Pseudo-Random Number Generator

menggunakan waktu lokal, konsep efek kupu-kupu silang

Farid Firdaus (13511091)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha, 10, Bandung 40132, Indonesia
firdaus.farid22@gmail.com

***Abstrak*—Dalam makalah ini, saya mengajukan sebuah konsep baru dalam melakukan pembangkitan bilangan pseudo-random dengan memanfaatkan waktu lokal komputer, efek kupu-kupu, dan efek longsor. Waktu lokal komputer akan digunakan sebagai bilangan pembangkit awal untuk kemudian diproses menggunakan fungsi matematika. Nilai hasil pemrosesan secara berulang akan digunakan sebagai bilangan acak. Iterasi fungsi matematika diimplementasikan secara silang menggunakan dua buah nilai *seed* awal.**

Kata kunci—pseudo-random number generator, waktu lokal, efek kupu-kupu.

I. PENDAHULUAN

Pseudo-Random Number Generator (PRNG) merupakan algoritma yang digunakan untuk menghasilkan beberapa bilangan dengan acak sehinggabilangan selanjutnya tidak dapat diprediksi. Pemanfaatan algoritma ini banyak digunakan dalam bidang kriptografi, simulasi, dan permainan elektronik.

Algoritma ini disebut *pseudo* dikarenakan algoritma PRNG membutuhkan *seed* (nilai pembangkit), dimana nilai *seed* ini sangat menentukan hasil bilangan acak. Jika nilai *seed* yang digunakan adalah bilangan yang benar-benar acak, maka bilangan yang dihasilkan juga akan acak secara sempurna (tidak memiliki pola).

Dalam makalah ini, saya mengajukan sebuah nilai *seed* yang menggunakan waktu lokal dari mesin yang menjalankan PRNG. Nilai waktu lokal ini kemudian dimodifikasi sesuai fungsi matematika yang akan digunakan sebagai pembangkit bilangan acak. Fungsi matematika yang akan digunakan adalah fungsi matematika sederhana yang memenuhi efek kupu-kupu. Efek kupu-kupu ini bertujuan agar hasil yang dihasilkan menjadi sensitif pada perubahan nilai *seed* yang sangat kecil. Selain itu, efek ini akan diperkuat dengan menerapkan efek longsor dimana saat melakukan iterasi, dilakukan secara saling silang, sehingga diharapkan nilai bias yang dihasilkan dapat muncul secepat mungkin.

Agar nilai bias yang dihasilkan semakin besar saat melakukan iterasi, maka proses iterasi dilakukan melalui dua buah proses terpisah dengan menggunakan dua buah nilai *seed* dari waktu lokal komputer. Nilai yang akan digunakan untuk iterasi selanjutnya diproses secara berganti-gantian.

II. DASAR TEORI

A. *Pseudo-Random Number Generator*

PRNG merupakan algoritma pembangkit bilangan acak yang sangat bergantung pada nilai pembangkit awal (*seed*). Nilai *seed* yang sama akan menghasilkan bilangan acak yang sama. Untuk menutupi kekurangan ini, maka terkadang digunakan bilangan yang *truly random* sebagai *seed*. Salah satu contoh PRNG yang menggunakan bilangan *truly random* sebagai *seed* adalah pada situs random.org. Pada situs tersebut, *seed* yang digunakan adalah kuantitas pada alam (dapat berupa kuantitas partikel yang berada di udara sekitar sensor, dsb) yang dibaca menggunakan sensor yang sangat sensitif.

B. *Efek Kupu-Kupu*

Efek kupu-kupu merupakan efek dimana perubahan kecil pada kondisi awal suatu sistem non-linier akan menghasilkan perubahan yang cukup signifikan pada kondisi akhir. Efek ini pertama kali dikemukakan oleh Edward Lorenz, seorang meteoriologi dan matematikawan Amerika Serikat pada tahun 1972. Beliau mengajukan pertanyaan yaitu "Apakah kepakan sayap kupu-kupu di Brazil dapat menyebabkan badai topan di daerah Texas ?".

