Fungsi Pembangkit Bilangan Acak Semu Dengan Memanfaatkan Konstanta Matematika dan Konsep Pemfaktoran

Sigit Aji Nugroho 13510021
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510021@studets.stei.itb.ac.id

***Abstract*—Makalah ini menjabarkan tentang algoritma fungsi pembangkit bilangan semu acak yang ringan secara komputasi namun berat sulit untuk dipecahkan. Ide yang digunakan ialah dengan menggunakan pemfaktoran bilangan bulat. Ide selanjutnya menggunakan nilai kontanta matematika.**

Kontanta matematika akan diambil nilai dibelakang komanya untuk dijadikan bialangan acak semu juga. Sementara nilai hasil pemfaktoran yang akan digunakan untuk memilih bilangan acak semu mana dari kontanta yang akan digunakan.

***Index Terms*—konstanta, pemfaktoran, *array of* puluhan, dan nilai faktor.**

I. PENDAHULUAN

Saat ini semakin banyak algoritma kriptografi yang ditemukan. Penemunya berlomba-lomba untuk membuat algoritma kriptografi yang paling aman, yaitu algoritma yang tidak bisa dipecahkan. Hanya saja, sampai saat ini membuat algoritma yang tidak bisa dipecahkan adalah sesuatu yang mustahil. Yang bisa dilakukan hanyalah membuat algoritma yang sangat sulit diecahkan, misalnya membutuhkan waktu hingga jutaan tahun komputasi agar bisa dipecahkan.

Sebenarnya konsep algoritma yang tidak bisa dipecahkan itu sudah ada, yaitu algoritma *one-time-pad*. Algoritma ini akan menghasilkan nilai-nilai yang benar-benar acak dan tidak akan bisa diulang kembali. Dengan prinsip ini tentu nilai keluaran tidak dapat dilacak kembali. Hanya saja algoritma semacam ini tidak akan berguna dalam dunia kriptografi. Suatu pesan yang telah dienkripsi harus bisa didekripsi.

Saat melakukan dekripsi, nilai-nilai yang digunakan tadi harus bisa dilacak kembali. Karena itulah muncul bilangan acak semu. Sebuah bilangan yang seolah-olah acak murni tapi sebenarnya memiliki pola keluaran tergantung algoritma yang digunakan di dalam fungsi pembangkitnya.

Agar benar-benar mendekati acak murni dan sulit diprediksi, tinggal membuat algoritma yang serumit mungkin di dalam fungsi pembangkit. Semisal menggunakan rumus:

$$N=N^6*2N^5*3N3*N*5 \quad (1)$$

Rumus 1 di atas cukup sederhana untuk diterapkan, tetapi sudah cukup rumit untuk dipecahkan. Agar lebih rumit lagi tinggal diperpanjang, diperbanyak pangkatnya, dan dikombinasikan operasinya misal dengan bagi, tambah, dan kurang.

Hanya saja algoritma semacam ini membutuhkan waktu komputasi yang cukup panjang. Selain itu juga membutuhkan penampung tipe bilangan yang sangat besar serta komputer dengan kemampuan komputasi yang besar. Terlebih dalam kebanyakan algoritma kriptografi, sering kali fungsi ini dipanggil berulang. Jika satu kali fungsi saja membutuhkan komputasi yang cukup lama, artinya proses enkripsi dan dekripsi akan berkali lipat lebih lama.

Dari sanalah muncul ide membuat algoritma yang komputasinya sederhana, tetapi sulit dipecahkan. Sudah ada beberapa ide yang telah digunakan dalam algoritma kriptografi modern. Beberapa diantaranya telah digunakan dalam algoritma kriptografi kunci publik, yaitu pemfaktoran, logaritma diskrit, dan *Elliptic Curve Discrete Logarithm Problem (ECDLP)* [1]. Beberapa ide ini dapat diadaptasi untuk digunakan dalam fungsi pembangkit bilangan acak semu.

