

.....

TUGAS

Keamanan Sistem Lanjut EL 7010

ENKRIPSI DATA KUNCI SIMETRIS DENGAN ALGORITMA KRIPTOGRAFI LOKI97

.....

Oleh :

Avon Budiyo

NIM : 23202049



PROGRAM MAGISTER TEKNOLOGI INFORMASI
DEPARTEMEN ELEKTRO
INSTITUT TEKNOLOGI BANDUNG
2004

Daftar Isi

1. Pendahuluan
2. Kriptografi
 - 2.1 Terminologi
 - 2.2 Algoritma Kriptografi
 - 2.3 Fungsi *Hash* Satu Arah
 - 2.4 Tanda Tangan Digital
 - 2.5 Dasar Matematis
 - 2.6 Kunci Lemah dan Kunci Setengah Lemah
 - 2.7 Mode Operasi
 - 2.8 Proses Padding
3. Algoritma Kriptografi LOKI97
 - 3.1 Elemen Pembangun LOKI97
 - 3.2 Algoritma Enkripsi
 - 3.3 S-Box
 - 3.4 Algoritma Dekripsi
 - 3.5 Sub Kunci
4. Unjuk Kerja Algoritma Kriptografi LOKI97
 - 4.1 Kajian Secara Teoritis
 - 4.2 Avalanche Effect
 - 4.3 Kunci Lemah dan Kunci Setengah Lemah
 - 4.4 Kelemahan Algoritma LOKI97
5. Kesimpulan

Referensi

ENKRIPSI DATA KUNCI SIMETRIS DENGAN ALGORITMA KRIPTOGRAFI LOKI97

1. Pendahuluan

Masalah keamanan merupakan salah satu aspek penting dari sebuah system informasi. Salah satu hal yang penting dalam komunikasi menggunakan komputer dan dalam jaringan komputer untuk menjamin keamanan pesan, data, ataupun informasi adalah enkripsi. Disini enkripsi dapat diartikan sebagai kode atau *chipper*. Sebuah sistem pengkodean menggunakan suatu tabel atau kamus yang telah didefinisikan untuk kata dari informasi atau yang merupakan bagian dari pesan, data, atau informasi yang di kirim. Sebuah *chipper* menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) bit dari suatu pesan asli (*plaintext*) menjadi *cryptogram* yang tidak di mengerti. Karena sistem *chipper* merupakan suatu sistem yang telah siap untuk di outomasi, maka teknik ini digunakan dalam sistem keamanan jaringan komputer.

National Institute of Standard and Technology (NIST) untuk pertama kalinya mengumumkan suatu algoritma standar untuk enkripsi yaitu DES (Data Encryption Standard). Hal ini menyebabkan algoritma DES menjadi algoritma yang digunakan diseluruh dunia untuk proses enkripsi.

Selain kepopulerannya itu, DES mempunyai kelemahan, yaitu kunci yang digunakan pada algoritma DES terlalu pendek untuk dapat diterima pada keamanan data tingkat tinggi seperti yang dibutuhkan saat ini. Triple-DES muncul sebagai alternative solusi untuk masalah-masalah yang membutuhkan keamanan data tingkat tinggi seperti, perbankan, tetapi ia terlalu lambat pada beberapa penggunaan.

Untuk menanggapi keinginan agar mengganti algoritma DES sebagai standar enkripsi, pada tahun 1997 NIST mengeluarkan program Advanced

Encryption Standard (AES). NIST bertugas menilai algoritma-algoritma yang ditawarkan oleh para ahli-ahli kriptografi untuk nantinya dijadikan sebagai algoritma standar pengganti DES dengan memperhatikan kepuasan para pengguna kriptografi.

2. Kriptografi

2.1. Terminologi

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. Kriptografi (*Cryptography*) berasal dari bahasa Yunani yaitu “*Crypto*” berarti “*secret*” (rahasia) dan “*graphy*” berarti “*writing*” (tulisan). Para pelaku atau praktisi kriptografi disebut **cryptographers**. Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut **cipher**, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat.

Proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah **enkripsi** (*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah. Menurut ISO 7498-2, terminologi yang lebih tepat digunakan adalah “*encipher*”. Proses sebaliknya, untuk mengubah *ciphertext* menjadi *plaintext*, disebut **dekripsi** (*decryption*). Menurut ISO 7498-2, terminologi yang lebih tepat untuk proses ini adalah “*decipher*”. *Cryptanalysis* adalah seni dan ilmu untuk memecahkan *ciphertext* tanpa bantuan kunci. *Cryptanalyst* adalah pelaku atau praktisi yang menjalankan *cryptanalysis*. *Cryptology* merupakan gabungan dari *cryptography* dan *cryptanalysis*.

Untuk mengenkripsi dan mendekripsi data. Kriptografi menggunakan suatu algoritma (cipher) dan kunci (key). Cipher adalah fungsi matematika yang digunakan untuk mengenkripsi dan mendekripsi data. Sedangkan kunci

merupakan sederetan bit yang diperlukan untuk mengenkripsi dan mendekripsi data.

