Algoritma SAFER K-64 dan Keamanannya

Andi Setiawan – NIM : 13506080

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if16080@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang algoritma SAFER K-64 dan permasalahan-permasalahan keamanannya. SAFER(*Secure and Fast Encryption Routine*) adalah sebuah algoritma block cipher yang menjadi salah satu nominasi AES (*Advanced Encryption Standard*). Algoritma ini menggunakan kunci sepanjang 64 bit, dan panjang blok 64 bit juga. Algoritma ini menerapkan proses enkripsi berkali-kali, normalnya sebuah teks akan mengalami perputaran enkripsi sebanyak enam ronde dan tiap ronde akan menggunakan kunci berbeda yang dibangkitkan dari kunci eksternal. Algoritma ini hanya menggunakan fungsi pergeseran, xor, penambahan, dan pengurangan bit dan fungsi matematika untuk memetakan bit, tanpa melibatkan jaringan feistel. Namun algoritma ini sudah dapat memenuhi prinsip desain *confussion* dan *diffusion*.

Pada algoritma ini terdapat transformasi linear yang tidak lazim (*unorthodox linear transform*) yang disebut dengan *Pseudo-HadamardTransform* (PHT). Selain itu, pada proses pembuatan kunci pada tiap ronde digunakan sebuah fungsi yang dapat menyebabkan kunci pada tiap ronde tidak menghasilkan kunci lemah (*weak key*).

Makalah ini juga akan membahas kelemahan algoritma SAFER dalam membuat jadwal kuncinya. Terutama saat menghadapi serangan Related-Key Choosen *Plaintext* Attack. Selanjutnya, makalah ini akan membahas improvisasi yang dapat dilakukan untuk memperbaiki keamanan dari algoritma ini.

***Kata kunci**: Safer K-64, Advanced Encryption Standard*, *enkripsi, dekripsi, keamanan.*

1. Pendahuluan

SAFER adalah sebuah singkatan dari *Secure and Fast Encryption Routine*. SAFER K-64 adalah algoritma SAFER yang pertama kali dipublikasikan pada tahun 1993 oleh James L. Massey yang bernaung pada *Cylink Corporation* (Sunnyvale, USA)..

SAFER adalah algoritma *block cipher* dengan panjang bloknya adalah 64 bit. Algoritma ini tidak menerapkan jaringan feistel, dan hanya menggunakan pergeseran, penjumlahan, pengurangan bit, dan sejumlah fungsi matematika seperti modulo, logaritma, dan eksponensial untuk memetakan *byte* ke *byte* lain. Algoritma ini adalah algoritma *iterated cipher* dimana proses enkripsinya dilakukan dengan iterasi dapat dilakukan sebanyak enam kali atau lebih. K-64 memiliki sebuah arti, yaitu panjang kunci yang digunakan adalah 64 bit. Untuk memperbaiki keamanannya, kemudan panjang kunci dari algoritma ini kemudian dikembangkan menjadi 128 bit, namun Lars Knudsen dan Sean Murphy masih menemukan kekurangan dari penjadwalan kunci algoritma ini. Itulah sebabnya algoritma ini kemudian berkembang menjadi algoritma SAFER SK. SK disini berarti *Strengthened Key Schedule* (penjadwalan kunci yang diperkuat), dengan membuat fungsi penjadwalan kunci yang lebih baik. Untuk memperbaiki kekurangan-kekurangannya, algoritma ini kemudian berkembang menjadi banyak versi seperti SAFER K-128, SAFER SK-64, SAFER SK-128, SAFER+, SAFER++, dan lain sejenisnya.

Ada dua konsep kriptografi baru yang diperkenalkan algoritma ini yaitu:

1. Transformasi linear yang tidak biasa
   Yaitu dengan menggunakan *Pseudo-Hadamard Transform (PHT).* Fungsi transformasi ini membuat cipher yang dihasilkan memenuhi sifat diffusion. Dengan PHT, meskipun pada teks asli, perubahan karakternya kecil, namun hasil enkripsinya dapat berbeda jauh. Dengan statistik, telah dapat dibuktikan bahwa algoritma SAFER adalah

algoritma yang paling dapat menjamin prinsip diffusion ini dibandingkan algoritma lainnya.

