# Perbandingan Algoritma Pembangkit Bilangan Acak dengan Distribusi *Non Uniform*

Dininta Annisa - 13513066
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*dinintaannisa@gmail.com*

***Abstrak*—Dalam bidang kriptografi, bilangan acak biasa digunakan untuk pembangkitan kunci dan *initialization vector*. Terdapat berbagai metode untuk membangkitkan bilangan acak tersebut, tetapi tidak semua metode bisa digunakan dalam kriptografi, sebab dalam kriptografi, algoritma pembangkit bilangan acak harus benar-benar menjamin keacakan bilangan dan tahan terhadap serangan. Algoritma pembangkit bilangan acak yang populer digunakan dalam kriptografi diantaranya adalah BBS dan CSPRNG berbasis RSA. Keduanya dapat digolongkan dalam metode komputasi, sebab menggunakan rumus-rumus matematika yang menghasilkan bilangan acak dengan distribusi *uniform*. Makalah ini akan membandingkan serta membahas pengembangan algoritma metode komputasi tersebut agar dapat menghasilkan bilangan acak dengan distrbusi *non uniform*.**

***Kata Kunci*— Kriptografi, Pembangkit Bilangan Acak, Distribusi Probabilitas**

## I. PENDAHULUAN

Bilangan acak adalah bilangan yang kemunculannya tidak dapat diprediksi. Bilangan acak ini banyak digunakan dalam kehidupan sehari-hari, misalnya untuk mengacak "gerakan" komputer pada *game*, mengacak urutan lagu dalam pemutar musik (*shuffle playlist*), serta untuk permainan lotre dan mesin slot. Bilangan acak juga umum digunakan untuk simulasi kejadian tertentu serta *sampling* data dalam berbagai penelitian. Salah satu bidang yang banyak menggunakan bilangan acak adalah kriptografi. Kriptografi adalah ilmu yang menjaga kerahasiaan pesan dengan menyandikan pesan menjadi bentuk yang terlihat tidak bermakna. Dalam proses menyandikan pesan tersebut, beberapa algoritma membutuhkan sebuah nilai yang berperan sebagai *initialization vector*. *Initialization vector* ini dapat dibangkitkan dengan pembangkit bilangan acak. Selain sebagai inisialisasi, bilangan acak juga dapat digunakan sebagai parameter pembangkitan kunci yang akan digunakan untuk mengenkripsi dan mendekripsi pesan.

Pada mulanya, bilangan acak dibangkitkan secara manual, misalnya dengan melempar koin atau dadu. Seiring dengan berkembangnya teknologi, komputer mulai digunakan untuk membangkitkan bilangan acak. Berbagai algoritma pembangkit bilangan acak telah dikembangkan, tetapi hingga saat ini, belum ada algoritma yang benar-benar menghasilkan bilangan acak sempurna. Bilangan acak yang dihasilkan lebih tepat disebut sebagai bilangan acak semu (*pseudo random number*), sebab bilangan acak tersebut dapat dibangkitkan kembali dengan pola-pola tertentu. Meskipun begitu, beberapa algoritma berhasil membangkitkan bilangan acak yang aman karena kemunculannya sulit diprediksi. Algoritma yang cukup aman tersebut dapat digunakan dalam kriptografi dan disebut sebagai *cryptographically secure pseudorandom generator* (CSPRNG).

Ada berbagai metode atau pendekatan dalam membangun CSPRNG. Setiap pendekatan memiliki kelebihan serta kekurangannya masing-masing. Namun, setiap algoritma tersebut harus memenuhi dua syarat utama untuk dianggap sebagai CSPRNG, yaitu lolos uji keacakan statistik dan tahan terhadap serangan yang serius.

## II. DASAR TEORI

### *A. Pembangkit Bilangan Acak*

Pembangkit bilangan acak yang baik memiliki kriteria sebagai berikut.

1. Antar bilangan acak tidak saling bergantung satu sama lain.
2. Jika terjadi pengulangan, maka pengulangan tersebut terjadi setelah periode yang cukup lama.
3. Proses pembangkitan berjalan cepat dengan memori yang sedikit.

Secara umum, metode pembangkit bilangan acak dapat dibagi menjadi empat, yaitu sebagai berikut.

