Pembangkitan Kunci Berantai Semi-*Random* Untuk Algoritma *One Time Pad*

Made Harta Dwijaksara [1)]

1) Program Studi Teknik Informatika, ITB, Bandung 40132, email: if14137@students.if.itb.ac.id

Abstraksi – *One time pad adalah algoritma yang sampai saat ini diakui memiliki tingkat kekuatan yang sangat tinggi, oleh karenanya akan sangat sulit untuk dipecahkan hasil chiperteks dari algoritma ini. Kekuatan ini terletak pada penggunaan kunci yang panjangnya sama dengan panjang teks yang akan dienkripsi, sehingga didapatkan hasil yang benar-benar acak. Namun penggunaan kunci yang sedemikian panjang ini menimbulkan masalah, karena akan dibutuhkan mekanisme khusus dalam penyimpanannya.*

Kata Kunci*:One time pad, chiperteks, enkripsi.*

1. PENDAHULUAN

One Time Pad telah diklaim sebagai satu-satunya algoritma kriptografi sempurna sehingga tidak dapat dipecahkan [3]. Sebagai algoritma yang menggunakan metode chipper subtitusi , algoritma ini mampu memetakan plainteks menjadi chiperteks dimana tidak akan ada lagi keterhubungan antara plaintext dan chipper text yang bisa dimanfaatkan oleh seorang kripanalis untuk memecahkan chipperteks tersebut. Aturan enkripsi yang dimanfaatkan pada algoritma ini persis sama dengan Vigenere Chiper . Pengiriman pesan menggunakan setiap karakter kunci untuk mengenkripsi suatu karakter plainteks. Proses enkripsi dan dekripsi dapat dinyatakan sebagai [2]:

$$c_i = (p_i \oplus k_i) \quad (1)$$

dan

$$p_i = (c_i \oplus k_i) \quad (2)$$

yang dalam hal ini,

ci = karakter chiperteks
pi = karakter plainteks
ki = karakter kunci

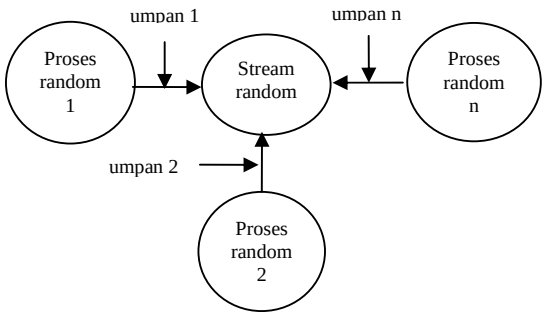
Kekuatan dari algoritma ini didapatkan karena panjang kunci yang dipergunakan untuk mengenkripsi setiap plainteks sama dengan panjang plainteks itu sendiri. Sehingga didapatkan subtitor yang acak untuk setiap karakter pada plainteks. Hal ini mengindikasikan bahwa tidak akan terdapat suatu pola tertentu yang menghubungkan antara chiperteks hasil dengan plainteks.

One time pad sebagai algoritma enkripsi yang dianggap sangat aman juga memiliki kelemahan yaitu kunci yang terlalu panjang akan sangat menyulitkan. Sehingga setiap kali akan melakukan proses enkripsi, terlebih dahulu kita harus membangkitkan kunci acak yang panjang sama dengan panjang pesan secara terpisah. Dengan demikian jelas tidak mungkin kita untuk mengingat kunci yang kita pergunakan, jadi haruslah ada mekanisme khusus untuk penyimpanan kunci. Dari solusi ini maka akan timbul masalah lagi yaitu akan seberapa amankah metode penyimpanan kunci itu? Jelas solusi ini tidaklah begitu bagus karena solusi itu sendiri menimbulkan masalah baru.

Permasalahan diatas akan dipecahkan melalui makalah ini. Panjang kunci yang sama dengan panjang pesan akan coba direduksi sehingga pada akhirnya kita tidak perlu lagi membangkitkan kunci random yang sedemikian panjangnya secara terspisah. Kita hanya akan perlu memasukan kunci seperti biasa (dengan panjang tertentu) dengan tidak akan mengurangi kekuatan dari algoritma ini. Metode yang dipergunakan adalah metode pembangkitan kunci berantai semi-random, yaitu pembangkitan kunci yang panjangnya sama dengan panjang plainteks dengan memanfaatkan masukan kunci yang lebih pendek. Kunci yang lebih pendek ini akan dipergunakan secara berantai guna membangkitkan kunci untuk one time pad.

