hanya berguna pada beberapa kasus saja, seperti pembangkitan kunci untuk enkripsi *one-time pad*.

Di sini, kita bisa menggunakan sifat dari bilangan irrational yang tidak periodik, sebagai pembangkit bilangan acak. Sebelumnya, akan dibuktikan terlebih dahulu mengenai ketidakperiodikan ada bilangan irrational :

Bukti bilangan irrational tidak memiliki digit yang periodik :

Andaikan, Q bilangan irrational, dan periodik, misal mantissa-nya memiliki periode t. Misal, $mantissa\ Q = mQ$. Sehingga, untuk setiap blok digit sepanjang t, di mantissa Q, maka akan berulang di k.t digit berikutnya, dengan k asli. Misal, blok itu adalah $a_1 a_2 \dots a_t$, sehingga, $mQ = a_1 a_2 \dots a_t a_1 a_2 \dots a_t a_1 a_2 \dots a_t \dots$., sehingga $mQ \times 10^t = a_1 a_2 \dots a_t a_{t+1}, a_1 a_2 \dots a_t a_{t+1} a_1 a_2 \dots a_t a_{t+1} a_1 a_2 \dots a_t a_{t+1} \dots$

Dengan mengurangkan keduanya, diperoleh bahwa

$$(10^t - 1)mQ = a_1 a_2 \dots a_t a_{t+1}$$

$$mQ = \frac{a_1 a_2 \dots a_t a_{t+1}}{(10^t - 1)}$$

Andaikan, $\gcd\left((10^t - 1), a_1 a_2 \dots a_t a_{t+1}\right) = d$, sehingga $a_1 a_2 \dots a_t a_{t+1} = dm$ dan $(10^t - 1) = dn$, dan jelas $\gcd(m, n) = 1$ atau m dan n saling prima. Sehingga :

$$mQ = \frac{m}{n}$$

Kontradiksi, dengan definisi bilangan irrational di atas. Atau dengan kata lain, tidak ada periode dalam digit-digit bilangan irrational.

Metode Pembangkitan

Setelah membuktikan bahwa bilangan irrational tidak memiliki periode, sekarang akan disusun salah satu pemnangkit barisan bilangan acak atau pembangkit barisan bit acak dengan menggunakan bilangan irrational.

1. PRNG

   Ketika membangkitakn barisan bilangan acak semu dengan menggunakan bilangan irrational, kita cukup mengekspansi suatu bilangan irrational hingga suatu panjang yang diperlukan. Namun, dengan catatan, kita menganggap 1 digit di mantissa sebagai satu suku di bilangan irrational. Sehingga, yang dihasilkan hanya bilangan $0,1, \dots,9$. Jika ingin melakukan konversi ke

domain bilangan lain, bisa menggunakan beberapa trik sederhana. Namun, di makalah ini tidak menekankan pada hal itu. Contoh pembangkitan bilangan acak dari bilangan irrational, dalam kasus ini e atau bilangan euler sebanyak 1000 digit:

```
61803398874989484820458683436563811772030917
98057628621354486227052604628189024
24497072072041893911374847540880753868917521
26633862223536931793180060766726354
43338908659593958290563832266131992829026780
88067520876689250171169620703222104
32162695486262963136144381497587012203408058
87954454749246185695364864449241044
43207713449470495658467885098743394422125448
87706647809158846074998871240076521
70575179788341662562494075890697040002812104
27621771117778053153171410117046665
59914669798731761356006708748071013179523689
94275219484353056783002287856997829
77834784587822891109762500302696156170025046
64338243776486102838312683303724292
67526311653392473167111211588186385133162038
84005222165791286675294654906811317
15993432359734949850904094762132229810172610
70596116456299098162905552085247904
35240602017279974717534277759277862561943208
82750513121815628551222480939471234
14517022373580577278616008688382952304592647
87870178899219902707769038953219681
98615143780314997411069260886742962267575605
523172777520353613936...
```

2. PRBG

Ketika kita ingin membangkitkan bit acak semu dengan menggunakan bilangan irrational, kita perlu melakukan pembangkitan bilangan acaknya terlebih dahulu. Kemudian, melakukan transformasi sederhana. Namun, perlu diingat, transformasi ini haruslah memetakan secara sama $0,1,2,\ldots,9$ ke bit $0,1$ secara adil. Beberapa contoh sederhananya adalah :

a. Ganjil untuk bit 1, genap untuk bit 0 atau sebaliknya
b. 0-4 untuk bit 1, 5-9 untuk bit 0, dan sebaliknya
c. Untuk 0-7, ubah ke bentuk binernya, dan 8 untuk bit 0, 9 untuk bit 1. Atau sebaliknya.