Algoritma kriptografi modern tidak lagi mengandalkan keamanannya pada kerahasiaan algoritma tetapi kerahasiaan kunci. Plaintext yang sama bila disandikan dengan kunci yang berbeda akan menghasilkan ciphertext yang berbeda pula. Dengan demikian algoritma kriptografi dapat bersifat umum dan boleh diketahui oleh siapa saja, akan tetapi tanpa pengetahuan tentang kunci, data tersandi tetap saja tidak dapat terpecahkan. Sistem kriptografi atau *Cryptosystem* adalah sebuah algoritma kriptografi ditambah semua kemungkinan plaintext, ciphertext dan kunci.

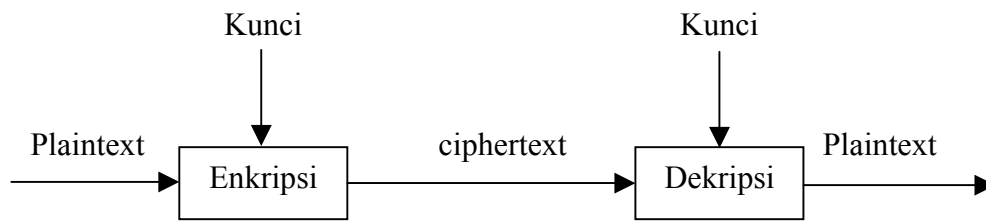
2.2. Algoritma Kriptografi

Berdasarkan kunci yang dipakai, algoritma kriptografi dapat dibedakan atas dua golongan, yaitu :

a. Kunci Simetris

Kunci Simetris adalah jenis kriptografi yang paling umum digunakan. Kunci untuk membuat pesan yang di sandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut termasuk pihak-pihak yang tidak diinginkan dapat membuat dan membongkar rahasia *ciphertext*. Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*).

Proses enkripsi-dekripsi algoritma kriptografi kunci simetris dapat dilihat pada gambar dibawah ini :



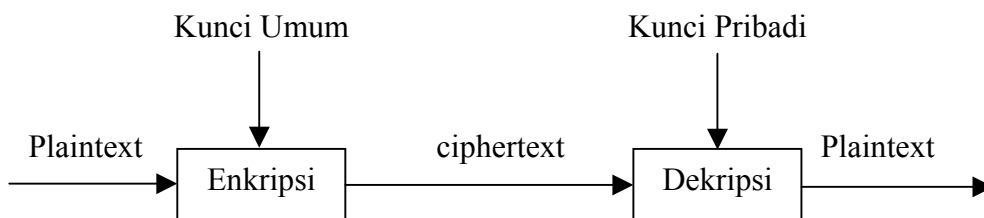
Gambar 1. Proses enkripsi-dekripsi kunci simetris

Algoritma kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu bit atau satu byte data. Sedangkan pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau byte data (per blok).

b. Kunci Asimetris

Kunci asimetris adalah pasangan kunci kriptografi yang salah satunya digunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi. Semua orang yang mendapatkan kunci publik dapat menggunakannya untuk mengenkripsikan suatu pesan, data ataupun informasi, sedangkan hanya satu orang saja yang memiliki rahasia tertentu dalam hal ini kunci privat untuk melakukan pembongkaran terhadap sandi yang dikirim untuknya. Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA.

Proses enkripsi-dekripsi algoritma kunci asimetris dapat dilihat pada gambar dibawah ini :



Gambar 2. Proses enkripsi-dekripsi kunci Asimetris

Pada algoritma public key ini, semua orang dapat mengenkripsi data dengan memakai public key penerima yang telah diketahui secara umum. Akan tetapi data yang telah terenkripsi tersebut hanya dapat didekripsi dengan menggunakan private key yang hanya diketahui oleh penerima.

2.3. Fungsi *Hash* Satu Arah

Fungsi *hash* satu arah (*one-way hash function*) digunakan untuk membuat sidik jari (*fingerprint*) dari suatu dokumen atau pesan X. Pesan X (yang besarnya dapat bervariasi) yang akan di-*hash* disebut *pre-image*, sedangkan outputnya yang memiliki ukuran tetap, disebut *hash-value* (nilai *hash*). Fungsi *hash* dapat diketahui oleh siapapun, tak terkecuali, sehingga siapapun dapat memeriksa keutuhan dokumen atau pesan X tersebut. Tak ada algoritma rahasia dan umumnya tak ada pula kunci rahasia. Contoh algoritma fungsi *hash* satu arah adalah MD-5 dan SHA. *Message Authentication Code* (MAC) adalah salah satu variasi dari fungsi *hash* satu arah, hanya saja selain *pre-image*, sebuah kunci rahasia juga menjadi input bagi fungsi MAC.