2. KUNCI SEMI RANDOM BERANTAI

Seperti yang telah dijelaskan pada bab bagian pendahuluan diatas, bahwa untuk masalah pada algoritma *one time pad* ini ditawarkan solusi dengan melakukan pembangkitan kunci semi *random* berantai. Tujuan dari mekanisme ini adalah untuk memperpendek masukan kunci yang diperlukan namun dengan tidak menghilangkan unsur acak yang diberikan oleh kunci. Semi *random* disini dimaksudkan adalah ketika suatu kunci dibangkitkan akan memerlukan suatu umpan tertentu untuk membentuk *stream random*, dan ketika pada kesempatan yang berbeda akan dibangkitkan bilangan *random*, maka bilangan *random* dibangkitkan dengan umpan yang sama adalah suatu *stream* yang sama dengan *stream* yang dibangkitkan sebelumnya. Disamping itu pembangkitan bilangan random yang dilakukan menggunakan suatu mekanisme konvensional yaitu memanfaatkan teknik permutasi dan kombinasi untuk mendapatkan efek *conffusion* dan *disffusion*. Hal ini berbeda teknik pembangkitan bilangan random yang menggunakan persaam matematis.



catatan : umpan 1 = umpan 2 = umpan n

Gambar 1: Semi-*random* menghasilkan *stream random* yang sama

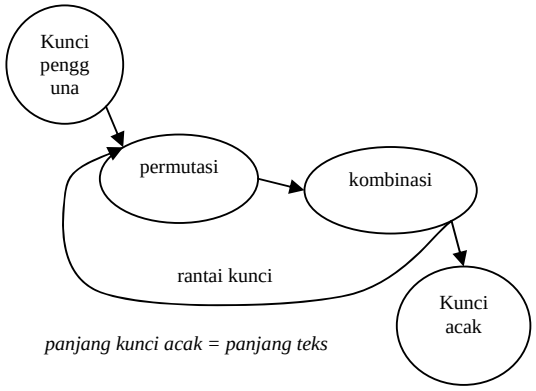
Dari gambar diatas dapat dilihat bahwa semi random menghasilkan suatu stream bilangan random yang sama untuk setiap umpan yang sama. Hal ini akan bermanfaat ketika proses denkripsi karena akan diperlukan kunci yang sama dengan kunci pada saat melakukan enkripsi. Jika *stream* yang dihasilkan tidak sama maka tidak akan dapat dilakukan proses dekripsi lagi untuk mendapatkan *plainteks.*

Prinsip random ini akan dikombinasikan dengan prinsip berantai untuk mendapatkan efek *stream* yang benar-benar bersifat acak. Maksud dari berantai disini adalah akan adanya keterhubungan antara *stream random* yang satu dengan *stream random* yang lainnya, namun dengan tetap menjaga bahwa hal ini tidak akan berpengaruh pada kualitas kunci yang dihasilkan.

3. MEKANISME PEMBANGKITAN KUNCI SEMI-*RANDOM* BERANTAI

Dalam pembangkitan kunci semi-*random* berantai ini digunakan mekanis konvensioanal yaitu dengan memanfaatkan konsep permutasi dan kombinasi untuk mendapatkan efek acak.

Seperti yang telah dijelaskan bahwa kunci disini akan dibangkitkan dari kunci masukan yang diberikan oleh pengguna. Adapun mekanisme umum dalam pembangkitan kunci dapat dilihat pada Gambar 2. Pertama kunci dari pengguna akan diproses dengan metoda permutasi untuk mendapatkan efek acak dalam kunci hasil. Untuk menambahkan efek acak maka hasil dari proses permutasi sebelumnya akan diproses kembali dengan metoda kombinasi. Hasil dari kombinasi ini akan disimpan untuk dijadikan blok pertama kunci acak dan juga akan dijadikan masukan untuk pembentukan blok selanjutnya (proses berantai). Proses tersebut terus diulang sampai didapatkan hasil kunci acak yang panjangnya sama dengan panjang pesan yang akan dienkripsikan.