Dalam matematika, salah satu fungsi sederhana yang menghasilkan efek kupu-kupu jika terus dilakukan berulang-ulang adalah sebagai berikut :

$$x_{n+1} = 4x_n(1 - x_n), \quad 0 \le x_0 \le 1, \ldots \text{(II.1)}$$

Dengan menggunakan nilai $x_0 = 0.1$ sebagai nilai awal, maka dihasilkan nilai x_{1000} yang dihasilkan adalah 0.05280025168118729. Sedangkan untuk $x_0 = 0.100001$, maka nilai x_{1000} yang dihasilkan adalah 0.9999149157964466. Perubahan nilai awal sebesar 0.000001 menghasilkan nilai hasil akhir yang memiliki perbedaan sangat signifikan, yaitu berukuran hampir sama dengan nilai rentang domain hasil yaitu antara 0 dan 1. Efek ini muncul dalam bidang komputerisasi dikarenakan adanya keterbatasan prosesor dalam menangani bilangan *real,* sehingga terjadi pembulatan yang tidak diinginkan. Pembulatan inilah yang akan menimbulkan bias jika dilakukan secara berulang-ulang.

III. RANCANGAN ALGORITMA

Pada algoritma PRNG yang diajukan, akan digunakan waktu lokal pada mesin untuk kemudian dirubah kedalam bentuk dua buah nilai *seed*. Fungsi matematika yang akan digunakan dalam melakukan pembangkitan nilai acak , adalah fungsi II.1.

Pada fungsi tersebut, nilai rentang x adalah antara 0 dan 1, sehingga waktu lokal saat akan diformat kedalam bentuk *seed*, haruslah berada direntang nilai tersebut. Bentuk modifikasi yang diajukan adalah melakukan konkatenasi terhadap tahun, bulan, tanggal, dan jam, menit, deti, sehingga akan dihasilkan dua buah nilai *integer* yang cukup besar. Nilai tersebut kemudian dibagi dengan nilai 10^n . dimana nilai n adalah jumlah dogit dari bilangan tersebut. Contoh pengubahan nilai dari waktu lokal dapat dilihat sebagai berikut :

Waktu Lokal : 09 Mei 2015, 10:00:21

maka akan dihasilkan dua buah nilai *seed* sementara sebagai berikut :

$$Seed\ I \quad = 20150509$$

$$Seed\ II \quad = 100021$$

dikarenakan nilai rentang yang diperbolehkan dala, fungsi II.1 adalah nilai antara 0 dan 1, maka kedua nilai *seed* diatas akan dibagi dengan jumlah digit masing-masing sehingga menjadi

$$Seed\ I \quad = 0.20150509$$

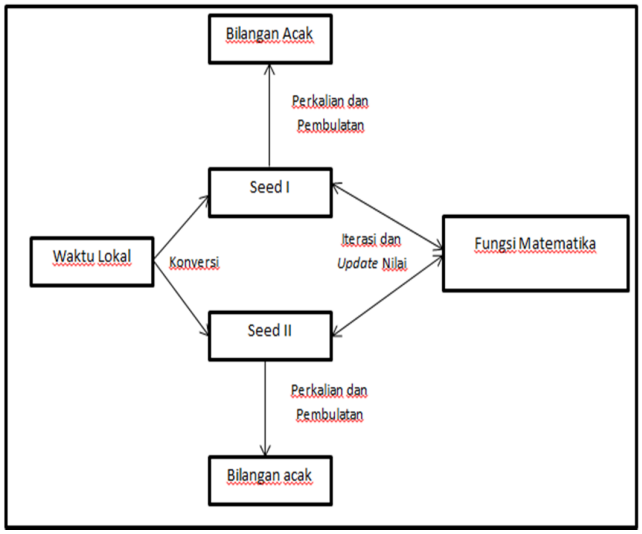
$$Seed\ II \quad = 0.100021$$

Kedua nilai tersebut kemudian dimasukkan kedalam fungsi II.1 untuk dilakukan iterasi sebanyak 1000 + n, dimana n adalah banyaknya bilangan acak yang ingin dibangkitkan. Proses iterasi dilakukan secara bergantian dengan cara saling silang seperti berikut

```
real seed1 ;
real seedII;

for(int n=0; n<1000; n++) {
    if(n=genap) {
        print(seedI);
        seedI  = 4*seed2*(1-seed2);
    } else {
        print(seedII);
        seedII = 4*seed1*(1-seed1);
    }
 }
```

Nilai yang dihasilkan dari setiap melakukan pencetakan nilai seedI atau seedII, akan selalu berada didalam rentang 0 dan 1, dan nilai yang dihasilkan tidak teratas jumlahnya. Oleh karena itu, agar nilai yang dihasilkan berupa bilangan *integer* atau bilangan *real* pada rentang tersebut perlu dikalikan dengan nilai maksimum yang diinginkan. Jika nilai yang dibutuhkan berupa bilangan *integer*, maka cukup dilakukan pembulatan pada hasil perkalian tersebut.