2.4. Tanda Tangan Digital

Selama ini, masalah tanda tangan digital masih sering di permasalahkan keabsahannya, hal ini terjadi karena pengertian dan konsep dasarnya belum dipahami. Penandatanganan digital terhadap suatu dokumen adalah sidik jari dari dokumen tersebut beserta *timestamp*-nya di enkripsi dengan menggunakan kunci privat pihak yang menandatangani. Tanda tangan digital memanfaatkan fungsi *hash* satu arah untuk menjamin bahwa tanda tangan itu hanya berlaku untuk dokumen yang bersangkutan saja. Keabsahan tanda tangan digital itu dapat diperiksa oleh pihak yang menerima pesan.

2.5. Dasar matematis

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu himpunan berisi elemen plaintext dan himpunan berisi elemen ciphertext. Enkripsi dan dekripsi merupakan fungsi transformasi antara dua himpunan tersebut. Bila himpunan plaintext dinotasikan dengan P dan himpunan ciphertext dinotasikan dengan C, sedang fungsi enkripsi dinotasikan dengan E dan fungsi dekripsi dengan D maka proses enkripsi-dekripsi dapat dinyatakan dalam notasi matematis dengan :

$$E(P) = C \text{ dan}$$

$$D(C) = P$$

Karena proses enkripsi-dekripsi bertujuan memperoleh kembali data asal, maka :

$$D(E(P)) = P$$

Relasi antara himpunan plaintext dengan himpunan ciphertext harus merupakan fungsi korespondensi satu-satu (*one to one relation*). Hal ini merupakan keharusan untuk mencegah terjadinya ambiguitas dalam dekripsi yaitu satu elemen ciphertext menyatakan lebih dari satu elemen plaintext.

Pada metode kriptografi simetris atau konvensional digunakan satu buah kunci. Bila kunci dinotasikan dengan 'K' maka proses enkripsi-dekripsi metode kriptografi simetris dapat dinotasikan dengan :

$$E_k(P) = C \text{ dan}$$

$$D_k(C) = P$$

Dan keseluruhan sistem dinyatakan sebagai :

$$D_k(E_k(P))=P$$

Pada metode kriptografi nirsimetris digunakan kunci umum untuk enkripsi dan kunci pribadi untuk dekripsi. Bila kunci umum dinotasikan dengan 'PK' dan kunci pribadi dinotasikan dengan 'SK' maka proses enkripsi-dekripsi metode kriptografi nirsimetris dapat dinotasikan dengan :

$$E_{PK}(P) = C \text{ dan}$$

$$D_{SK}(C) = P$$

Dan keseluruhan sistem dinyatakan sebagai

$$D_{SK}(E_{PK}(P)) = P$$

Aritmetika Modular

Aritmetika modular merupakan operasi matematika yang banyak diimplementasikan pada metode kriptografi. Pada metode kriptografi simetris, operasi aritmetika modular yang sering dipakai adalah operasi penjumlahan modulo dua dan operasi XOR (Exclusive OR) dengan simbol \oplus . operasi modulo dua ini melibatkan bilangan 0 dan 1 saja sehingga identik dengan bit pada komputer. Seluruh kemungkinan nilai operasi XOR ini dapat dilihat pada table dibawah ini :

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Dari tabel diatas dapat dilihat sifat-sifat unik operasi XOR yaitu :

$A \oplus A = 0$, $A \oplus 0 = A$, $A \oplus 1 = A'$, dengan A' adalah komplemen dari A .

2.6. Kunci Lemah dan Kunci Setengah Lemah

Beberapa kunci yang digunakan pada proses penyandian dapat merupakan kunci lemah atau kunci setengah lemah. Dalam algoritma kriptografi yang baik, jumlah kunci lemah maupun setengah lemah adalah sangat kecil.

Kunci lemah

Kunci lemah (*weak key*) adalah sebuah kunci yang mengakibatkan proses enkripsi tidak berbeda dengan proses dekripsi [1]. Bila kunci lemah 'K' maka secara matematis dapat dinyatakan :

$$E_K(P) = D_K(P)$$

Sehingga dipenuhi pula persamaan :

$$E_K(E_K(P)) = P \text{ dan}$$

$$D_K(D_K(P))=P$$

Kunci setengah lemah

Sepasang kunci setengah lemah (*Pair of Semi Weak Key*) adalah sepasang kunci yang sering mengakibatkan proses enkripsi dengan menggunakan kunci yang satu sama dengan proses dekripsi dengan menggunakan kunci yang lain [1]. Bila sepasang kunci setengah lemah adalah 'K' dan 'K*' maka secara matematis dapat dinyatakan :

$$E_K(P)=D_{K^*}(P) \text{ dan}$$

$$D_K(P)=E_{K^*}(P)$$

Sehingga dipenuhi pula persamaan

$$E_{K^*}(E_K(P))=P \text{ dan}$$

$$D_{K^*}(D_K(P))=P$$

2.7. Mode Operasi

Ada beberapa mode operasi yang digunakan dalam mengenkripsi / mendekripsi data. Mode operasi tersebut diantaranya :