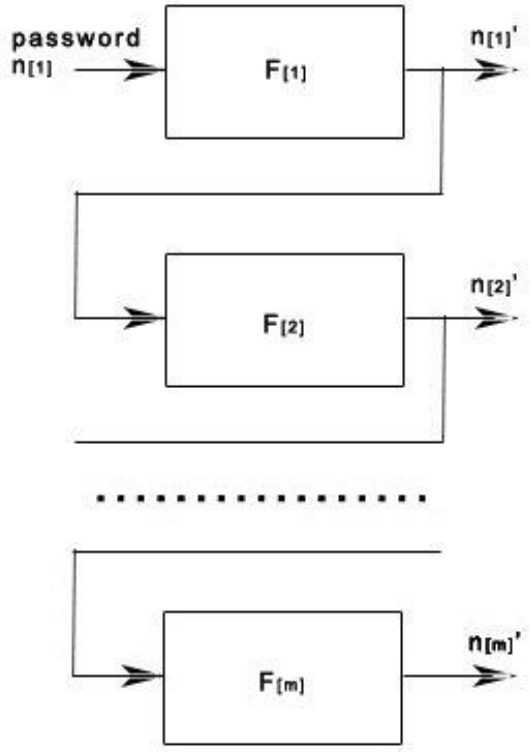
II. DASAR TEORI

Ada dua ide yang akan digunakan dalam fungsi pembagkit bilangan acak semu yang dibahas dalam makalah ini. Yang pertama adalah pemanfaatan konstanta matematika. Kedua adalah penggunaan konsep pemfaktoran.

A. Fungsi Pembangkit Bilangan Acak Semu

Fungsi pembangkit bilangan acak semua adalah sebuah fungsi yang menerima nilai masukan **n**. Nilai keluarannya adalah **n'** yang merupakan hasil akhir **n** setelah dimasukkan dalam algoritma.

Seperti yang telah dijelaskan sebelumnya, fungsi pembangkit ($\mathbf{F_{[i]}}$) ini digunakan secara berulang. Nilai masukan untuk pemanggilan fungsi yang pertama berasal inputan pengguna. Biasanya dalam bentuk *password*. Jika pasword yang digunakan berupa *string* maka harus diterjemahkan dahulu ke dalam bentuk bilangan agar bisa digunakan sebagai nilai masukan fungsi. Pada pemanggilan fungsi kedua dan seterusnya, nilai masukan untuk $\mathbf{F_{[i]}}$ adalah nilai keluaran dari fungsi sebelumnya $\mathbf{n_{[i-1]}'}$.



Gambar 1. Kerja Fungsi Pembangkit

B. Kontanta Matematika

Konstanta matematika yang dimaksud adalah bilangan-bilangan konstan yang sudah dikenal dunia. Sebenarnya tidak hanya bilangan-bilangan yang dikenal dalam cabang ilmu matematika, bisa juga kontanta di cabang ilmu eksak lainnya, semisal kimia dan fisika. Contohnya kecepatan cahaya (**c**) = **299.792.458** meter per detik atau yang lebih biasa digunakan $\mathbf{3x10^8}$ m/s.

Jika diamati urutan kemunculan angka dalam **c** seolah-olah membentuk bilangan acak. Awalnya **299** kemudian **792** lalu **458**. Kondisi inilah yang akan dimanfaatkan karena memunculkan nilai acak semu tapi dapat dilacak kembali karena sudah menjadi konstanta yang dipatenkan.

Tentunya tak semua konstanta dapat digunakan. Akan dipilih kontanta yang nilainya cukup panjang. Ini akan memberikan variasi nilai acak semu yang lebih banyak.

Tabel 1. Beberapa nilai konstanta matematika

π (pi)	3.14159265358979323846264 3...
e	2.71828182845904523536028 7...
φ (rasio emas)	1.61803398874989484820458 6

C. Konsep pemfaktoran

Dalam ilmu matematika, konsep ini akrab untuk memfaktorkan bentuk aljabar. Artinya mengubah bentuk penjumlahan suku-suku aljabar menjadi bentuk perkalian faktor-faktornya [2]. Berikut adalah contoh sederhananya :

Faktorkan X2+5X+6!

X2+5X+6 = (X+2)(X+3)

Sementara, memfaktorkan bilangan bulat artinya menyatakan suatu bilangan dalam bentuk faktor-faktornya. Faktor-faktor bilangan bulat adalah bilangan bulat yang membagi habis suatu bilangan bulat. Membagi habis artinya sisa pembagiannya adalah 0 (tidak tersisa) [2]. Berikut adalah contoh sederhananya:

Nyatakan 12 sebagai perkalian dua faktornya!