2. Penambahan fungsi untuk membuat kunci tampak random. Hal ini menyebabkan algoritma pada SAFER tidak memiliki kunci lemah (*weak key*). Penjadwalan kunci algoritma SAFER memastikan bahwa kunci yang dihasilkan tiap ronde bukanlah kunci lemah.

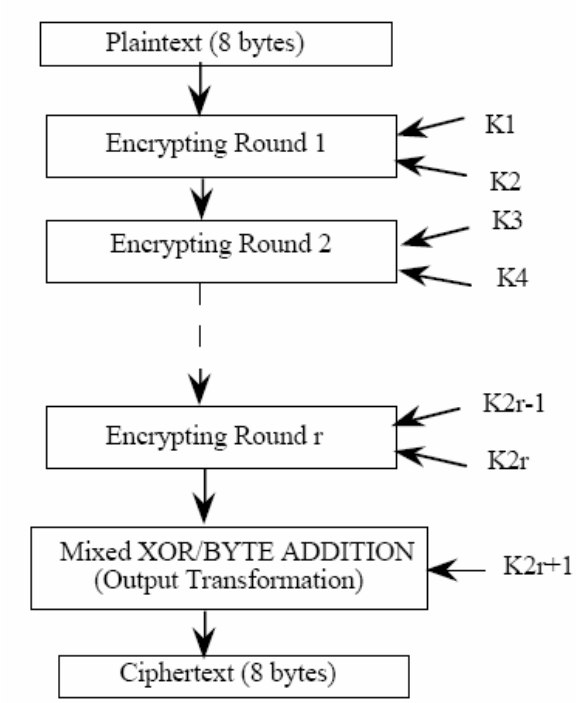
Desain dari algoritma SAFER ini telah menerapkan kedua prinsip dari Shannon, yaitu *confussion* dan *diffusion*. Hasil dari enkripsi algoritma adalah ciphertext yang acak, yang sulit dilacak dengan metode statistik. Selain itu, penerapan transformasi linear yang tidak biasa, menyebabkan hasil *ciphertext*-nya sesuai dengan prinsip *diffusion* Shannon. Karena, tidak ada pola antara satu teks dengan teks yang lainnya.

Permasalahan keamanan utama dalam algoritma ini adalah lemahnya penjadwalan kunci. Hal ini dibuktikan dengan melakukan serangan *related key on choosen plaintext* yang akan dibahas pada makalah ini selanjutnya.

2. Algoritma Enkripsi dari SAFER

Algoritma enkripsi SAFER terdiri dari r-ronde. Setiap blok akan mengalami fungsi transformasi yang sama pada tiap putaran sebelum masuk ke fungsi transformasi output dan menghasilkan ciphertext. Sangat direkomendasikan, jumlah ronde untuk tiap proses enkripsi pada sebuah blok adalah enam ronde. Tapi, jika proses enkripsi ingin dilakukan lebih dari sepuluh ronde, hal ini juga sangat bagus. Setiap ronde membutuhkan dua buah *subkey* yang dibangkitkan dari kunci eksternal yang dimasukan user. Maka untuk ronde berjumlah r, akan dibangkitkan sebanyak 2r+1 buah kunci yang akan dibahas pada bab pembangkitan kunci nantinya. Panjang kunci masukan user biasanya delapan *byte*. Dan panjang semua *plaintext*, chipertext, dan subkey yang terlibat pada setiap proses enkripsi adalah sama yaitu 8 *byte* (64 bit).