Pada makalah ini, akan digunakan konversi ke tiga. Untuk *e* 1000 digit di atas, jika diubah ke bit 0,1 akan menjadi seperti berikut :

```
11010011111001111001011000100010010010101
11001110011110101110110111111111001101111
11100101111110100110101111011001000110101
10111010110110010011010010101010010011111
01111001111001001011111111111100010011111
01100000011110111011001111110110110110111
01111011010101011101111101111111111110001
10011111011011110110111011001001111110101
01101011101111110101010101110110111010111
01101111111001010101101110001101111011000
01111101100110101011111111011101001110111
10101010100111011101011011011000110101101
10111011111110110010011011001111101011101
11010011100000101001111101100100101100111
11001101001101010111011011111010001101001
10010011010010100100111001111111111001001
11001110100110111010101001101110010101011
11100111111001001010110101100100011111101
11011010011100111010010011001111001100111
11101000011111010110111101011111011111111
11001110011101101010111010100110001111011
01011011110100000100110101001011111010111
11111111111111111001011111011111111100101
11111010011011011010111110011011011111011
11111111110111101110001101110011111000011
11011111111101101111001100101111011011110
00100111011101011101110110010100111010111
10111110101111111011100111010010101110101
10011101111110101010011010110111011011110
11110010101010011010011110101001111111111
10100011010100110111011001111011111101010
01011011011110110110111111010111111101001
01110101111011110100101110101101011101011
01011011011000101101001111011101101000110
00100111110111111111001111111110111001010
11111111101110111111011010101110111001110
00010111101010111111010110111010010110111
10101010000111110011111011100110010111111
01010111111110100101111111101110110111001
00110001101011011011010010111011010011101
11100111100111011101011101111111101011011
10111101111001101101101110011111001111001
11111100110110110110000110111100101110101
10110111101111101110010110111111101111111
11110110011101111101111111110
```

konverter menggunakan program kecil sederhana seperti berikut ini :

```
for (int i = 0; i < s.Length; ++i)
            {

                int n =
Int32.Parse(s[i]+"");
                //Console.WriteLine(i+",
"+n+", "+s[i]);

                if (n < 8)
                {
                    temp =
Convert.ToString(n, 2);
                }
                else if (n == 8)
                {
                    temp = "0";
                }
                else
                {
                    temp = "1";
                }
                sb.Append(temp);
            }
```

IV. PENGUJIAN

Pada bagian, ini akan dilakukan pengujian terhadap keacakan bit-bit yang telah dibangkitkan di atas. Ada beberapa metode yang telah dikembangkan oleh ahli untuk menguji kerandoman suatu barisan, antara lain :

1. Entropi. entropi merupakan ukuran yang menunjukan kepadatan suatu informasi yang terkandung dalam sebuah barisan bit. Barisan bit random, cenderung memiliki kepadatan informasi yang tinggi. Dengan kepadatan informasi yang tinggi, maka makin tinggi pula nilai entropinya. Begitu pula sebaliknya, makin tidak acak suatu barisan, maka nilai entropinya makin kecil pula.
2. Chi-square test. Hasil uji chi square biasanya dinyatakan dalam angka presentase. Jika nilai dari uji chi square ini kurang dari 1% atau lebih dari 99%, bisa dikatakan barisan tersebut tidak random. Sedangkan jika memiliki nilai dalam rentang 1%-4% atau 95%-99% maka barisan tersebut dikatakan mungkin tidak random. Semakin dekat nilai uji chi-square dengan 50%, maka barisan itu makin random.
3. Uji frekuensi. Ketika suatu barisan acak diambil setiap sub barisan n digit, maka rata-rata dari subbarisan yang terbentuk tadi akan mendekati suatu nilai tertentu.
4. Uji Pi, Monte Carlo. Persegi dengan panhang sisi tertentu dan sebuah lingkaran berada di tengah-tengahnya, denagn diameter yang sama panjang dengan sisi persegi. Selanjutnya, barisan randm yang akan diuji dipecah-pecah menjadi beberapa blok, kemudian setiap blok dipecah 2 menjadi elemen posisis koordinat, absis dan oordinat. Kemudian, untuk setiap pasangan tadi, diuji, apakah berada di dalam lngkaran atau di luar lingkaran. Dari hasil tersebut, bisa diperkirakan berapa nilai pi. Semakin dekat dengan nilai pi, maka barisan tersebut makin random. Begitu pula sebaliknya.
5. Serial correlation coefficient – SCC. Nilai yang dihasilkan di sini merupakan representasi dari nilai ketergantungan bit yang muncul dari bit-bit sebelumnya. Semakin mendekati 0, maka barisan tersebut makin random.