1. Electronic Code Book (ECB)

Pada mode ini setiap blok plaintext dienkrpsi secara independen menjadi blok cipher. Secara matematis dapat dinyatakan :

$$C_i=E_k(P_i)$$

$$P_i=D_k(C_i)$$

2. Cipher Block Chaining (CBC)

Pada proses enkripsi mode CBC, blok plaintext terlebih dahulu di-XOR-kan dengan blok ciphertext hasil enkripsi blok sebelumnya. Blok pertama plaintext di-XOR-kan dengan suatu Initialization Vector (Vektor Awal) yang besarnya sama dengan blok plaintext. Secara matematis dapat dinyatakan :

$$C_i = E_k(P \oplus C_{i-1})$$

$$P_i = C_{i-1} \oplus D_k(C_i)$$

2.8. Proses padding

Proses padding adalah proses penambahan byte-byte dummy berupa karakter NULL pada byte-byte sisa yang masih kosong pada blok terakhir plaintext, sehingga ukurannya menjadi sama dengan ukuran blok penyandian. Byte terakhir kemudian diisi dengan suatu informasi mengenai ukuran file pada blok terakhir.

Ukuran arsip yang akan disandikan sebagian besar tidak merupakan kelipatan ukuran blok penyandian. Hal ini mengakibatkan blok terakhir mungkin akan memiliki ukuran yang lebih kecil dari blok penyandian. Karena pada Block Cipher mengharuskan blok yang akan disandikan memiliki panjang yang tetap maka pada blok terakhir tersebut harus ditambahkan byte-byte tertentu sehingga ukurannya menjadi sama dengan ukuran blok penyandian.

3. Algoritma Kriptografi LOKI97

Algoritma Kriptografi LOKI97 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan kepada NIST. Algoritma Kriptografi LOKI97 dirancang oleh L. Brown dan J. Pieprzyk. NIST menentukan beberapa kriteria, diantaranya adalah kunci yang digunakan harus panjang, ukuran blok yang digunakan harus lebih besar, lebih cepat, dan fleksibel. Salah satu kandidat untuk AES adalah algoritma LOKI97.

Algoritma ini telah memenuhi semua criteria yang diajukan oleh NIST. LOKI97 adalah algoritma yang menggunakan ukuran blok data sebesar 128 bit dan menggunakan kunci yang dapat bervariasi yaitu 128,192 dan 256 bit.

3.1. Elemen Pembangun LOKI97

Algoritma LOKI97 dalam implementasinya menggunakan elemen-elemen sebagai berikut :

- **Feistel Network**

Feistel Network adalah metode yang umum digunakan pada algoritma kriptografi block cipher. Bagian utama dari feistel network adalah fungsi f , yaitu fungsi pemetaan string input menjadi string output. Pada tiap iterasi, blok sumber merupakan input bagi fungsi f dan kemudian, keluaran dari fungsi f tersebut di XOR dengan blok tujuan, setelah itu kedua blok tersebut dipertukarkan. Algoritma LOKI97 menggunakan 16 iterasi.

- **Data Computation**

LOKI97 melakukan enkripsi data plaintext dengan ukuran blok 128 bit untuk menghasilkan ciphertext dengan ukuran 128 bit. Data computation akan membagi 128 bit plaintext menjadi dua-64 yaitu L dan R.

- **Penjadwalan Kunci**

LOKI97 menggunakan penjadwalan kunci berdasarkan pada feistel network [SK96}, pengoperasiannya pada empat buah 64 bit data. Dengan menggunakan fungsi f yaitu $f(A,B)$ sebagai penghitungan data untuk memberikan non-linear yang cukup dan memastikan bahwa penghitungan dengan kunci dapat dilakukan. Penjadwalan kunci dapat diinisialisasi dengan ukuran yang bervariasi yaitu 128 bit, 192 bit dan 256 bit, berdasarkan ukuran dari kunci yang dapat dipakai yaitu pada 4 buah 64 bit data $[K_4|K_3|K_2|K_1]$.

- **Fungsi f(A,B)**

Fungsi f(A,B) non linear akan menerima dua buah 64 bit masukan nilai A dan B, memprosesnya dengan menggunakan dua buah S box, untuk menghasilkan keluaran 64 bit. Tiga buah permutasi digunakan untuk memastikan avalanche yang maximal pada fungsi tersebut untuk semua bit.

- **S-Box**

Merupakan suatu table substitusi yang digunakan pada kebanyakan algoritma block cipher. S-box pertama kali digunakan pada Lucifer, kemudian DES dan setelah itu banyak algoritma yang menggunakan LOKI97 menggunakan dua buah S-box yang kesemuanya dibangun dengan menggunakan permutasi bit 8x8 dan kunci.