Gambar 2: Mekanisme pemangkitan kunci semi-*random*

Dapat dilihat disini bahwa terdapat tiga proses penting dalam pembangkitan kunci acak ini, yaitu permutasi, kombinasi, dan rantai kunci dimana dua prinsip awal sering digunakan pada algoritma modern. Detail ke-tiga proses tersebut adalah :

3.1. Proses Permutasi

Pada proses permutasi yang dilakukan adalah mengacak urutan posisi dari elemen pada block suatu blok masukan sehingga pola yang ada aan berubah sesuai dengan pengacakan yang dilakukan. Tujuan dari pengacakan urutan posisi dari elemen blok ini adalah untuk menghasilkan suatu blok baru yang beok rbeda dari blok sebelumnya. Proses permutasi dilakukan melalui dua langkah permutasi yatiu pembalikan urutan elemen blok dengan modifikasi dan punukaran elemen antar elemen tetangga. Adapun proses pengacakan/permutasi yang dilakukan pada posisi elemen blok adalah mengikuti rumus berikut ini:

$$P_i^{'} = \begin{cases} P_{(n-i)} & panjang(P) > 1 \\ P_i << 4 & panjang(P) = 1 \end{cases} \quad (3)$$

yang mana :

$P_i^{'}$ = posisi elemen ke-i pada blok hasil

P_i = posisi elemen ke-i pada blok asal

i = nomor urut posisi blok [0...n]

n = panjang blok + 1

Rumus diatas berfungsi untuk membalikan urutan elemen-elemen pada blok. Sedangkan untuk pertukaran antar elemen tetangga akan mengikuti rumus sebagai berikut:

$$P_i^{''} = P_{i+1}^{'}$$
$$P_{i+1}^{''} = P_i^{'} \quad (4)$$

P' = elemen blok setelah proses pembalikan urutan.
P'' = posisi elemen akhir hasil prose permutasi
i = indeks posisi dimana nilainya ganjil {1,3,5,7,9,11,13.....}

3.2. Proses Kombinasi

Setelah melewati proses permutasi yaitu guna mengacak susunan elemen blok, kemudian dilanjutkan dengan melakukan kombinasi setiap elemen blok dengan elemen lain. Proses ini dikenal dengan proses kombinasi, yaitu suatu teknik mengkombinasikan suatu elemen sehingga didapat elemen baru hasil kombinasi tersebut yang berbeda dari elemen sebelumnya. Proses kombinasi yang dilakukan adalah dengan memanfaatkan fungsi modulo yang dengan mengkombinasikan prinsip pembangkitan nilai random menggunakan sistem LCG (*Linear Congruential Generator*). Adapun fungsi yang dipergunakan dalam proses kobinasi ini adalah:

$$P_i''' = P_i \oplus P_i'' \quad (5)$$

$$P_i'''' = \left(8P_{i-1}''' + panjang(P)\right)(\text{mod}128) \quad (6)$$

Nilai dari *P''''* inilah yang merupakan elemen ke-*i* baru dari suatu blok. Terlihat bahwa pada proses kombinasi juga dilalui dengan dua tahap yaitu menerapkan fungsi *xor* dengan blok sebelumnya guna mendapatkan efek keterkaitan antar blok dan menerapkan proses pembangkitan nilai random berdasarkan prinsip LCG. Hal ini ditujukan untuk mendapatkan efek acak dari hasil sebelumnya.

3.3. Proses Rantai Kunci

Pada proses ini yang dilakukan adalah mengirimkan hasil proses kombinasi untuk dijadikan masukan dalam iterasi selanjutnya untuk proses permutasi. Selain itu hasil dari proses kombinasi juga akan disimpan dan dijadikan blok baru penyusun kunci acak. Dari sini terlihat bahwa penggunaan hasil dari suatu proses sebagai masukan proses selanjutnya mengakibatkan terbentuknya suatu hubungan antar blok didalam kunci hasil.

