ALGORITMA RC4 SEBAGAI METODE ENKRIPSI

Karina Novita Suryani - 13508048

Program Studi Teknik Informatika – Sekolah Teknik Elektro dan Informatika ITB
Jl. Ganesha 10 Bandung 40132
e-mail: if18048@students.if.itb.ac.id

ABSTRAK

Makalah ini akan membahas Algoritma RC4 sebagai salah satu metode enkripsi. Enkripsi adalah proses penyamaran suatu pesan atau sandi menjadi bentuk yang tidak mempunyai makna untuk menjaga kerahasiaan, keamanan, atau keotentikan suatu pesan atau sandi. Pada proses ini plainteks atau teks asli dikonversikan menjadi cipherteks (teks tersandi).
RC4 atau Rivest Cipher 4 dibuat oleh Ron Rivest di Laboraturium RSA pada tahun 1987[2]. RC4 adalah salah satu jenis stream cipher yang sinkron yaitu cipher yang memiliki kunci simetris dan mengenkripsi atau mendekripsi plainteks secara digit per digit atau bit per bit dengan cara mengkombinasikan secara operasi biner (biasanya operasi XOR) dengan sebuah angka semiacak. RC4 dapat dijalankan dengan panjang kunci variabel dan beroperasi dengan orientasi byte.
Metode enkripsi RC4 memiliki kelemahan. Kelemahan yang paling dikenal adalah Bit-Flipping Attack atau BFA di mana penyerang dapat mengetahui sample atau keseluruhan plainteks dari cipherteks tanpa harus mengetahui kunci enkripsi.
Walaupun punya banyak kelemahan, tetapi metode enkripsi RC4 saat ini masih menjadi metode enkripsi yang banyak digunakan. RC4 saat ini masih diaplikasikan pada pengenkripsian PDF, pengamanan WEP, dan SSL.

Kata kunci: *RC4,* Stream Cipher sinkron, enkripsi

1. PENDAHULUAN

Di era globalisasi saat ini, mendapatkan informasi sangatlah mudah. Setiap orang dengan mudah mendapatkan data ataupun berita yang diinginkan. Hal ini didukung dengan teknologi informasi dan komunikasi yang terus berkembang pesat dari tahun ke tahun. Akan tetapi kemudahan mendapatkan informasi juga memberikan ancaman. Beberapa ancaman yang diberikan adalah masalah tentang keamanan, kerahasiaan, dan keotentikan data. Oleh karena itu diperlukan suatu sistem pengamanan data yang bertujuan untuk meningkatkan keamanan data, melindungi suatu data atau pesan agar tidak dibaca oleh pihak yang tidak berwenang, dan mencegah pihak yang tidak berwenang untuk menyisipkan, menhapus, ataupun merubah data.

Salah satu ilmu pengamanan data yang terkenal adalah kriptografi. Kriptografi adalah ilmu sekaligus seni untuk menjaga kerahasiaan pesan, dtaa, atau informasi dengan cara menyamarkannya menjadi bentuk tersandi yang tidak mempunyai makna[1].

Dalam kriptografi, terdapat 2 proses utama, enkripsi dan dekripsi. Enkripsi adalah proses penyandian pesan asli atau plainteks menjadi cipherteks (teks tersandi). Sedangkan dekripsi adalah proses penyandian kembali cipherteks menjadi plainteks. [1].

Salah satu metode enkripsi yang terkenal adalah metode RC4. RC4 pertama kali dibuat oleh Ron Rivest di Laboraturium RSA pada tahun 1987. Awalnya RC4 adalah sebuah rahasia dagang, akan tapi pada September 1994, kode tersebut dikirim oleh seseorang yang tidak diketahui ke milist Chypermunks dan menyebar ke banyak situs internet. Kode yang bocor tersebut akhirnya dikonfirmasi sebagai RC4 karena memiliki output yang sama dengan software dengan license RC4 di dalamnya. Karena algoritma sudah diketahui, RC4 tidak lagi menjadi rahasia dagang. Nama "RC4" sekarang adalah sebuah merek dagang, namun sering disebut sebagai "ARCFOUR" atau "ARC4" (artinya diduga RC4, karena algoritma ini tidak pernah dirilis secara resmi oleh RSA), untuk menghindari kemungkinan masalah tentang merek dagang[2].