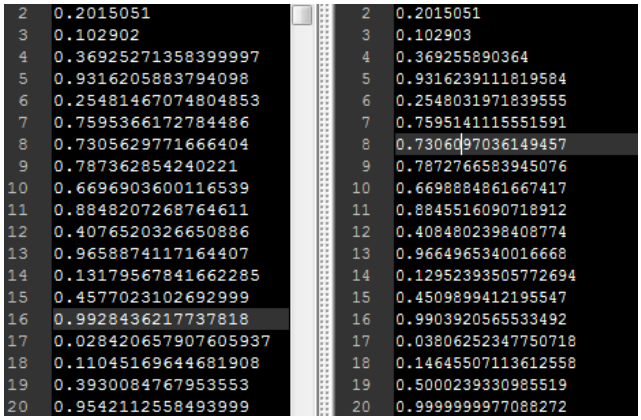


Gambar III.1. Diagram proses algoritma yang PRNG.

IV. ANALISIS

Dalam implementasi algoritma ini, pengguna tidak perlu memasukkan *seed* sehingga implementasi akan tampilan pengguna sederhana. *Seed* yang akan digunakan hanya dapat berulang apabila pengguna mengubah waktu pada sistem operasi yang digunakan. Kelemahan ini dapat diatasi dengan memindahkan waktu lokal sebagai bagian dari aplikasi. Dengan demikian, setiap akan dijalankan, aplikasi akan mengecek waktu sistem operasi terlebih dahulu, apakah lebih kecil dari rekor waktu sebelumnya. Jika iya, maka aplikasi akan mengabaikan waktu lokal sistem operasi, namun menggunakan iterasi waktu dari rekor waktu yang disimpan.

Dalam algoritma yang diajukan, dilakukan iterasi terhadap dua buah nilai *seed* secara bergantian. Kondisi demikian dilakukan agar tingkat sensivitas yang dihasilkan lebih baik, sehingga pengubahan kecil kondisi awal, dapat menjadikan hasil dengan dengan bias yang cukup signifikan. Berikut adalah hasil nilai yang dijalankan menggunakan algoritma yang diajukan.



2	0.2015051		2	0.2015051
3	0.102902		3	0.102903
4	0.36925271358399997		4	0.369255890364
5	0.9316205883794098		5	0.9316239111819584
6	0.25481467074804853		6	0.2548031971839555
7	0.7595366172784486		7	0.7595141115551591
8	0.7305629771666404		8	0.7306097036149457
9	0.787362854240221		9	0.7872766583945076
10	0.6696903600116539		10	0.6698884861667417
11	0.8848207268764611		11	0.8845516090718912
12	0.4076520326650886		12	0.4084802398408774
13	0.9658874117164407		13	0.9664965340016668
14	0.13179567841662285		14	0.12952393505772694
15	0.4577023102692999		15	0.4509899412195547
16	0.9928436217737818		16	0.9903920565533492
17	0.028420657907605937		17	0.03806252347750718
18	0.11045169644681908		18	0.14645507113612558
19	0.3930084767953553		19	0.5000239330985519
20	0.9542112558493999		20	0.9999999977088272

Gambar III.2 : Kondisi Awal : SeedI = 0.2015051, (Kiri) Seed II = 0.102902, (Kanan) Seed II = 0.102903.

Pada Gambar III.2, dengan mengubah nilai seed I sebesar, 0.000001, maka pada iterasi ke 19, telah terlihat bias yang cukup signifikan, yaitu sebesar 0.107. Dengan demikian, pada iterasi yang ke-1000, bias yang dihasilkan untuk perbedaan sebesar 0.000001 pada kondisi awal, akan sangat sulit di prediksi. Selain itu, dengan menggunakan iterasi silang, diharapkan terjadi efek longsor, dimana nilai awal tersebut kemudian menghasilkan nilai berantai yang semakin akhir, bias yang dihasilkan semakin besar.

DAFTAR PUSTAKA

[1] Chatur. Prashnant, Bhoge, Jayant. Avalanche Effect of AES Algorithm, IJCSIT, 2014

[2] Etienne, GHYS, The Butterfly Effect, CNRS-UMPA ENS, Lyon, 2012

[3] Lorenz, Edward, Computational Chaos - A Prelude to Computational Instability, MIT, Cambridge, 1988.