1. Metode Fisik
   Metode ini merupakan cara manual seperti mengocok kartu dan melempar koin atau dadu. Cara lainnya adalah dengan mengamati fenomena atomik atau subatomik yang sulit diprediksi, misalnya peluruhan radioaktif.
2. Metode Manusia
   Metode ini dilakukan dengan menggunakan manusia sebagai pembangkit bilangan acak. Namun, metode ini jarang digunakan karena

manusia memiliki sisi subjektif sehingga keacakan bilangan yang dihasilkan tidak begitu baik.

3. Metode Komputasi
   Metode ini memanfaatkan komputer untuk menghasilkan bilangan acak melalui rumus-rumus matematika seperti penjumlahan dan perkalian. Bilangan acak yang dihasilkan memiliki distribusi yang sama (*uniform*). Artinya, bilangan yang diambil sebagai bilangan acak memiliki peluang terambil yang sama besar. Metode ini menghasilkan bilangan acak yang berulang, tetapi sebagian algoritma dirancang sedemikian rupa hingga memiliki periode perulangan yang cukup besar.
4. Metode Distribusi Probabilitas
   Metode ini menghasilkan bilangan acak dari fungsi kepadatan probabilitas (*probability density function* atau PDF) supaya dapat menghasilkan bilangan acak dengan distribusi tertentu, misalnya eksponensial dan normal. Artinya, bilangan dalam rentang tertentu memiliki peluang terambil yang lebih besar dibanding rentang yang lain. Peluang terambilnya bilangan tersebut membentuk pola sesuai dengan distribusinya.

Berdasarkan kriteria pembangkit bilangan acak yang baik, metode fisik tidak memenuhi kriteria kecepatan, sementara metode manusia tidak dapat digunakan secara terus menerus. Oleh karena itu, metode yang paling umum digunakan saat ini adalah metode komputasi dan distribusi probabilitas.

### *B. Metode Komputasi*

Salah satu metode komputasi yang paling sederhana adalah dengan *Linear Congruential Generator* (LCG) yang memiliki bentuk sebagai berikut.

$$Z_i = (aZ_{i-1} + c) \bmod m$$

LCG memiliki waktu komputasi yang cukup singkat karena hanya membutuhkan sedikit operasi bit. Namun, cara ini tidak dapat digunakan dalam kriptografi karena urutan kemunculan bilangannya dapat diprediksi.

Algoritma CSPRNG yang umum digunakan saat ini adalah *Blum Blum Shub* (BBS) karena sederhana dan mangkus secara kompleksitas. BBS memiliki bentuk sebagai berikut.

$$x_{n+1} = x_n^2 \bmod M$$

*M* yang disebut sebagai bilangan bulat Blum merupakan hasil perkalian dua buah bilangan prima besar *p* dan *q*, dimana keduanya kongruen dengan 3 (mod 4). Keamanan algoritma BBS ini terletak pada sulitnya memfaktorkan nilai *M*, sebab hingga saat ini belum ada algoritma mangkus yang dapat memfaktorkan perkalian bilangan prima. Untuk setiap iterasi, bilangan acak yang dihasilkan dapat berupa *bit parity* dari $x_{n+1}$ atau *least significant bit* (LSB) dari $x_{n+1}$. Berikut merupakan contoh implementasi dari algoritma BBS dengan menggunakan bahasa Java.

```
public class bbs {
  private double p, q, M, seed, actual;
  public bbs(double p, double q, double
   seed) {
    this.p = p;
    this.q = q;
    this.M = p*q;
    this.seed = seed;
    this.actual = seed;
  }
  public double getrandom() {
    double r = actual*actual%M;
    actual = r;
    return r/M;
  }
}
```

Gambar 1 – Implementasi algoritma BBS

### *C. Metode Distribusi Probabilitas*

Algoritma yang termasuk dalam metode distribusi probabilitas, umumnya membutuhkan bilangan acak yang *uniform* untuk kemudian ditransformasikan agar sesuai dengan distribusi yang diinginkan. Jadi, pada dasarnya, metode ini merupakan pengembangan dari metode komputasi. Salah satu algoritma yang umum digunakan untuk metode ini adalah dengan metode invers. Cara ini dilakukan dengan menginverskan fungsi kepadatan probabilitas dari distribusi yang diinginkan. Langkah-langkah pembangkitan bilangan acak dengan distribusi *f(x)* adalah sebagai berikut.

