Analisis & Perbandingan Blum Blum Shub dan Inversive Congruential Generator Beserta Implementasinya

Andreas Parry Lietara ~ NIM 13506076
Jurusan Teknik Informatika ITB, Bandung, email : andreas.parry@gmail.com

Abstract – *RNG atau Random Number Generator memiliki peranan penting dalam bidang kriptografi. RNG seringkali digunakan utuk membangkitkan seed dari berbagai algoritma kriptografi yang ada, misalnya saja untuk membangkitkan seed yang akan digunakan dalam algoritma block cipher atau membangkitkan bilangan prima untuk algoritma RSA. Makalah ini akan membahas mengenai dua algoritma pembangkit bilangan acak, yaitu Blum Blum Shub dan Inversive Congruential Generator, baik itu cara algoritmanya, cara kerjanya serta perbandingan kedua algoritma tersebut.*

Kata Kunci*: Blum Blum Shub Generator, Inversive Congruential Generator.*

1. 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 dua buah algoritma pembangkit bilangan acak, yaitu algoritma Blum Blum Shub Generator (BBS) dan algoritma Inversive Congruential Generator (ICG). Di sini akan dibahas masing – masing algoritma pembangkit bilangan acak tersebut serta perbandingan dan implementasinya.

Baik algoritma Blum Blum Shub Generator maupun algoritma Congruential Generator sama – sama termasuk dalam *algorithmic RNG* dan sama – sama berbasiskan pada operasi arimatika, oleh karena itu keduanya termasuk mudah untuk dipelajari dan diimplementasikan bahkan bagi seseorang yang baru mempelajari kriptografi.

2. LANDASAN TEORI

2.1. Blum Blum Shub Generator

Blum Blum Shub Generator merupakan algoritma pembangkit bilangan yang cukup sederhana dan mangkus. BBS diperkenalkan pada tahun 1986 pada Lenore Blum, Manuel Blum, dan Michael Shub.
Berikut ini dibeirkan algoritma dari BBS :

1. Pilih dua bilangan prima p dan q, di mana p dan q keduanya kongruen 3 modulo 4. $p \equiv 3 \bmod 4$ dan $q \equiv 3 (\bmod 4)$.
2. Hasilkan bilangan bulat blum n dengan menghitung $n = p \times q$.
3. Pilih lagi sebuah bilangan acak s sebagai umpan, bilangan yang dipilih harus memenuhi kriteria:
   a. $2 \le s < n$.
   b. s dan n relatif prima.
4. Hitung nilai $x_0 = s^2 \bmod n$
5. Hasilkan bilangan bit acak dengan cara :
   a. Hitung $x_i = {x_{i-1}}^2 \bmod n$.
   b. Hasilkan $z_i =$ bit - bit yang diambil dari x_i. Bit yang diambil bisa merupakan LSB (*Least Significant Bit*) / hanya satu bit atau sebanyak j bit (j tidak melebihi $\log_2(\log_2 n)$).
6. Bilangan bit acak dapat digunakan langsung atau diformat dengan aturan tertentu, sedemikian hingga menjadi bilangan bulat.

2.2. Inversive Congruential Generator

Inversive Congruential Generator atau pembangkit bilangan acak kongruen invers merupakan sebuah pembangkit bilangan acak dengan bentuk :

$$X_n = \overline{(aX_{n-1} + b)} \bmod m$$

Dengan :
X_n = bilangan acak ke n.

X_{n-1} = bilangan acak sebelumnya, atau *seed* jika X_0.
a = faktor pengali.
b = faktor *increment*.
m = modulus.
$\overline{(aX_{n-1}+b)}$ berarti inversi modulus m dari $(aX_{n-1}+b)$.

2.3. Chi Square Test

Chi square test adalah pengetesan pembangkit bilangan acak dengan metoda statistik. Berikut ini adalah langkah – langkah pengetesan dengan metodah *chi square test* :

1. Bagilah hasil keluaran yang mungkin menjadi sejumlah interval yang berukuran sama.
2. Bangkitkan sejumlah besar bilangan acak.
3. Kelompokkan setiap hasil yang muncul ke dalam interval yang telah ditetapkan.
4. Hitung jumlah bilangan yang termasuk dalam tiap – tiap interval.
5. Hitung nilai *chi square* berdasarkan nilai – nilai tersebut dan nilai harapannya.