RC4 (Rivest Cipher 4) adalah sebuah *synchrone stream cipher*, yaitu cipher yang memiliki kunci simetris dan mengenkripsi plainteks secara digit per digit atau byte per byte dengan cara mengkombinasikan dengan operasi biner (biasanya XOR) dengan sebuah angka semiacak.

2. MEKANISME DASAR KERJA RC4

Algoritma RC4 adalah algoritma kriptografi simetrik. Disebut algoritma kriptografi simetrik karena menggunakan kunci yang sama untuk mengenkripsi ataupun mendekripsi suatu pesan, data, ataupun informasi. Kunci enkripsi didapat dari sebuat 256 bit *state-array*

yang diinisialisasi dengan sebuah *key* tersendiri dengan panjang 1-256 bit. Setelah itu, *state-array* tersebut akan diacak kembali dan diproses untuk menghasilkan sebuah kunci enkripsi yang akan di-XOR-kan dengan plainteks ataupun cipherteks.

Secara umum, algoritma RC4 terbagi menjadi dua, inisalisasi *state-array* dan penghasilan kunci enkripsi serta pengenkripsiannya.

2.1 Penginisialisasian *State-Array*

Dalam penginisialisasian *state-array*, terdapat 2 *state-array* yang harus diinisialisasi, S dan K. Array S sebesar 256 bit diinisialisasi dengan angka dari 0 sampai dengan 255. Sedangkan array K sebesar 256 bit diisi dengan key dengan panjang 1-256 bit secara berulang sampai seluruh array K terisi penuh. Setelah itu, dilakukan Key Scheduling Algorithm untuk menghasilkan permutasi dari array S berdasarkan key yang tersedia.

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod
keylength]) mod 256
    swap(&S[i],&S[j])
endfor
```

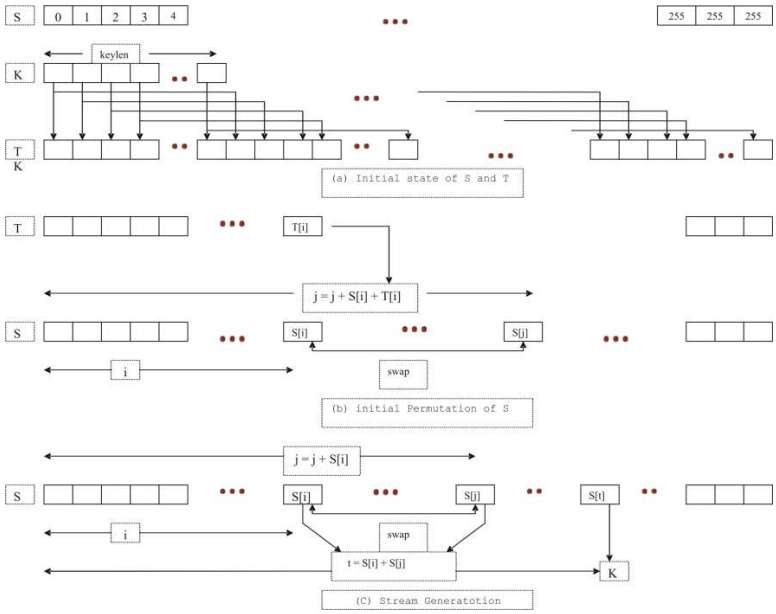
Proses swap di atas adalah proses menukar nilai antara S[i] dengan S[j]. Proses swap memiliki algoritma sebagai berikut.