Untuk menjelaskan prosesnya, dapat kita lihat pada gambar 1. Sebuah blok *plaintext* sepanjang delapan *byte*. Kemudian mengalami transformasi dengan memasuki proses enkripsi tiap ronde. Hingga ronde ke r. Lalu memasuki tahap Mixed Xor/*Byte* addition. Pada tahap terakhir ini, *byte* ke-1, 5, 4, dan 8 mengalami Xor dengan *byte* yang berkorespondensi pada *subkey* K2r+1. Sedangkan *byte* ke-2, 3, 6, dan 7 mengalami penjumlahan modulo 256 dengan *byte* yang berkorespondensi pada *subkey* K2r+1. Lalu setelah melewati tahap tersebut, maka akan diperoleh hasil ciphertext pada ronde tersebut. Perlu diperhatikan bahwa, pada tiap ronde enkripsi membutuhkan dua buah *subkey*, hal ini akan dijelaskan kemudian pada fungsi enkripsi.



gambar 1. Proses Enkripsi SAFER

2.1 *Fungsi Enkripsi*

Fungsi Enkripsi pada algoritma SAFER pertama kali akan menggunakan operasi Xor, add. Memasuki proses transformasi Mixed Xor/*byte* operation and addition.

Operasi Xor adalah ekivalen dengan operasi penjumlahan bit bermodulo dua. Contohnya adalah:

0 Xor 1 = (0+1) mod 2 = 1
1 Xor 1 = (1+1) mod 2 = 0
0 Xor 0 = (0+0) mod 2 = 0

Sedangkan operasi add adalah operasi penjumlahan *byte* dengan modulo 256. Perlu diperhatikan bahwa representasi *byte* di sini

adalah dalam bentuk integer. Yang berarti nilainya ada pada wilayah 0 hingga $2^8 = 255$. Itulah sebabnya mengapa penjumlahannya dimodulo dengan 256. Contohnya adalah:

250 add 110=(250+116)mod 256=110
11 add 25 = (11+25) mod 256 = 36

Fungsi enkripsi ini juga terdapat simbol $45^{(.)}$ dan $\log_{45.}$ Maksud dari $45^{(.)}$ adalah apabila fungsi tersebut menerima input integer j, yang berisi data *byte* dari input. Maka fungsi tersebut akan mengubah nilai j menjadi $((45^{(j)} \bmod 257) \bmod 256)$. Nilai output akan bernilai 0 bila j = 128.

Sedangkan $\log_{45}$ berarti fungsi itu akan merubah nilai input integer j, yang berisi informasi *byte* input, menjadi $\log_{45}$ (j). Nilai $\log_{45}$ (j) akan bernilai 128 jika nilai j adalah 0.

Fungsi $\log_{45}$ dan fungsi $45^{(.)}$ sebenarnya adalah fungsi pemetaan *byte*. Dan fungsi $\log_{45}$ adalah fungsi kebalikan dari fungsi $45^{(.)}$. Dengan kata lain, kita dapat memperoleh sebuah nilai j kembali setelah j diubah kedua fungsi.

$$j = \log_{45} (45^j)$$

Implementasi kedua fungsi tersebut dalam program adalah dengan membuat dua buah tabel, yaitu tabel logaritma(logtab) dan tabel eksponensial(exptab). Dimana nilai dari masing-masing tabel adalah:

```
logtab[1]       = 0;
exptab[0]       = 1;
exptab[128]     = 0;
logtab[0]       = 128;
exptab[0]       = 1;
```

dan untuk nilai tabel lain kecuali yang telah terisi di atas adalah:

```
exptab[i] = (45 *exptab[i-1]) mod 257;
logtab[ exptab[i] ] = i;
```