1. Hitung fungsi distribusi kumulatif dari *f(x)*, yaitu *F(x)*.
2. Hitung invers dari fungsi *F(x)* yaitu $F^{-1}(x)$.
3. Bangkitkan sebuah bilangan random *u* yang memiliki distribusi *uniform*.
4. Bilangan acak yang dihasilkan adalah $x = F^{-1}(u)$.

Meskipun sederhana, metode invers tidak dapat selalu digunakan karena sulitnya menginverskan beberapa fungsi kepadatan probabilitas. Oleh karena itu, muncul metode-metode lainnya, misalnya metode Box-Muller. Metode ini dikhususkan untuk membangkitkan pasangan bilangan acak yang berdistribusi normal (*gaussian*). Untuk membangkitkan bilangan tersebut, mula-mula dibutuhkan dua bilangan acak berdistribusi *uniform* dalam selang 0 hingga 1, yaitu *x* dan *y*. Maka, pasangan bilangan yang dihasilkan adalah

$$x_0 = \sqrt{-2 \ln x} \cos(2\pi y)$$

$$x_1 = \sqrt{-2 \ln x} \sin(2\pi y)$$

Variasi metode Box-Muller yang lain menggunakan bilangan acak berdistribusi *uniform* yang berada di selang -1 hingga 1. Hal ini dilakukan supaya transformasi bilangan tersebut tidak perlu menggunakan fungsi sinus dan cosinus. Cara ini disebut juga sebagai metode Polar Box-Muller. Berikut merupakan contoh implementasi metode Polar Box-Muller dalam bahasa Java.

```
public double getRandomPolar() {
  double x1, x2, w;
  do {
    x1 = 2*(rand.getRand()/(RAND_MAX))-1;
    x2 = 2*(rand.getRand()/(RAND_MAX))-1;
    w = x1 * x1 + x2 * x2;
  } while ( w >= 1.0 );
  w = Math.sqrt((-2.0 * Math.log(w))/w);
  return x1 * w;
}
```

Gambar 2 – Implementasi metode Polar Box-Muller

Metode lainnya adalah menggunakan algoritma *acceptance-rejection*. Algoritma ini membangkitkan bilangan acak dengan cara melakukan *sampling* pada grafik kartesian bilangan acak *uniform* dan menjaga agar *sample* yang dihasilkan tetap berada di area di bawah grafik fungsi kepadatan probabilitasnya. Salah satu algoritma yang termasuk dalam metode ini adalah algoritma Ziggurat. Algoritma Ziggurat membagi area di bawah grafik fungsi kepadatan probabilitas menjadi *n* buah bagian dan masing-masing memiliki luas yang sama. Algoritma ini membutuhkan dua buah bilangan acak *uniform* dalam proses pembangkitan. Berikut adalah langkah-langkah pembangkitan bilangan acak dengan algoritma Ziggurat.

1. Pilih secara acak *i*, dengan $0 \leq i < n$
2. Hitung $x = u_0 x_i$
3. Jika $x < x_i+1$, maka bilangan acak yang dihasilkan adalah *x*
4. Jika $i = 0$, maka hitung *x* sedemikian hingga *x* berada di bawah grafik fungsi kepadatan probabilitasnya
5. Hitung $y = y_i + u_i (y_{i+1} - y_i)$
6. Jika $y < f(x)$, maka bilangan acak yang dihasilkan adalah *x*
7. Jika tidak, maka bangkitkan ulang bilangan acak $u_0$ dan $u_1$ lalu ulang kembali langkah pertama

## III. Eksperimen dan Hasil Perbandingan

Pada eksperimen ini, algoritma BBS dikembangkan dengan beberapa metode distribusi probabilitas agar menghasilkan bilangan acak dengan distribusi normal (*gaussian*) dan distribusi eksponensial. Pengembangan algoritma dilakukan dengan bahasa Java, sementara plot bilangan acak yang dihasilkan pada histogram dibuat dengan bahasa R. Untuk masing-masing distribusi, akan dibandingkan waktu yang dibutuhkan untuk membangkitkan bilangan serta histogram dari bilangan yang dihasilkan. Berikut adalah parameter yang digunakan untuk algoritma BBS.