12 = 1 * 12
12 = 2 * 6
12 = 3 * 4

12 = -1 * -12
12 = -2 * -6
12 = -3 * -4

Maka faktor-faktor dari 12 adalah 1, 2, 3, 4, 6, 12 atau -1, -2, -3, -4, -6, -12

III. ANALISIS DAN IMPLEMENTASI

A. Jangkauan Nilai Acak

Nilai keluaran fungsi pembangkit bisa saja bebas sesuai dengan jangkauan tipe yang digunakan. Misalnya integer, nilai keluarannya bisa antara -2,147,483,648 hingga 2,147,483,647. Tapi juga bias disesuaikan dengan kebutuhan.

Pada kriptografi *watermarking*, nilai keluaran fungsi pembangkit digunakan untuk menentukan jarak titik selanjutnya yang akan disisipi pesan *watermark*. Tentunya nilainya tak mungkin negatif, karena jika dibiarkan negatif artinya suatu saat bisa jatuh di titik yang sebenarnya sudah disisipi pesan. Artinya nilainya selalu positif.

Selanjutnya, maksimal jarak yang digunakan juga harus disesuaikan. Misalnya yang hendak diberi watermark adalah video berukuran besar, sementara *watermarking* yang disisipkan hanya berupa gambar kecil, artinya titik

yang digunakan bisa disebar berjauhan. Jadi jangkauan nilai keluaran fungsi pembangkit juga bisa dibuat besar.

Sementara jika yang hendak diberikan *watermark* adalah gambar juga, tentu arak titik selanjutnya yang akan disisipi pesan tidak bisa terlalu jauh. Maka fungsi pembangkit bisa diatur agar keluarannya kecil, misalanya antara 1-100.

B. Pemanfaatan Konstanta Matematika

Kontanta matematika yang dipilih, akan digunakan sebagai pembangkit bilangan semu acak yang lebih sederhana. Misalnya yang digunakan adalah kontanta **e**. Seperti yang telah dijelaskan sebelumnya, angka-angka dibelakang koma bisa dipecah-pecah menjadi nilai-nilai yang seperti acak. Pemecahannya menjadi bilangan puluhan maupun ratusan atau mungkin ribuan akan ditentukan oleh algoritma yang akan digunakan. Agar lebih sederhana untuk dijelaskan, dalam makalah ini konstanta e akan dipecah menjadi *array of* bilangan puluhan.

e = 2.71828182845904523536028 7…

Nilai angka dibelakang koma dari kontanta **e** sebenarnya lebih panjang dari itu, seberapa banyak yang mau digunakan juga bisa disesuaikan.

Tabel 2. **Array of** puluhan dari konstanta **e**

Indeks	Nilai
0	71
1	82
2	81
3	82
4	84
5	59
6	04
7	52
8	35
9	36
10	02
11	87

C. Pemanfaatan Konsep Pemfaktoran

Suatu bilangan puluhan jika difaktorkan bisa memiliki banyak nilai. Padahal fungsi pembangkit hanya akan mengeluarkan satu nilai. Karenanya harus ditentukan nilai faktor mana yang akan digunakan.

Misal faktor dari 20 adalah 1, 2, 4, 5, 10, 20, nilai faktor yang diambil hanya yang (poitif). Bisa saja ditentukan adalah nilai yang akan diambil adalah nilai faktor yang nomor tiga terkecil, artinya yang diambil **4**. Tapi nilai ketiga terkecil ini bisa saja tidak dimiliki, misalnya jika nilai masukan adalah 2, karena faktor dari 2 adalah 1 dan 2.

Oleh karena itu, ditentukan bahwa yang akan diambil adalah bilangan terkecil kedua (misal dinamakan **s**). Mengapa tidak pertama, karena nilai faktor terkecil dari suatu bilangan selalu **1**. Mengapa kedua, karena setiap bilangan minimal memiliki 2 faktor, yaitu satu dan bilangan itu sendiri (bilangan prima).

Algoritma untuk menentukan nilai faktor terkecil kedua adalah

```
Iterasi i=2 sampai i=n do
        If n mod i = 0 then
                Return i
        Else
                i++
//end of iterasi
```

Algoritma di atas adalah cara paling sederhana untuk menentukan nilai faktor terkecil kedua. Hanya saja algoritma ini bisa tidak mangkus jika **n** yang masuk adalah bilangan prima. Artinya iterasi akan terus berlangsung hingga **i** = **n**. Untuk mengatasinya bisa dengan cara membuat *array of* bilangan prima antara jangkauan nilai keluaran.