Penulis telah melakukan uji terhadap bilangan pi yang telah dikonversi ke bit. Bilangan pi yang digunakan adalah 100.000 digit. Dan setelah diubah menjadi bit, menjadi 200.000an bit. Berikut hasil tes yang menggunakan tester yang saya ambil dari situs : http://www.fourmilab.ch/random/.

```
Entropy = 0.934036 bits per byte.

Optimum compression would reduce
the size
of this 200069 byte file by 88
percent.

Chi square distribution for 200069
samples is27714656.25, and randomly
would exceed this value 0.01
percent of the times.

Arithmetic mean value of data bytes
is 48.6500 (127.5 = random).
Monte Carlo value for Pi is
4.000000000 (error 127.5 percent).
Serial correlation coefficient is -
0.0100705 (totally uncorrelated =
0.0).
```

Maksud dari hasil di atas adalah sebagai berikut :

1. Entropi
   Hasil uji entropi menunjukan bahwa barisan yang saya coba bisa dikompresi sebesar 0.934036 bits per byte. Ini menunjukan hasil yang tidak terlalu bagus. Karena, kepadatan informasinya kurang.
2. Kompresi optimum
   Nilai yang dihasilkan adalah seberapa besar kompresi yang bisa dilakukan terhadap barisan ini. Dan, bisa dilihat, dengan kompresi yang makin besar, makin jelek pula. 88% merupakan angka yang tidak terlalu bagus. Karena, informasi yang terkandung bisa dikompresi hingga 88%.
3. Chi square distribution

Dengan memperoleh nilai 0.01%, bisa dikatakan barisan ini tidak random. Seperti yang telah dijelaskan sebelumnya.

4. Rataan aritmatika.
   Rataan aritmatika yang diperoleh barisan ini adalah 48.6500. masih jauh dari rataan suatu barisan random, yakni 127.5.
5. Monte Carlo Pi
   Nilai monte carlo yang diperoleh adalah 4. Dan ini tidak cukup bagus, karena relatif jauh dengan nilai pi yang sebenarnya.
6. SCC – Serial corelation Coefficient.
   Nilai yang diperoleh di sini adalah -0.0100705 yang cukup dekat dengan 0. Dari tes ini, cukup baik.

Berikut ini jika dibandingkan dengan hasil yang diperoleh uji serupa dengan materi uji berupa beberapa fungsi pembangkit lain :

	Rand() pada .NET	Rand pad C()
Entropi	7.999985	7.999995
Chi square	71.09	72.28
Rata-rata	127.5139	127.2891
Monte carlo	0.01	0.01
SCC	0.000177	0.000042

VII. KESIMPULAN

Berikut ini adalah kesimpulan yang bisa diambil dari makalah ini :

1. Pemabangkitan bilangan acak semu dengan menggunakan bilangan irrational bisa dilakukan
2. Secara matematis, tidak ada keperiodikan dalam barisan tersebut.
3. Pembangkitan dengan metode ini juga memiliki keunggulan di bidang pembangkitna ulang.
4. Hasil pengujian dengan 5 tes, menunjukan barisan yang dihasilkan tidak cukup random

DAFTAR PUSTAKA

[1] http://math.about.com/library/bli.htm
[2] http://jproc.ca/crypto/terms.html tanggal akses 22 Maret 2011
[3] http://www.rsa.com/rsalabs/node.asp?id=2174 tanggal akses 2 Maret 2011
[4] http://www.kremlinencrypt.com/algorithms.htm tanggal akses 2 Maret 2011
[5] http://www.rossbach.to/tech/crypto/IrratCrypto.htm tanggal akses 2 Maret 2011
[6] http://kryptosfan.wordpress.com/speculations/is-k4-enciphered-using-a-one-time-pad/ tanggal akses 2 Maret 2011
[7] Ghodosi, Houssei, 1995, Pseudorandom Sequences obtained from Expansions of Irrational Numbers, Department of Computer Science Center for Computer Security Research, University of Wollongong Australia,
[8] Menezes, A, 1996, Handbook of Applied Cryptography,
[9] Munir, Rinaldi, Ir.,M.T. 2007. Diktat Kuliah IF-3058Kriptografi. Bandung : Informatika ITB
[10] http://www.fourmilab.ch/random/.
[11] https://calomel.org/entropy_random_number_generators.html
[12] https://calomel.org/entropy_random_number_generators.html
[13] http://www.random.org/randomness/
[14]

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

Ismail Sunni 13508064