| Parameter | Nilai |
|---|---|
| *p* | 15485867 |
| *q* | 23878409 |
| *seed* | 8353 |

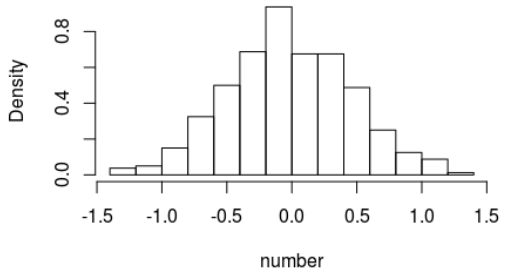
Tabel 1 – Parameter algoritma BBS

### A. Distribusi Normal

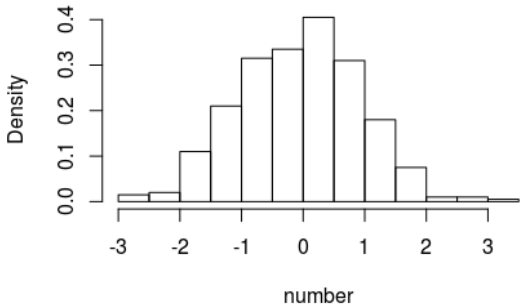
Distribusi normal adalah salah satu distribusi penting dalam statistika karena menggambarkan berbagai fenomena alam maupun hasil penelitian. Bentuk grafik dari distribusi ini menyerupai bentuk lonceng. Secara matematis, fungsi distribusi normal dapat ditulis sebagai berikut.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

Fungsi distribusi kumulatif dari distribusi normal adalah salah satu contoh fungsi yang sulit untuk diinverskan, sehingga metode invers tidak cocok untuk diterapkan pada distribusi ini. Oleh karena itu, untuk distribusi normal, diimplementasikan pembangkit bilangan acak dengan metode Box-Muller dan metode Polar Box-Muller. Grafik histogram dari fungsi kepadatan probabilitas kedua metode tersebut dapat dilihat pada Gambar 3 dan 4, sementara perbandingan waktunya dapat dilihat pada Tabel 2. Untuk pembuatan grafik, jumlah bilangan acak yang digunakan adalah sebanyak 400 buah.




Gambar 3 – Histogram bilangan acak dengan metode Box-Muller




Gambar 4 – Histogram bilangan acak dengan metode Polar Box-Muller

| Jumlah Bilangan Acak | Metode Box-Muller | Metode Polar Box-Muller |
|---|---|---|
| 100 | 12 ms | 7 ms |
| 1000 | 78 ms | 16 ms |
| 10000 | 166 ms | 73 ms |
| 100000 | 594 ms | 297 ms |
| 1000000 | 2500 ms | 1828 ms |

Tabel 2 – Waktu yang dibutuhkan untuk membangkitkan bilangan acak

Dari Gambar 3 dan 4, dapat dilihat bahwa distribusi bilangan acak yang dihasilkan sesuai dengan distribusi normal. Untuk performansi, waktu eksekusi metode Polar Box-Muller lebih cepat dibandingkan dengan metode Box-Muller. Perbedaan waktu ini semakin signifikan seiring dengan bertambahnya jumlah bilangan acak yang dibangkitkan. Hal ini disebabkan karena pada metode Box-Muller, banyak dilakukan pemanggilan fungsi matematika, yaitu logaritma natural, sinus, dan cosinus. Pada metode Polar Box-Muller, meskipun terdapat iterasi, tetapi iterasi tersebut hanya melakukan penjumlahan saja dan fungsi matematika yang dipanggil hanya logaritma natural yang dipanggil satu kali setelah selesai iterasi. Hal ini menyebabkan waktu eksekusi metode Polar Box-Muller lebih cepat dibandingkan dengan metode Box-Muller.