3.2. Algoritma Enkripsi

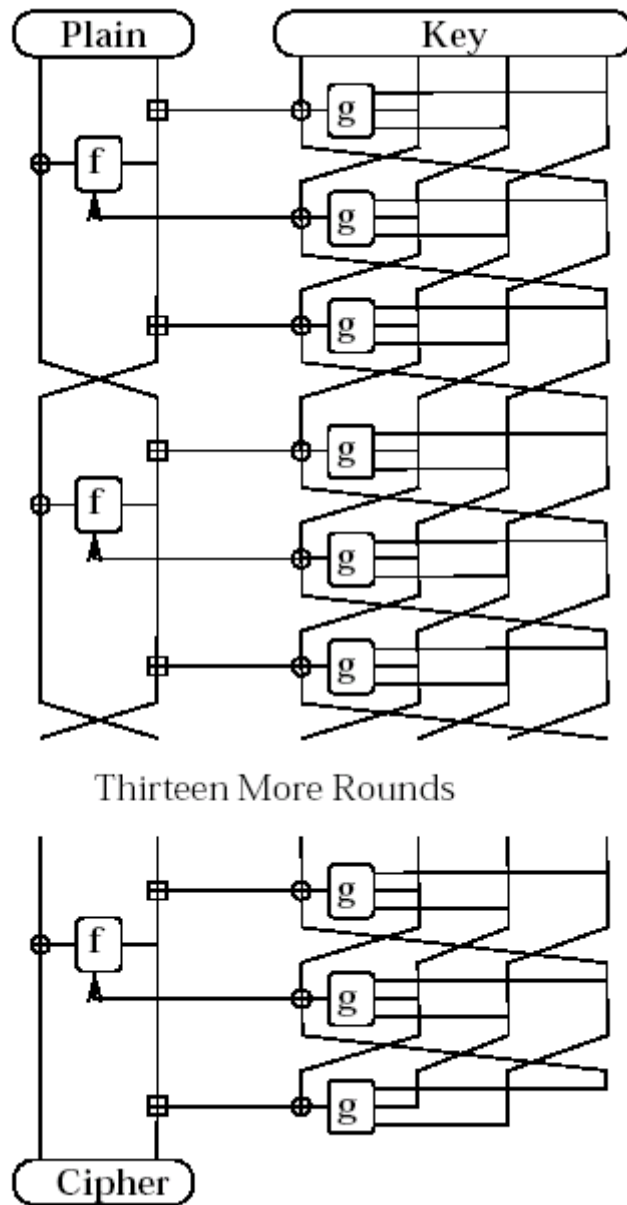
Dalam menggunakan enkripsi suatu blok, langkah yang pertama dilakukan adalah penghitungan data dengan membagi dua masukan blok ukuran 128 bit plaintext menjadi dua buah 64 bit data yaitu [L|R] terdiri dari $L_0=L$; $R_0=R$.

Proses ini akan mengalami 16 kali iterasi pada feistel network. Setiap iterasi menggunakan operasi XOR dan operasi + untuk nilai 64 bit data dengan keluaran dari fungsi kompleks non linear $f(A,B)$ yang akan menghasilkan nilai avalanche yang maksimal dari semua masukan bit sebagai berikut :

$$R_i = L_{i-1} \text{XOR } f(R_{i-1} + SK_{3i-2}, SK_{3i-1})$$

$$L_i = R_{i-1} + SK_{3i-2} + SK_{3i}$$

Hasil dari proses enkripsi dari masukan 128 bit data plaintext menjadi keluaran 128 bit ciphertext adalah sebagai berikut : $[R_{16}|L_{16}]$



Gambar 3. Algoritma LOKI97

3.2.1. Penjadwalan Kunci

LOKI97 menggunakan penjadwalan kunci berdasarkan pada feistel network, pengoperasiannya pada empat buah 64 bit data. Dengan menggunakan fungsi f yaitu $f(A,B)$ sebagai penghitungan data untuk memberikan non-linear yang cukup dan memastikan bahwa penghitungan dengan kunci dapat dilakukan.

Penjadwalan kunci dapat diinisialisasi dengan ukuran yang bervariasi yaitu 128 bit, 192 bit dan 256 bit, berdasarkan ukuran dari kunci yang dapat dipakai yaitu pada 4 buah 64 bit data $[K4_0|K3_0|K2_0]$ sebagai berikut:

- Dengan 256 bit kunci

$$[K_a|K_b|K_c|K_d], [K4_0|K3_0|K2_0|K1_0] = [K_a|K_b|K_c|K_d]$$

- Dengan 192 bit kunci

$$[K_a|K_b|K_c], [K4_0|K3_0|K2_0|K1_0] = [K_a|K_b|K_c|f(K_a, K_b)]$$

- Dengan 128 bit kunci

$$[K_a|K_b], [K4_0|K3_0|K2_0|K1_0] = [K_a|K_b|f(K_b, K_a)|f(K_a, K_b)]$$

Kesemua kunci ini akan digunakan dalam proses melalui 48 iterasi untuk menghasilkan 48 buah sub kunci SK_i sebagai berikut :

$$SK_i = K1_i = K4_{i-1} \text{ xor } g_i(K1_{i-1}, K3_{i-1}, K2_{i-1})$$

$$K4_i = K3_{i-1}$$

$$K3_i = K2_{i-1}$$

$$K2_i = K1_{i-1}$$

Dimana $g_i(K1, K3, K2) = f(K1+K3+(\Delta * I), K2)$

$$\Delta = [(\text{sqrt}(5)-1) * 2^{63}] = 9E3779B97F4A7C15_{16}$$

Tiga proses dari penjadwalan kunci sangat dibutuhkan untuk menghasilkan tiga sub kunci untuk setiap iterasi dari penghitungan data. Jadi jumlah keseluruhannya 48 iterasi yang dibutuhkan untuk penjadwalan kunci.

Proses deskripsi sama dengan enkripsi yaitu menggunakan sub kunci untuk membalikannya yaitu dengan cara membalikan sub kunci tersebut yaitu SK_{3i-2} dan SK_{3i} .

3.2.2. Fungsi G

Fungsi g yaitu $g(K1, K3, K2)$ digunakan untuk proses penjadwalan kunci, sangat jelas bahwa tidak adanya keterhubungan antara bit sub kunci,

tergantung pada cara kompleks pada semua masukan kunci bit, sehingga jelas tidak ada kunci yang lemah. Fungsi g yaitu :

$$g_i(K1, K3, K2) = f(K1+K3+(\Delta * I), K2)$$

$$\Delta = [(\text{sqrt}(5)-1) * 2^{63}] = 9E3779B97F4A7C15_{16}$$

3.2.3. Fungsi f(A,B)

Fungsi f(A,B) non linear menerima dua buah 64 bit masukan nilai A dan B, memprosesnya dengan menggunakan dua buah S box, untuk menghasilkan keluaran 64 bit. Tiga buah permutasi digunakan untuk memastikan avalanche yang maximal pada fungsi tersebut untuk semua bit. Fungsi f(A,B) sebagai berikut :

$$F(A,B) = Sb(P(Sa(E(KP(A,B))))), B)$$

- **Keyed Permutation KP (A,B)**

Permutasi Kunci yang sederhana yang membagi dua masukan A sebesar 64 bit menjadi 32 bit dan menggunakan 32 bit yang paling kanan dari masukan B untuk menentukan untuk menukar pasangan bit yang cocok yaitu bit 1 atau bit 0 mirip dengan yang digunakan pada ICE.

- **Expansion E()**

Fungsi perluasan yaitu dengan menggunakan memilih bagian berada diluar batas yaitu 13 bit (S1) atau 11 bit (S2) jadi setidaknya ada beberapa bit yang mempengaruhi dua buah S-Box secara serempak. E menghasilkan keluaran 96 bit dari masukan 64 bit sebagai berikut:

$$[4-0, 63-56 | 58-48 | 52-40 | 42-32 | 34-24 | 28-16 | 18-8 | 12-0]$$

- **S-Box**

Merupakan suatu table substitusi yang digunakan pada kebanyakan algoritma block cipher. S-box pertama kali digunakan pada Lucifer, kemudian DES dan setelah itu banyak algoritma menggunakan kunci Yaitu $Sa() = [S1, S2, S1, S2, S2, S1, S2, S1]$ dan $Sb() = [S2, S2, S1, S1, S2, S2, S1, S1]$.

3.3. S Box

S Box yang digunakan pada LOKI97 adalah menggunakan pangkat tiga pada galois field GF (2ⁿ) dengan n bilangan ganjil. Pada saat penggunaan masukkan dengan bilangan 11 bit. Ini merupakan suatu alternative untuk mengkombinasikan proses masukkan blok dengan ukuran yang genap. Fungsi S box sebagai berikut :

$$S1[x] = ((x \text{ xor } 1FFF)^3 \text{ mod } 2911) \& FF, \text{ in GF } (2^{13})$$

$$S2[x] = ((x \text{ xor } 7FF)^3 \text{ mod } AA7) \& FF, \text{ in GF } (2^{11})$$

3.4. Algoritma Deskripsi

Untuk proses deskripsi yaitu dengan membagi ciphertest dengan 2 buah 64 bit data yaitu: [R₁₆|L₁₆]

Setelah membagi ciphertext menjadi dua maka melakukan proses kebalikan dari proses enkripsi sebanyak 16 kali iterasi. Maka diperoleh hasil deskripsi sebesar 128 bit plaintext, nilainya yaitu :

$$L_{i-1} = R_i \text{ xor } f(L_i - SK_{3i}, SK_{3i-1})$$

$$R_{i-1} = L_i - SK_{3i} - SK_{3i-2}$$

$$[L_0 | R_0]$$

Hasilnya akan sama dengan menampilkan perhitungan enkripsi dengan sub kunci yaitu menggunakan perintah kebalikannya dengan menggunakan SK_{3i}, SK_{3i-2}.