Ketiga proses diatas akan diulang beberapa kali sehingga didapatkan hasil suatu *stream random* untuk kunci yang panjang sama dengan panjang teks yang akan dienkripsi. Setelah didapatkan hasil yang sama panjang, maka proses diselesaikan kemudian dilanjutkan dengan enkripsi pesan dengan lagoritma *one time pad* menggunakan persamaan (1).

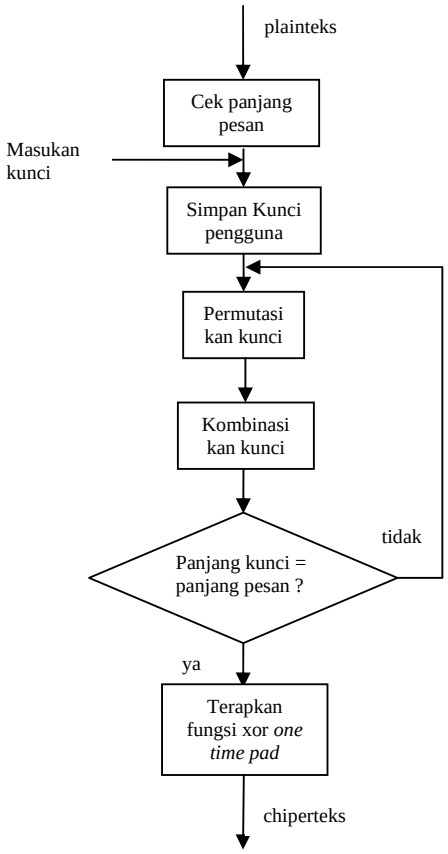
Untuk dekripsi juga akan dilkukan hal yang sama yaitu diawali dengan pembangkitan kunci *stream random* dari kunci masukan pengguna. Setelah kunci terbentuk maka akan diterapkan algorimta *one time pad* dengan persamaan (2). Disini jika masukan kunci pengguna tidak sama dengan masukan kunci pengguna ketika enkripsi, maka *stream random* yang dihasilkan dari pembangkitan kunci akan berbeda sehingga hasil dekripsinya pun tidak akan sama dengan pesan teks yang sesungguhnya. Disamping itu penggunaan prinsip berantai disini akan menambah kebingungan, karena ketika satu elemen kunci masukan saja salah akan berbepangaruh pada keseluruhan hasil dekripsi pesan. Oleh karena itu sulit bagi seorang kriptanalis untuk mengidentifikasi pesan ini dengan metoda pencarian keterhubungan antara chiperteks dan plainteks.

Secara umum maka algoritma one time pada ini setelah diekstensi dapat dijabarkan sebagai berikut:

Algoritma:

1. Simpan kunci pengguna
2. Masukan kunci pengguna ke-dalam proses pembangkitan kunci acak.
    a. Lakukan proses permutasi
    b. Lakukan proses kombinasi
    c. Terapkan prinsip kunci berantai
3. Ulangi proses 3 sampai didapatkan panjang blok kunci acak sama dengan panjang teks yang akan dienkripsi.
4. Terapkan algoritma *one time pad* antara kunci acak dengan pesan teks

Untuk flow chart dari algoritma ini dapat dilihat pada gambar berikut ini.



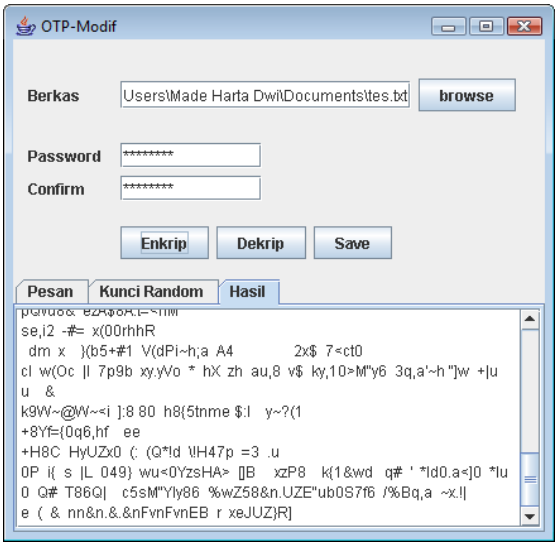
Gambar 3: *Flow-chart* algoritma pembentukan kunci dan *one time pad*

Untuk algoritma proses dekripsi pada prinsipnya sama dengan algoritma untuk enkrispi yang berbeda hanya masukan dan keluarannya saja. Dimana pada proses

dekripsi ini adalah kebalikan dari proses enkripsi.