Tabel 3. Tabel bilangan prima antara 1 - 100

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47
53	59	61	67	71
73	79	83	89	97

Sebelum **n** dimasukkan ke fungsi di atas, n terlebih dahulu dibandingkan dengan isi-isi di dalam **array of** bilangan prima. Jika ada yang sama, maka itu adalah nilai keluarannya, jika tidak baru dimasukkan ke dalam fungsi di atas. Maka kejadian paling buru ialah ketika **i** berhenti di **n/2**. Misalnya untuk **n** = 38, maka iterasi **i** berlangsung hingga nilainya 19 (**n**/2).

D. Algoritma Fungsi Pembangkit

Sebelumnya telah diperoleh *Array of* puluhan dari konstanta **e** dan nilai faktor terkecil kedua dari nilai masukan fungsi pembangkit . Kemudian tinggal menemukan nilai keluarannya. Salah satu cara termudah ialah dengan menjadikan **s** sebagai jarak indeks selanjutnya dari *array of* puluhan dari kontanta **e** yang akan diambil nilainya.

Misal nilai yang **s** pertama adalah 3, artinya nilai keluaran dari fungsi pembangkit yang pertama adalah *array* indeks ke-3, yaitu 81. Misal nilai **s** kedua adalah 5, maka nilai yang selanjutnya diambil adalah dari *array* dengan indeks ke 3 + 5 = 8, yaitu 52.

Suatu saat nilai ini akan terus bertambah, padahal panjang *array* ada batasnya. Karena itu, setelah dijumlahkan, nilai hasil penjumlahan domodulo dengan

panjang *array* terlebih dahulu. Hasilnya menjadi nomor indeks yang nilainya akan dikeluarkan. Selain itu juga disimpan menjadi penjumlah untuk fungsi pembangkit selanjutnya.

IV. PENGUJIAN

Algoritma ini belum memiliki batasan *password* yang jelas. Namun saat dimasukkan nilai integer yang cukup besar tetap bisa dijalankan. Hal ini karena nilai yang dibutuhkan dari *password* itu adalah nilai faktor terkecil kedua. Jika nilai yang dihasilkan masih terlalu besar juga, tetap aman digunakan karena saat digunakan untuk mencari indeks dari *array of* puluhan **e** sudah dimodulo terlebih dahulu dengan nilai panjang *array*.

V. KESIMPULAN

Algoritma ini akan lebih susah dipecahkan jika nilai jangkauan yang digunakan lebih besar. Dengan begitu nilai hasil fungsi pencari nilai faktor terkecil kedua akan lebih beragam nilainya. Hanya saja ini berbanding lurus dengan komputasi yang dilakukan. Artinya semakin besar jangkauan akan semakin berat komputasinya. Komputasi ini terjadi teruta amketika melakukan *searching* di dalam *array* dan saat melakukan iterasi **i**.

VII. ACKNOWLEDGMENT

Penulis ingin mengucapkan terima kasih yang paling besar kepada Allah SWT. Dialah yang senantiasa memberikan kesehatan dan kesempatan bagi penulis sehingga bisa menyusun makalah ini hingga selesai. Alhamdulillah.

Ucapan terima kasih selanjutnya ingin penulis sampaikan kepada dosen mata kuliah IF3058, Bapak Rinaldi Munir. Berkat bimbingan beliau penulis bisa memahami ilmu-ilmu dalam bidang Kriptografi. Semoga kedepannya, ilmu ini akan terus bermanfaat bagi penulis dan orang-orang di sekitarnya.

Ucapan terima kasih terakhir diberikan kepada teman-teman sekelas penulis. Khususnya ucapan terima kasih yang sangat besar kepada rekan sekelompok tugas. Atas kerja samanya, tugas besar dan tugas kecil yang diberikan pada mata kuliah ini bisa dikerjakan dengan baik.

REFERENCES

[1] Slide Kuliah IF3058 Kriptografi 2013, *Kriptografi Kunci-Publik*. Rinaldi Munir. Teknik Informatika STEI ITB.

[2] Sumber: http://deni11math.wordpress.com/2011/11/01/pemfaktoran-bentuk-kuadrat-ax2-bx-c/

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, 20 Mei 2013

Sigit Aji Nugroho
13510021