2.4. Chi Square Statistic

Dalam suatu percobaan, nilai harapan yang ditentukan seringkali tidak sesuai dengan hasil percobaan. Misalnya saja, dalam pelemparan sebuah dadu sebanyak 6 kali, kita mengharapkan muncul mata dadu 1,2,3,4,5 dan 6 masing – masing satu kali. Namun, bisa saja ada satu / lebih mata dadu yang muncul lebih dari satu kali atau tidak muncul sama sekali. Untuk menghitung penyimpangan yang terjadi dari nilai harapan itu, kita gunakan penghitungan *chi square*. Berikut ini adalah metoda penghitungan penyimpangan *chi square* :

$$X^2 = \sum_x ((f_{hx} - f_x)^2 / f_{hx}).$$

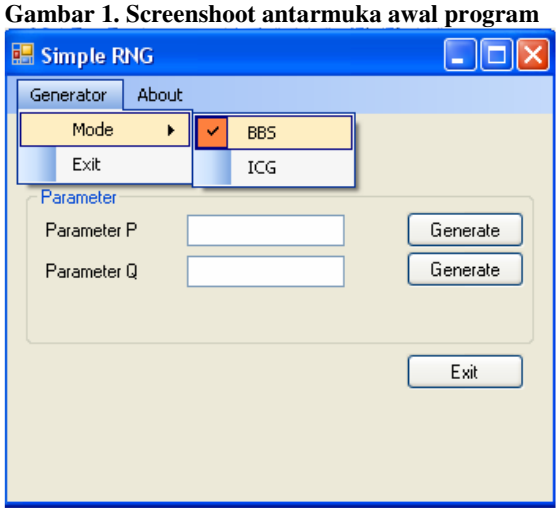
Dengan :
f_{hx} = frekuensi harapan dari interval x
f_x = frekuensi bilangan dari interval x

3. HASIL DAN PEMBAHASAN

3.1. Implementasi Algoritma

Pada kesempatan kali ini, penulis mencoba mengimplementasikan dua buah algoritma pembangkit bilangan acak yang telah dibahas di atas, yakni Blum – Blum Shub Generator dan Inversive Congruential Generator.
Pada program yang dibuat, pengguna dapat memilih salah satu dari dua mode algoritma yang diimplementasikan (BBS / ICG).

Gambar 1. Screenshoot antarmuka awal program



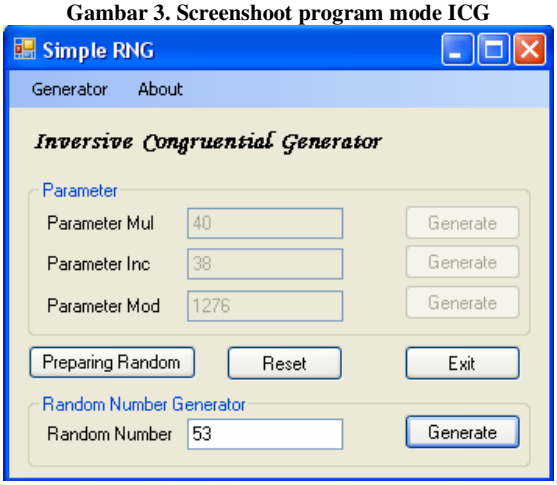
Pada mode BBS, diperlukan parameter masukan P dan Q, pengguna dapat memasukkan sendiri atau bisa juga meminta program membangkitkan parameter yang dimaksud. Berdasarkan parameter yang dimasukkan program akan membangun mesin *random* (*preparing random*), lalu membangkitkan bilangan acak. *Seed* yang dibutuhkan oleh algoritma BBS dibangkitkan secara acak dari dalam program.

Gambar 2. Screenshoot program mode BBS

Untuk algoritma Blum – Blum Shub ini, bit acak dihasilkan dengan mengambil adalah *Least Significant Bit* dari tiap pembangkitan bit sebanyak 4 kali. Keseluruhan bit itu digabungkan dan dikonversi menjadi sebuah bilangan (antara 0 - 15) yang kemudian digabungkan dengan 2 bilangan lain menjadi bilangan random (bilangan pertama ratusan, kedua puluhan dan ketiga satuan).
Berikut ini adalah contoh urutan keluaran dari program yang dijalankan (dengan algoritma BBS) :
1085, 1503, 103, 31, 1154, 250, 1615, 359, 724, 553, 1133, 655, 851, 529, 329, 646, 519, 845, 981, dst.
Sementara itu, pada mode ICG, program memerlukan parameter masukan faktor pengali (*parameter mul*), faktor *increment* (*parameter inc*) dan faktor modulus

(*parameter mod*). Setelah itu, prosesnya kurang lebih sama dengan algoritma BBS, program membangun mesin *random* dengan algoritma ICG (*preparing random*) dan membangkitkan bilangan acak. Sama seperti algoritma BBS, *seed* yang dibutuhkan algoritma ICG juga dibangkitkan secara acak dari dalam program.