4. IMPLEMENTASI

Proses yang dijelaskan diatas telah diimplementasikan dengan bahasa pemrograman java dengan memanfaatkan tools NetBeans 5.0 untuk tampilannya. Program yang dibuat memerlukan parameter masukan yang didefinisikan oleh user berupa pesan yang akan dienkripsi atau didekripsi dan masukan kunci pengguna.



Gambar 4 : Hasil implementasi program dengan menggunakan NetBeans

Oleh karena keterbatasan pada bahasa pemrograman, dimana java tidak mendukung tipe *unsigned* [1], maka file hasil enkripsi dan kunci random memuat karakter-karakter aneh yang merepresentasikan nilai minus pada kode ASCII-nya.

5. PENGUJIAN

Untuk pengujian akan dipergunakan file tes yang tidak terlalu panjang demikian juga dengan ukuran kunci dipergunakan ukuran kunci 8 *byte* atau 64 *bit*. Adapun pengujian dilakukan dengan skenario sebagai berikut :

1. Enkripsi skenario normal
   pesan : `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
   kunci : `12345678`
   kunci random : `_ _00_8HP__8(@Xh__XPH(xh__`
   hasil : `Ib[tuV __RCte__8AB __~/0AB`
2. Dekripsi skenario normal
   pesan : `Ib[tuV __RCte__8AB __~/0AB`
   kunci : `12345678`
   kunci random : `_ _00_8HP__8(@Xh__XPH(xh__`
   hasil : `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
3. Dekripsi dengan pengubahan satu karakter pada kunci.
   pesan : `Ib[tuV __RCte__8AB __~/0AB`
   kunci : `12349678`
   kunci random : `_ _0H@8HP___`8_XH_P0_X(PPX`
   hasil : `ABCD=_GHIJKd_6_` J[4_& `__`
4. Dekripsi dengan penambahan satu karakter pada kunci.
   pesan : `Ib[tuV __RCte__8AB __~/0AB`
   kunci : `123495678`
   kunci random : `!_1Q_AA__! _ Y_)Q !a Q_!_`
   hasil : `@CBE$O>A_Cb}t N)h__%|w~1`C`
5. Dekripsi dengan penghapusan satu karakter pada kunci.
   pesan : `Ib[tuV __RCte__8AB __~/0AB`
   kunci : `1234678`
   kunci random : `_/7_Ow_7 /W_7?/wG_w7W 7w`
   hasil : `N}L[BY0wFe<[2Q n5L_jIxOv5`

Pada pengujian diatas terlihat bahwa yang dipergunakan untuk mengenkripsi dan mendekripsi pesan adalah kunci random yang panjangnya sama dengan panjang pesan. Kunci random inilah yang dibentuk dari mekanisme yang telah dijelaskan sebelumya.

Hasil uji memperlihatkan bahwa perubahan satu buah karakter saja pada kunci (kunci pengguna) akan menyebabkan perubahan yang signifikan terhadap hasil dekripsi. Sehingga jika ingin mendapatkan hasill dekripsi yang benar maka harus dimasukan kunci pengguna yang tepat sama antara kunci yang dipergunakan untuk enkripsi dan dekripsi.

6. SERANGAN (*ATTACK*)

Seperti yang telah dijelaskan diatas bahwa *one time pad* adalah satu-satunya algoritma kriptografi yang tidak dapat dipecahkan. Jadi akan tidakm mungkin seseorang melakukan penyerangan terhadapnya untuk mendapatkan pesan asli yang sebelumnya telah terenkripsi. Oleh karena pada mekanisme ini juga memanfaatkan algoritma *one time pad* maka juga tidak akan mungkin melakukan penyerangan terhadap chiper teks dari hasil enkripsi dengan metode ini. Satu-satunya jalan untuk melakukan penyerangan pada metode ini adalah dengan melakukan penyerangan pada kunci random. Seketika setelah kunci pengguna dapat dipecahkan maka pesan yang terenkripsi dapat diketahui.