3.5. Sub Kunci

Sub kunci untuk LOKI97 dibangkitkan dengan memodifikasi cipher itu sendiri. Masukkan ke sub kunci tersusun dari 4 buah 64 bit, yaitu K4, K3, K2 dan K1. Untuk penggunaan kunci 256 bit, maka 64 bit yang pertama menjadi K4, 64 bit yang kedua menjadi K3 dan seterusnya.

Kunci 192 bit diubah menjadi kunci 256 bit dengan cara kunci yang diberikan pertama yaitu 192 bit menjadi bagian dari kunci 256 bit yang akan digunakan dan 64 bit sisanya akan diperhitungkan dengan menggunakan fungsi f LOKI97, dengan 64 bit yang pertama sebagai masukan A (masukan sebelah kanan) dan 64 bit yang kedua sebagai masukan B (masukan sub kunci).

Kunci 128 bit diubah menjadi kunci 256 bit dengan cara kunci yang diberikan pertama yaitu 128 bit menjadi bagian kunci 256 bit yang akan digunakan dan bagian ketiga dari 64 bit (K2) adalah fungsi f dari 64 bit yang kedua disebut masukan A dan 64 bit yang pertama disebut masukan B, dan 64 bit yang keempat (K1) adalah penghitungan fungsi f dengan 64 bit yang pertama sebagai masukan A dan 64 bit yang kedua adalah sebagai masukan B, sama seperti kasus 192 bit.

Setelah masukan kunci dalam bentuk 256 bit maka K4, K3, K2 dan K1 dan 48 sub kunci digunakan untuk penghitungan sebagai berikut fungsi f LOKI97 menghitung masukan A sebagai K1 ditambah K3 ditambah dengan bilangan heksadesimal yaitu 9E3779B97F4A7C15 dikalikan dengan dari sejumlah pembangkitan sub kunci dengan iterasi dari 1 sampai 48, dan masukkan B sebagai K2.

4. Unjuk Kerja Algoritma Kriptografi LOKI97

4.1. Kajian Secara Teoritis

Pada sub bab ini dilakukan analisis terhadap LOKI97 ditinjau dari kajian teoritis. Analisa dilakukan berdasarkan pemahaman yang diperoleh setelah mempelajari algoritma LOKI97.

Struktur Cipher

Metoda yang digunakan pada algoritma LOKI97 ini adalah “*Feistel Network*”. Mekanisme jaringan feistel adalah blok penyandian dibagi menjadi dua bagian yang sama besar : R dan L. Kita dapat menentukan hasil dari suatu iterasi dengan menggunakan hasil iterasi sebelumnya :

$$R_i = L_{i-1} \text{ xor } f(R_{i-1} + SK_{3i-2}, SK_{3i-1})$$

$$L_i = R_{i-1} + SK_{3i-2} + SK_{3i}$$

K_i adalah sub-kunci yang digunakan pada iterasi ke-I dan f adalah suatu fungsi sembarang. Hal ini menyebabkan tidak diperlukannya lagi proses tambahan untuk proses deskripsi.

Banyak Round

Penentuan banyak round didasarkan atas azas keseimbangan. Dimana jika jumlah round sedikit akan menyebabkan cipher menjadi mudah untuk dipecahkan. Sedangkan jika jumlah iterasi semakin banyak, akan menyebabkan kecepatan proses enkripsi/deskripsi akan semakin berkurang. Sebenarnya jumlah round sebanyak 16 cukup beralasan [BP98] untuk memberikan perlindungan keamanan yang tinggi terhadap LOKI97.

Penjadwalan Kunci

Penjadwalan kunci pada LOKI97 menggunakan fungsi $f(A,B)$, dimana fungsi $f(A,B)$ merupakan inti dari LOKI97.

Perubahan Besar Arsip

Arsip ciphertext mempunyai ukuran yang lebih besar dari arsip plaintext. Hal ini terjadi karena adanya proses padding.

Pada mode ECB dan CBC, perubahan maksimum besar arsip ciphertext adalah sebesar satu blok penyandian data (16 byte).

4.2. *Avalanche Effect*

Salah satu karakteristik untuk menentukan baik atau tidaknya suatu algoritma kriptografi adalah dengan melihat avalanche effect-nya. Perubahan yang kecil pada plaintext maupun key akan menyebabkan perubahan yang signifikan terhadap ciphertext yang dihasilkan. Atau dengan kata lain, perubahan satu bit pada plaintext maupun key akan menghasilkan perubahan banyak bit pada ciphertext. Jika perubahan bit yang terjadi adalah hampir separuh dari besar ciphertext maka akan semakin sulit bagi kriptanalis untuk dapat melakukan kriptanalisis.