```
temp = s[i];
s[i] = s[j];
s[j] = temp;
```

2.2 Penghasilan Kunci Enkripsi dan Pengenkripsian

Setelah memiliki state array yang telah teracak, maka kita akan menginisialisasi kembali i dan j dengan 0. Setekah itu, kita lakukan pseudo-random generation algorithm atau PRGA untuk menghasilkan kunci enkripsi (dalam hal ini cipher_byte) yang akan di-XOR-kan dengan plainteks. Untuk menghasilkan kunci enkripsi, PRGA meng-*incremen* i, menambahkan nilai S[i] dan S[j] menukar nilai keduanya, dan nilai kunci yang dihasilkan adalah S dengan indeks yang sama dengan jumlah S[i] dan S[j] di-modulo dengan 256.

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(&S[i],&S[j])
    byte_cipher := S[(S[i] + S[j])
mod 256]
    result_ciphered := byte_cipher
XOR byte_message
endwhile
```



Gambar 1 Ilustrasi Algoritma RC4

2.3 Contoh Enkripsi dengan metode RC4

Untuk menunjukkan bagaimana RC4 bekerja pada tingkat dasar, mari kita state-array 4 bit. Hal ini dikarenakan akan sangat sulit menggambarkan proses RC4 secara manual dengan state-array 256 bit.

Kali ini, kita akan mengenkripsi kata HALO dengan kunci 3123.

Pertama, kita menginisialisasi array S 4 bit sehingga terbentuk state-array S dan state-array K sebagai berikut,

Array S

0	1	2	3

Array K

2	5	7	3

Inisialisasi i dan j dengan 0 kemudian dilakukan KSA agar tercipta state-array yang acak. Penjelasan iterasi lebih lanjut dapat dijelaskan sebagai berikut.

Iterasi 1

```
i = 0
j = (0 + S[0] + K [0 mod 4]) mod 4
  = (0 + 0 + 2) mod 4 = 2
Swap (S[0],S[2])
```

Hasil Array S

2	1	0	3

Iterasi 2

```
i = 1
j = (2 + S[1] + K [1 mod 4]) mod 4
  = (2 + 1 + 5) mod 4 =0
Swap (S[1],S[0])
```

Hasil Array S

1	2	0	3

Iterasi 3

```
i = 2
j = (0 + S[2] + K [2 mod 4]) mod 4
  = (0 + 0 + 7) mod 4 = 3
Swap (S[2],S[3])
```

Hasil

1	2	3	0

Iterasi 4

```
i = 3
j = (3 + S[3] + K [3 mod 4]) mod 4
  = (3 + 0 + 3) mod 4 = 2
Swap (S[3],S[2])
```

Hasil Array S

1	2	0	3

Setelah melakukan KSA, akan dilakukan PRGA. PRGA akan dilakukan sebanyak 4 kali dikarenakan plainteks yang akan dienkripsi berjumlah 4 karakter. Hal ini disebakan karena dibutuhkan 1 kunci dan 1 kali pengoperasian XOR untuk tiap-tiap karakter pada plainteks.

Berikut adalah tahapan penghasilan kunci enkripsi dengan PRGA.

Array S

1	2	0	3

Inisialisasi

```
i = 0
j = 0
```

Iterasi 1

```
i = (0 + 1) mod 4 = 1
j = (0 + S[1]) mod 4 = (0+ 2) mod
4 = 2
swap (S[1],S[2])
```

1	0	2	3

```
K1 = S[(S[1]+S[2]) mod 4] = S[2
mod 4] = 2
K1 = 00000010
```

Iterasi 2

```
i = (1 + 1) mod 4 = 2
j = (2 + S[2]) mod 4 = (2+ 2) mod
4 = 0
swap (S[2],S[0])
```

2	0	1	3

```
K2 = S[(S[2]+S[0]) mod 4] = S[3
mod 4] = 3
K2 = 00000011
```

Iterasi 3

```
i = (2 + 1) mod 4 = 3
j = (0 + S[3]) mod 4 = (0+ 3) mod
4 = 3
swap (S[3],S[3])
```

1	0	2	3

```
K3 = S[(S[3]+S[3]) mod 4] = S[6
mod 4] = 2
K3 = 00000010
```

Iterasi 4

```
i = (3 + 1) mod 4 = 0
j = (3 + S[0]) mod 4 = (3+ 1) mod
4 = 0
swap (S[0],S[0])
```

1	0	2	3

```
K1 = S[(S[0]+S[0]) mod 4] = S[2
mod 4] = 2
K1 = 00000010
```

Setelah menemukan kunci untuk tiap karakter, maka dilakukan operasi XOR antara karakter pada plaintext dengan kunci yang dihasilkan. Berikut adalah tabel ASCII untuk tiap-tiap karakter pada plaintks yang digunakan.