Penyerangan kunci random hanya bisa dilakukan dengan teknik *brute force attack* atau *exhaustive attck.*

Hal ini dikarenakan kunci random dibentuk sedemikian rupa dengan teknik yang mengaplikasikan sistem pembangkitan bilangan random, sehingga tidak akan terdapat celah lagi yang bisa menghubungkan antara kunci random dengan kunci masukan pengguna. Oleh karenanya serang seperti : *analytical attck, related-key attack, dan rubber-hose attack* tidak dapat digunakan lagi.

Sehingga sekarang hanya tinggal satu jalan lagi untuk dapat membongkar kunci yang dipergunakan untuk mengenkripsi pesan. Teknik *brute force attack* yang dipergunakan akan memakan waktu yang sangat lama jika pengguna memasukan kunci dengan cerdik yaitu dengan menggunakan kombinasi karakter huruf (kapital-kecil) dan angka ataupun karakter simbol lainnya. Dengan mengkombinasikan huruf dan angka saja pada kunci maka pada kunci yang panjangnya berturut-turut:

1. 1 karakter terdapat 62 kombinasi
2. 2 karakter terdapat 3.844 kombinasi
3. 3 karakter terdapat 238.328 kombinasi
4. 4 karakter terdapat 14.776.336 kombinasi
5. 5 karakter terdapat 916.132.832 kombinasi
6. 6 karakter terdapat 56.800.235.584 kombinasi
7. 7 karakter terdapat 3.521.614.606.208 kombinasi
8. 8 karakter terdapat 218.340.105.584.896 kombinasi
9. 9 karakter terdapat 13.537.086.546.263.552 kombinasi
10. 10 karakter terdapat 839.299.365.868.340.224 kombinasi

dst.

Terlihat bahwa setiap penambahan satu karakter pada kunci akan mengakibatkan penambahan secara eksponensial pada jumlah kombinasi kunci yang mungkin. Sekarang anggap kita mempergunakan kunci dengan panjang 8 karakter dan kombinasi huruf dengan angka saja, maka akan terdapat 218.340.105.584.896 kombinasi kemungkinan kunci. Misal, jika waktu yang diperlukan untuk mencoba satu buah kemungkinan kunci adalah 0.5 detik, maka untuk jumlah kunci sebanyak itu diperlukan waktu komputasi kurang lebih selama 3.461.759 tahun. Bisa dibayangkan lamanya waktu yang dibutuhkan untuk memecahkan sebuah kunci saja. Jadi dapat dikatakan dengan kombinasi dan panjang kunci yang baik maka mekanisme ini sangat aman.

Jadi kekuatan dari mekanisme ini adalah pada pemilihan kunci pengguna. Semakin panjang dan semakin banyak kombinasi pada kunci maka mekanisme ini akan semakin kuat dan semakin sulit untuk dipecahkan. Namun jika pengguna memasukan kunci yang relatif pendek dan tanpa kombinasi maka pesan akan dapat dipecahkan dengan relatif mudah.

7. KESIMPULAN

Kekutan mekanisme pembangkitan kunci semi random berantai ini terletak pada pemilihan kunci oleh pengguna. Semakin panjang dan semakin terkombinasi kunci yang dipergunakan maka akan semakin baik.

Mekanisme pembangkitan kunci semi random berantai seperti ini akan memperkuat algoritma *one time pad*. Tidak peduli seperti apa proses yang dilakukan didalam permutasi, kombinasi dan rantai kunci, semasih memegang prinsip pembangkitan nilai random yang benar maka akan didapatkan kunci random yang bila digunakan untuk kunci *one time pad* maka akan didapatkan chiper teks yang tidak terpecahkan (*unbreakable chipper*).

DAFTAR REFERENSI

[1] Knudsen, Jonathan B, "*Java Cryptography*", 1998, O'Reilly.

[2] Munir, Rinaldi, "*Dikat Kuliah IF5054 Kriptografi*" ,2006.

[3] Schneir, Bruce, "*Applied Cryptography Second Edition*", 1996, John Willey & Sons.