LOKI97 memperlihatkan sebuah avalanche effect yang baik sekali. Hal tersebut ditunjukkan dari hasil yang diperlihatkan di bawah ini :

Perubahan plaintext 1 bit ditunjukkan di bawah ini :

Tabel Avalanche Effect perubahan plaintext 1 bit

Plaintext 1 : 00000000000000000000000000000000 (dalam heksadesimal)

Plaintext 2 : 80000000000000000000000000000000 (dalam heksadesimal)

Dengan kunci yang digunakan adalah :

Key : 00000000000000000000000000000000 (dalam heksadesimal)

Ciphertext yang dihasilkan adalah :

Ciphertext1: DBAD348CF30BBF8E64B5E5D3065D6898 (dalam heksadesimal)

Ciphertext2: D2734057410AE10710C4922DCC9B34FA (dalam heksadesimal)

Perbedaan bit pada ciphertext 1 dengan ciphertext 2 adalah sebanyak 67 bit lebih dari separuh besar blok (128 bit) sekitar 52%.

Dari percobaan-percobaan diatas maka dapat dirata-ratakan persentase Avallanche effect untuk perubahan plaintext pada bit-bit tertentu yaitu sebesar 49,2%.

Sedangkan untuk perubahan kunci 1 bit ditunjukkan dari hasil dibawah ini.

Tabel Avallanche Effect perubahan kunci 1 bit

Plaintext : 00000000000000000000000000000000 (dalam heksadesimal)

Dengan kunci yang digunakan adalah :

Key1 : 00000000000000000000000000000000 (dalam heksadesimal)

Key2 : 80000000000000000000000000000000 (dalam heksadesimal)

Chipertext yang dihasilkan adalah :

Chipertext1:DBAD348CF30BBF8E64B5E5D3065D6898(dalam heksadesima)

Chipertext2:48F00DF8C90822417D12ECAD682B014 (dalam heksadesimal)

Perbedaan bit pada chipertext 1 dengan chipertext 2 adalah sebanyak 72 bit atau sekitar 56% dari besar blok (128 bit).

4.3. Kunci Lemah dan Kunci Setengah Lemah

Pada algoritma LOKI97 belum ditemukan adanya kunci setengah lemah ataupun kunci lemah. Semua kunci yang digunakan mempunyai kekuatan yang sama dan dapat digunkana untuk enkripsi ganda (kunci lemah mempunyai sifat yaitu jika sebuah plaintext dienkripsi ganda menggunakan kunci lemah akan menghasilkan plaintext itu sendiri).

Beberapa kunci yang bernilai ekstrim (termasuk empat buah kunci lemah yang dimiliki algoritma DES) tidak berpengaruh pada algoritma LOKI97. Kunci setengah lemah adalah sepasang kunci yang mempunyai sifat yaitu jika sebuah plaintext dienkripsi dengan suatu kunci dan sideskripsi kembali dengan kunci pasangannya akan menghasilkan plaintext itu kembali.

Beberapa pasangan kunci yang bernilai ekstrim (termasuk enam pasang kunci setengah lemah yang dimiliki oleh algoritma DES) tidak berpengaruh pada algoritma LOKI97.

4.4. Kelemahan Algoritma LOKI97

Algoritma LOKI97 mempunyai dua kelemahan utama yaitu :

1. Mempunyai karakteristik iterasi dua-putaran (two-round) dengan probabilitas 2^{-8}
2. Fungsi f algoritma LOKI97 tidak balance, sebagai konsekuensinya untuk nilai tertentu dari round key, ada relasi iterasi dua-putaran (two-round) linier dengan bias 2^{-4}

5. Kesimpulan

Algoritma LOKI97 memiliki avalanche affect yang cukup baik, hal ini dapat ditunjukkan pada blok plaintext akan dihasilkan nilai avalanche effect sekitar 50% dari besar satu blok dan pada kunci dihasilkan nilai avalanche effect sekitar 50% dari besar satu blok.

Kelemahan yang utama dari algoritma LOKI97 adalah Mempunyai karakteristik iterasi dua-putaran (two-round) dengan probabilitas 2^{-8} dan Fungsi f algoritma LOKI97 tidak balance.

Referensi

- School of Computer Science, “Introducing the new LOKI97 Block Cipher” June 1998.
www.mirrors.wiretapped.net/security/cryptography/algorithms/loki97/
- Budi Raharjo, “Keamanan Sistem Informasi Berbasis Internet” PT Insan Infonesia – Bandung & PT INDOCISC – Jakarta, 2002.
<http://budi.insan.co.id/courses/el695/handbook.pdf>
- V Rijmen & LR Knudsen, “Weaknesses in LOKI97”, June 1998.
www.unsw.adfa.edu.au/~lpb/research/loki97/ – 4k –
- Diki Alfian, “Enkripsi Data Dengan Algoritma Kriptografi Kunci Simetris Menggunakan Metoda LOKI97”, STT Telkom, 2001.