Gambar 3. Screenshoot program mode ICG



Berikut adalah contoh urutan keluaran program yang dijalankan (dengan algoritma ICG) :
778, 1025, 0, 985, 809, 1024, 41, 962, 14, 26, 1023, 790, 869, 83, 994, 233, 5, 65, 37, 1019, 961, 185, 995, dst.

3.2. Perbandingan Algoritma

Pada kesempatan ini, penulis mencoba membandingkan kedua algoritma pembangkit bilangan acak yang telah dibahas di atas. Metoda perbandingan yang digunakan adalah dengan perhitungan statistik Chi Square. Dasar dari perbandingan adalah : sebuah pembangkit bilangan acak dikatakan baik apabila sebaran bilangan acak yang dihasilkan merata di seluruh rentang bilangan yang mungkin muncul.
Pada perbandingan yang penulis lakukan, penulis menggunakan parameter – parameter sebagai berikut :

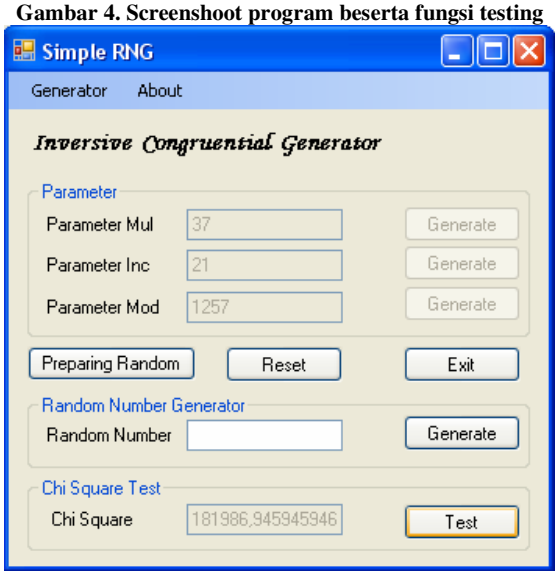
1. Rentang bilangan acak yang dihasilkan program (baik dengan algoritma BBS maupun algoritma ICG) adalah 0 s/d 1665.
2. Dalam pengujian, dibangkitkan bilangan acak sebanyak 16660 buah.
3. Bilangan acak yang dibangkitkan dikelompokkan dalam 10 interval / rentang.

Tabel 1. Tabel pembagian rentang percobaan perbandingan

Interval Bilangan	Kelompok
0 – 166	1
167 – 333	2
334 – 500	3
501 – 666	4
667 - 832	5
833 - 999	6
1000 - 1166	7
1167 - 1332	8
1333 - 1498	9
1499 - 1665	10

Seperti yang telah dijelaskan di dasar teori pada bab sebelumnya, tiap interval akan dihitung jumlah anggotanya. Setelah itu *Chi Square* akan dihitung dengan rumus yang ada. Pada percobaan ini, tiap algoritma (BBS dan ICG), masing – masing dihitung 10 kali nilai *Chi Square*-nya lalu dihitung rata – ratanya. Pada tiap penghitungan *Chi Square* pada tiap – tiap algoritma, digunakan parameter yang berbeda – beda.
Untuk menghitung nilai *Chi Square*, penulis telah memberikan fungsi tambahan terhadap program yang telah dibuat.

Gambar 4. Screenshoot program beserta fungsi testing



Berikut adalah hasil pengujian yang dilakukan :

Tabel 2. Hasil penghitungan *Chi Square* algoritma BBS

Percobaan ke -	Nilai *Chi Square*
1	138976,8198
2	474587,5586
3	135667,1502
4	143677,4925
5	614161,4925
6	154574,4895
7	482211,8709

8	206487,979
9	136523,6426
10	174707,4745
rata - rata	266157,597

Tabel 3. Hasil Penghitungan *Chi Square* algoritma ICG

Percobaan ke -	Nilai *Chi Square*
1	682602,3213
2	801145,3243
3	166419,006
4	616411,9309
5	441634,033
6	593520,2673
7	155444,8919
8	212920,7477
9	801745,2523
10	455623,8108
rata - rata	492746,7586