### *B. Distribusi Eksponensial*

Secara matematis, distribusi eksponensial dapat ditulis sebagai $f(x) = \lambda e^{-\lambda x}$. Jika dihitung fungsi distribusi kumulatifnya dengan cara pengintegralan, maka diperoleh

$$F(x) = 1 - e^{-\lambda x}$$

$$F^{-1}(x) = -\frac{1}{\lambda}\ln(1-x)$$

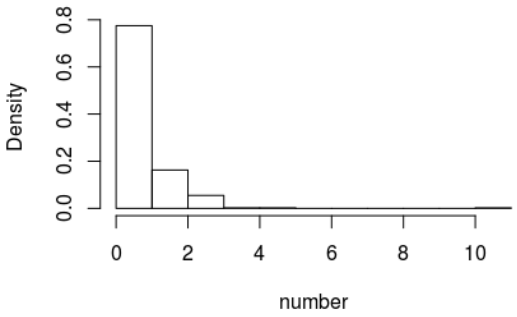
Jadi, untuk mendapatkan bilangan acak dengan distribusi eksponensial, dapat dilakukan dengan membangkitkan bilangan acak *uniform u*, dan menghitung nilai $F^{-1}(u)$.

Cara tersebut merupakan metode invers. Jika pembangkitan bilangan acak dilakukan dengan metode Box-Muller, maka fungsi pembangkit bilangan acak yang dihasilkan akan sama dengan $F^{-1}(u)$. Hal ini disebabkan karena pada dasarnya, metode Box-Muller memetakan dua buah bilangan acak *uniform* $u_1$ dan $u_2$ ke koordinat polar, yaitu R dan Θ. Keduanya kemudian dipetakan lagi menjadi dua bilangan acak berdistribusi tertentu, yaitu $x_0$ dan $x_1$. Hubungan keenam variabel tersebut adalah sebagai berikut.

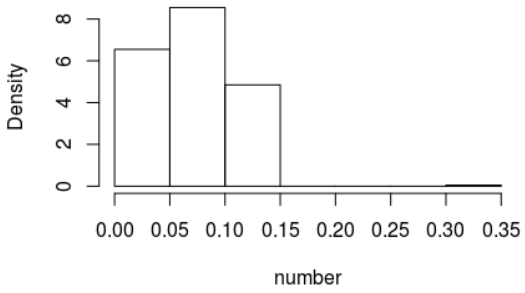
$$x_0 = R\cos\Theta = \sqrt{-2\ln u_1}\cos 2\pi u_2$$

$$x_1 = R\sin\Theta = \sqrt{-2\ln u_1}\sin 2\pi u_2$$

Dari kedua persamaan tersebut, diperoleh $R^2 = -2 \ln u_1$. Persamaan tersebut sama dengan $F^{-1}(x)$ dengan nilai $1/\lambda$ adalah 2. Oleh karena itu, dalam eksperimen ini, metode yang akan dibandingkan dengan metode invers adalah metode Ziggurat. Untuk eksperimen ini, grafik fungsi kepadatan dari distribusi eksponensial dibagi menjadi 256 bagian ($n = 256$). Grafik histogram dari fungsi kepadatan probabilitas kedua metode tersebut dapat dilihat pada Gambar 5 dan 6, sementara perbandingan waktunya dapat dilihat pada Tabel 3. Untuk pembuatan grafik, jumlah bilangan acak yang digunakan adalah sebanyak 400 buah.




Gambar 5 – Histogram bilangan acak dengan metode invers




Gambar 6– Histogram bilangan acak dengan metode Ziggurat

| Jumlah Bilangan Acak | Metode Invers | Metode Ziggurat |
|---|---|---|
| 100 | 16 ms | 15 ms |
| 1000 | 78 ms | 16 ms |
| 10000 | 126 ms | 62 ms |
| 100000 | 556 ms | 346 ms |
| 1000000 | 1939 ms | 1698 ms |