Tabel 1 Kode ASCII untuk setiap karakter plainteks yang digunakan

Huruf	Kode ASCII (Binary 8 bit)
H	01001000
A	01000001
L	01001100
O	01001111

Proses XOR dari kunci bisa dilihat pada tabel 2.

Tabel 2 Proses XOR kunci enkripsi dengan plainteks pada enkripsi

	H	A	L	O
Plainteks	01001000	01000001	01001100	01001111
Key	00000010	00000011	00000010	00000010
Cipherteks	01001010 (L)	01000010 (B)	01001110 (N)	01001101 (M)

Setelah terkirim, pesan yang telah dienkripsi akan didekripsikan. Proses pendekripsian dilakukan dengan proses XOR antara kunci dekripsi yang sama dengan kunci dekripsi dengan cipherteks yang dapat dilihat di tabel 3.

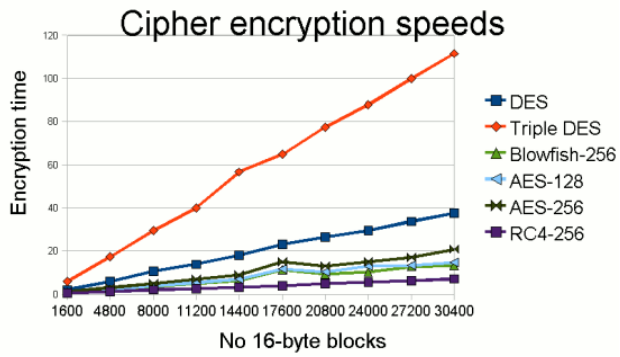
Tabel 3 Proses XOR kunci dekripsi dengan cipherteks pada dekripsi

	L	B	N	M
Cipherteks	01001010	01000010	01001110	01001101
Key	00000010	00000011	00000010	00000010
Plainteks	01001000 (H)	01000001 (A)	01001100 (L)	01001111 (O)

2.2. Kecepatan RC4 Sebagai Salah Satu Metode Enkripsi

Kinerja RC4 sebagai metode enkripsi tergolong sangat cepat. Selain cepat, waktu RC4 tidak terpengaruh dengan panjang keylength yang dipakai. Berikut adalah perbandingan waktu yang digunakan untuk enkripsi dari berbagai metode.

Gambar 2 Perbandingan waktu enkripsi dari berbagai metode



2.2. Permasalahan RC4 dan penanganannya

Walaupun cepat, RC4 salah satu metode enkripsi yang tidak aman dikarenakan banyaknya kelemahan yang bisa diserang oleh kriptoanalis. Akan tetapi, beberapa kelemahan dari RC4 telah bisa diatasi. Beberapa kelemahannya antara lain adalah terjadinya array S yang sama ataupun berulang dan Bit-Flipping Attack

Masalah pertama adalah tingginya peluang pengacakan array S yang akan menghasilkan array S yang sama seperti sebelumnya ataupun terisi berulang. Jika user memaksukkan kunci ‘ii’ dengan ‘iiii’, akan menghasilkan array S yang sama. Hal ini disebabkan karena kunci dari user diisi ke dalam array K secara berulang sampai array K terisi penuh. Walaupun memungkinkan memiliki kunci dengan 256 byte, akan tetapi sangat susah menemukan kombinasi kunci tersebut dan susah juga untuk mengingatnya.

Hal ini bisa diatasi dengan menggunakan kunci yang cukup panjang sehingga kemungkinan berulangnya key dalam array bisa diperkecil dan dipersulit kominasi kuncinya. Pengisian array K yang berbeda juga bisa mengatasi masalah ini. Yang dimaksud berbeda di sini adalah dengan cara mengisi array dengan key cukup sekali kemudian sianya diisi secara random.

Selain itu, masalah ini juga bisa diselesaikan dengan menerapkan hasil hash 160 bit SHA dari password kita untuk mencegah hal ini terjadi.