Pada hasil percobaan di atas, dapat dilihat bahwa hasil perhitungan *Chi Square* dari tiap algoritma sangat besar. Hal ini dapat dimaklumi mengingat jumlah data percobaan yang dilakukan sangat besar (16660 buah). Semakin banyak data, maka nilai penyimpangan makin besar. Misalnya saja, jika kita melempar sebuah dadu bermata 6 sebanyak 6 kali, ketika kita mengharapkan muncul mata dadu 6 tepat 1 kali, maka penyimpangan yang dapat terjadi bisa saja sebesar 1 (tidak muncul atau muncul dua kali) atau 2 (muncul 3 kali) dan lainnya. Namun, jika kita melempar mata dadu tersebut 600 kali dan kita mengharapkan muncul mata dadu 6 sebanyak 100 kali, bisa saja mata dadu 6 hanya muncul 50 kali, sehingga penyimpangannya sebesar 50. Ditambah lagi rumus dari *Chi Square* sendiri adalah pengkuadratan, sehingga dengan nilai penyimpangan yang besar, semakin besar pula nilai *Chi Square*. Walaupun demikian, metoda ini sudah cukup baik untuk membandingkan dua buah algoritma pembangkit bilangan acak.

Pada hasil di atas, dapat dilihat juga bahwa nilai *Chi Square* dari algoritma ICG jauh lebih besar daripada algoritma BBS. Hal ini menandakan penyebaran bilangan acak yang dibangkitkan oleh algoritma BBS jauh lebih merata ke seluruh kemungkinan nilanya dibandingkan algoritma ICG, dapat disimpulkan bahwa algoritma BBS pada dasarnya memiliki performa yang lebih baik dari algoritma ICG untuk menghasilkan bilangan acak.

Selain itu, dari data dapat dilihat bahwa perbedaan parameter masukan dari masing – masing algoritma, baik itu BBS maupun ICG, memberikan hasil *Chi Square* yang berbeda cukup jauh. Misalnya saja pada penghitungan nilai *Chi Square* algoritma BBS yang ke 3 dan ke 5, serta pada penghitungan nilai *Chi Square* algoritma ICG yang ke 3 dan ke 9, nilai *Chi Square* nya sangat berbeda jauh. Dapat disimpulkan bahwa untuk kedua algoritma ini (BBS dan ICG), pemilihan parameter pembangkit bilangan acak sangat berpengaruh terhadap performa pembangkit bilangan acak tersebut.

4. KESIMPULAN DAN SARAN

Berdasarkan hasil pembahasan di atas dapat disimpulkan :

1. Dilihat dari bilangan acak yang dihasilkan, baik algoritma ICG maupun BBS telah menghasilkan urutan bilangan acak yang tidak dapat diterka dan baik untuk digunakan.
2. Dilihat dari hasil perhitungan *Chi Square*-nya, bilangan acak yang dihasilkan algoritma BBS memiliki sebaran yang lebih merata daripada algoritma ICG, hal ini membuktikan performa algoritma BBS masih lebih baik dari algoritma ICG.
3. Parameter masukan dalam algoritma BBS dan ICG sangat berpengaruh terhadap bilangan acak yang dihasilkan.

Di luar pembahasan yang dilakukan, baik algoritma BBS dan algoritma ICG merupakan algoritma pembangkit bilangan acak yang berbasiskan operasi aritmatika yang mudah diterapkan. Oleh karena itu penggunaan kedua algoritma ini sangat dianjurkan bagi para pengguna awam pembangkit bilangan acak. Namun, dirasakan perlu juga untuk menggali lebih dalam tentang pengetahuan parameter – parameter kedua algoritma tersebut agar penggunaan keduanya dapat memberikan performa yang maksimal.

DAFTAR REFERENSI

[1] Slide kuliah Kriptografi IF 3058, "Pembangkit Bilangan Acak Semu (Bagian 1)", oleh Rinaldi Munir.

[2] http://www.cs.dartmouth.edu/~akapadia/project2 K. Waktu akses : 6 Mei 2009 Pukul 21.00

[3] Gonzalez, M. "Various Implementations of Blum Blum Shub Pseudo-Random Sequence Generator", Department of Electrical Engineering and Computer Science.

[4] http://www.mste.uiuc.edu/patel/chisquare/intro.html. Waktu akses : 20 Mei 2009 pkl 15.00

[5] http://www.ece.rice.edu/~bs/rng/test.html. Waktu akses : 18 Mei 2009 pkl 15.00