Tabel 3 – Waktu yang dibutuhkan untuk membangkitkan bilangan acak

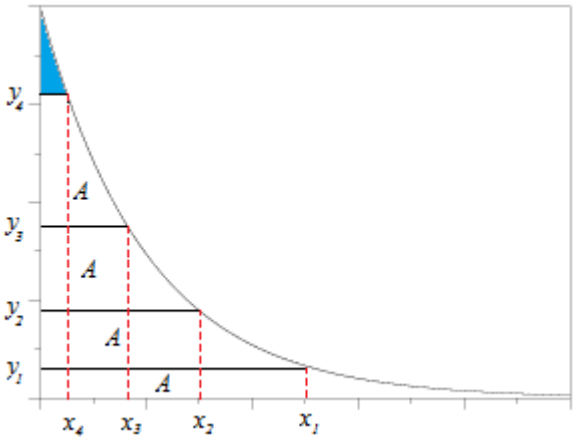
Dari Gambar 5, dapat dilihat bahwa metode invers berhasil membangkitkan bilangan acak dengan distribusi eksponensial. Untuk metode Ziggurat, terdapat perbedaan dengan grafik fungsi eksponensial yang sebenarnya, yaitu kerapatan untuk rentang nilai 0 hingga 0,005 lebih rendah dibandingkan kerapatan pada rentang 0,005 hingga 0,10. Hal ini disebabkan karena pilihan nilai $n$ dan $x_1$ yang kurang baik. Di awal proses pembangkitan, area di bawah kurva eksponensial dibagi menjadi $n$ bagian, yaitu *layer* 0 hingga *layer n-1*. Setiap bagian memiliki luas $A$, dan di tahap inisialisasi, perlu dihitung titik $x_i$ dan $y_i$ dengan $1 \leq i \leq n$. Nilai $x_1$ ditentukan secara acak sementara $y_1$ dapat dihitung dari $f(x_1)$ dimana $f(x)$ adalah fungsi distribusi eksponensial. Untuk nilai $x_i$ dan $y_i$ selanjutnya, dilakukan iterasi untuk menghitung dengan rumus berikut.

$$y_i = y_{i-1} + A / x_{i-1}$$

$$x_i = f^{-1}(y_i)$$

Di akhir iterasi, seharusnya diperoleh $y_n = f(0)$, tetapi hal ini tidak selalu terjadi, bergantung dari pilihan nilai $n$ dan $x_1$. Jika $y_n \neq f(0)$, maka area di bawah kurva eksponensial tidak seluruhnya masuk ke dalam pembagian $n$ *layer*. Ada area yang tidak termasuk sehingga tidak akan

ada bilangan acak yang dihasilkan dalam area tersebut. Ilustrasi kejadian ini dapat dilihat pada Gambar 7. Pada gambar tersebut, area di bawah kurva dibagi menjadi 4, tetapi area berwarna biru tidak termasuk ke dalam hasil pembagian.




Gambar 7 – Pembagian area di bawah kurva eksponensial

Untuk waktu eksekusi, metode invers membutuhkan waktu yang lebih lama dibandingkan dengan metode Ziggurat. Hal ini disebabkan karena pada metode invers, selalu dilakukan pemanggilan fungsi logaritma natural di setiap pembangkitan bilangan acak. Sementara itu, pada metode Ziggurat, pemanggilan fungsi logaritma natural hanya dilakukan jika *layer* yang digunakan adalah *layer* ke-0. Pemilihan *layer* ini dilakukan secara acak. Dalam eksperimen ini, terdapat 256 *layer*, sehingga kemungkinan pemanggilan fungsi logaritma natural hanya 1:256 saja. Meskipun begitu, perbedaan waktu antara metode invers dan Ziggurat tidak terlalu jauh, karena pada metode Ziggurat dilakukan pengecekan kondisi pada setiap pembangkitan bilangan acak.

## IV. Analisis

Dari keempat metode yang diimplementasikan, metode invers adalah metode yang paling sederhana serta mudah dipahami dan diimplementasikan. Namun, metode ini tidak dapat selalu dipakai karena tidak semua fungsi distribusi kumulatif dapat dihitung fungsi inversnya. Karena sederhana, metode ini memiliki waktu komputasi yang singkat. Namun, metode Zuggirat masih lebih baik dari sisi waktu komputasi. Selain itu, metode Zuggirat dapat dikembangkan untuk fungsi yang multi dimensional. Kekurangan dari metode ini adalah sulitnya menentukan parameter jumlah *layer* yang cocok agar seluruh area di bawah kurva terbagi rata. Selain itu, metode ini memerlukan waktu pada tahap inisialisasi untuk menghitung nilai $x_i$ dan $y_i$ untuk setiap $1 \leq i \leq n$. Meskipun begitu, waktu yang dibutuhkan untuk inisialisasi tidak banyak jika dibandingkan dengan pertambahan kebutuhan waktu eksekusi. Jadi, metode ini sangat cocok untuk membangkitkan bilangan acak dalam jumlah yang besar. Metode Zuggirat cocok untuk diimplementasikan pada fungsi yang monoton turun.