Masalah kedua adalah masalah Bit-Flipping Attack. Bit Flipping Attack adalah serangan terhadap sebuah sandi kriptografi di mana penyerang dapat mengubah ciphertext sedemikian rupa untuk menghasilkan perubahan yang dapat diprediksi plaintext, meskipun penyerang tidak dapat belajar plaintext itu sendiri. Pada RC4, hal ini dapat dilakukan dengan cara meng-XOR sebagian ciphertext dari 2 pesan yang berbeda dengan kunci yang sama. Jika diketahui plaintext dari salah satu pesan, maka dengan mudah dapat diketahui plainteks lain tanpa harus mengetahui rangkaian kuncinya. Hal ini menjadi sangat berbahaya apabila seseorang mengganti sebagian informasi penting dari plainteks. Apabila plainteks berbunyi “Saya berhutang padamu Rp 1000,00”, dengan

mudah plainteks dapat diubah menjadi “Saya berhutang padamu Rp 100.000”.

Masalah ini bisa diselesaikan dengan cara menggunaka Initialization Vector (IV) dalam kunci. IV adalah suatu blok bit yang diperlukan untuk memungkinkan sebuah stream cipher atau cipher blok menghasilkan keystream independen yang unik yang dihasilkan oleh kunci enkripsi yang sama. Hal ini memungkinkan untuk mendapatkan hasil enkripsi yang berbeda walaupun memiliki kunci yang sama. Selain itu. Mengacak plainteks sebelum dienkripsi juga memperkecil kemungkinan terjadinya bit-flipping attack pada RC4.

2.2. Aplikasi RC4

Walaupun RC4 memiliki banyak kelemahan akan tetapi RC4 masih banyak digunakan. RC4 diaplikasikan untuk pengamanan beberapa hal seperti berikut.

1. *WEP (Wired Equivalent Privacy)*
2. BitTorrent protocol encryption
3. Microsoft Point-to-Point Encryption
4. Gpcode.AK,
5. PDF

IV. KESIMPULAN

Kesimpulan yang dapat diambil dari studi pemakaian metode RC4 sebagai metode enkripsi adalah ,

1. RC4 adalah sebuah stream cipher yang sinkron yang dapat dijalankan dengan panjang kunci variabel dan mengenkripsi suatu plainteks secara digit per digit dengan kunci simetris
2. RC4 merupakan metode enkripsi tercepat dibandingkan dengan DES, Triple DES, Blowfish-256, AES-128, dan AES-256.
3. RC4 memiliki banyak kelemahan antara lain, tingginya peluang untuk menghasilkan array S yang sama ataupun berulang dan Bit-Flipping Attack. Akan tetapi bisa diatasi dengan memperbanyak bit kunci, mengubah cara pengisian K-array, menggunakan IV dalam setiap kunci, serta mengacak plainteks sebelum dienkripsi untuk mencegah terjadinya bit-flipping attack.

REFERENSI

[1] Munir, Rinaldi, “Diktat Kuliah IF2151 Matematika Diskrit Edisi Keempat”, Departemen Teknik Informatika Institut Teknologi Bandung, 2004, Hal V-25,.

[2] http://en.wikipedia.org/wiki/RC4
Tanggal akses 18 Desember 2009 19.00

[3] http://en.wikipedia.org/wiki/Bit-flipping_attack
Tanggal akses 18 Desember 2009 19.13

[4] http://bimacipta.com/rc4.htm
Tanggal akses 18 Desember 2009 19.40

[5] http://www.wireless-center.net/Wi-Fi-Security/2209.html
Tanggal akses 19 Desember 2009 9.55

[6] journal.uii.ac.id/index.php/Snati/article/view/1645/1420
Tanggal akses 19 Desember 2009 11.00

[7] http://en.wikipedia.org/wiki/Stream_cipher#Synchronous_stream_ciphers\
Tanggal akses 19 Desember 2009 13.26

[8] http://en.wikipedia.org/wiki/Initialization_vector
Tanggal akses 19 Desember 2009 16.10

[9] http://en.wikipedia.org/wiki/Known-plaintext_attack
Tanggal akses 19 Desember 2009 13.40

[10] http://www.javamex.com/tutorials/cryptography/ciphers.shtml
Tanggal akses 19 Desember 2009 19.00