Dalam beberapa literatur, disebutkan bahwa metode invers lebih baik daripada metode Box-Muller. Hal ini disebabkan karena Box-Muller membutuhkan waktu komputasi yang besar, dan jika pembangkit bilangan acak *uniform* yang digunakan tidak memiliki tingkat keacakan yang baik, maka bilangan acak yang dihasilkan Box-Muller tidak akan menyebar dengan baik sesuai dengan distribusinya. Namun, metode invers ini tidak dapat digunakan untuk fungsi distribusi normal, sehingga untuk kasus tersebut dapat digunakan metode Box-Muller. Untuk mengurangi waktu komputasi, dapat digunakan metode Polar Box-Muller. Namun, hal ini tidak terlalu penting jika bilangan acak yang dibangkitkan hanya sedikit.

Untuk metode Box-Muller dan Polar Box-Muller, kedua metode ini lebih cocok untuk pembangkitan pasangan bilangan acak. Pada eksperimen ini, keduanya digunakan untuk membangkitkan satu bilangan saja per iterasi. Jadi, bilangan yang kedua dapat disimpan untuk kemudian dikembalikan pada pembangkitan bilangan acak selanjutnya.

## V. Kesimpulan

Terdapat berbagai cara untuk membangkitkan bilangan acak. Dalam bidang kriptografi, bilangan acak yang akan digunakan sebagai kunci atau *initialization vector* umumnya harus memiliki distribusi *uniform*. Hal ini dapat dibuktikan dengan algoritma CSPRNG yang saat ini populer seperti BBS dapat digolongkan ke dalam metode komputasi. Algoritma CSPRNG tersebut dapat dikembangkan untuk menghasilkan bilangan acak dengan distribusi tertentu, sebab beberapa bidang memiliki kebutuhan khusus akan bilangan acak, misalnya simulasi cuaca yang membutuhkan bilangan acak berdistribusi normal. Pengembangan algoritma ini juga dapat dilakukan dengan berbagai cara. Masing-masing cara memiliki kelebihan dan kekurangannya tersendiri. Oleh karena itu, perlu disesuaikan antara pilihan metode dengan fungsi distribusi yang diinginkan.

## Referensi


[1] http://tribudi.lecturer.pens.ac.id/LN_Simulasi/Bab_4_Bilangan_Acak_00.pdf, diakses pada tanggal 17 Desember 2016 pukul 15.10

[2] https://www.eg.bucknell.edu/~xmeng/Course/CS6337/Note/master/node50.html, diakses pada tanggal 17 Desember 2016 pukul 17.05

[3] http://elib.unikom.ac.id/files/disk1/471/jbptunikompp-gdl-rianilubis-23527-5-06rando-r.pdf, diakses pada tanggal 17 Desember 2016 pukul 20.45

[4] https://languageaholic.wordpress.com/tag/metode-pembangkit-bilangan-acak/, diakses pada tanggal 17 Desember 2016 pukul 21.15

[5] http://heliosphan.org/zigguratalgorithm/zigguratalgorithm.html, diakses pada tanggal 17 Desember 2016 pukul 22.30

[6] http://www.mathematik.uni-ulm.de/stochastik/lehre/ss06/markov/skript_engl/node29.html, diakses pada tanggal 18 Desember 2016 pukul 08.30

[7] http://www.coe.utah.edu/~cs5960-02/lectures/Lecture7.pdf, diakses pada tanggal 18 Desember 2016 pukul 08.45

[8] http://web.ics.purdue.edu/~hwan/IE680/Lectures/Chap08Slides.pdf, diakses pada tanggal 18 Desember 2016 pukul 09.30
[9] http://www.win.tue.nl/~marko/2WB05/lecture8.pdf, diakses pada tanggal 18 Desember 2016 pukul 10.00
[10] http://www.design.caltech.edu/erik/Misc/Gaussian.html, diakses pada tanggal 18 Desember 2016 pukul 12.00
[11] http://www.sciencedirect.com/science/article/pii/S0895717710005935, diakses pada tanggal 18 Desember 2016 pukul 12.15

## 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, 19 Desember 2016

Dininta Annisa
13513066