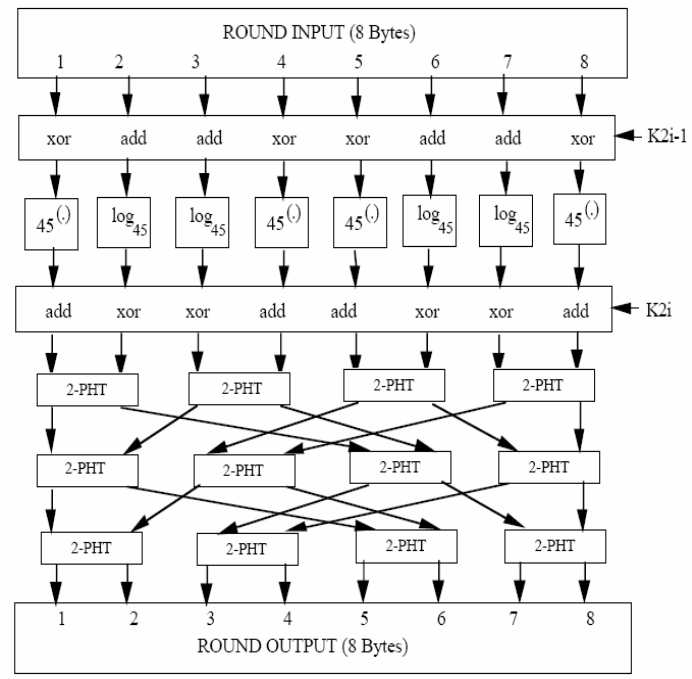
Dengan sifat invers dari kedua fungsi ini, proses dekripsi akan dapat dilakukan dengan mudah nantinya.

Hal menarik lainnya yang terdapat pada fungsi enkripsi SAFER adalah PHT (Pseudo-Hadamard Transform). Fungsi ini menerima dua buah integer dari *byte* input (a_1, a_2) dan menghasilkan dua buah integer dari *byte* output (b_1, b_2). Dimana fungsi transformasinya adalah:

$$b_1 = 2a_1 + a_2$$
$$b_2 = a_1 + a_2$$

Penjumlahan *byte* ini, juga merupakan penjumlahan bermodulo 256.

Jadi proses perubahan isi *plaintext* dari awal sampai akhir dapat kita lihat pada gambar 2. Fungsi enkripsi menerima blok masukan yang panjangnya delapan *byte*. Setiap *byte* masukan ini kemudian mengalami operasi xor dan add secara acak. Pada operasi xor dan add pertama yang melibatkan kunci K2i-1, *byte* ke-1, 4, 5,dan 8 mengalami Xor dengan *byte* yang berkorespondensi di K2i-1, sedangkan *byte* sisanya mengalami operasi add dengan modulo 256 dengan *byte* yang berkorespondensi pada K2i-1, dimana i adalah urutan ronde fungsi enkripsi. Setelah melewati fungsi xor dan add *byte* secara acak yang pertama, kemudian masing-masing *byte* yang sebelumnya mengalami xor, dimasukan pada fungsi $45^{(.)}$. Sedangkan *byte* yang mengalami operasi add akan dimasukan ke fungsi $\log_{45}$.



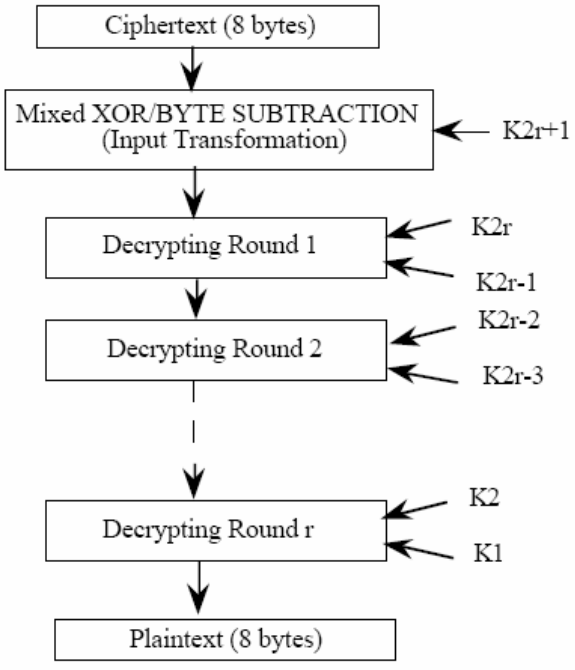
gambar 2. Diagram Proses Transformasi pada Fungsi Enkripsi

Setelah melewati fungsi $45^{(.)}$ dan $\log_{45,}$ *byte* tersebut kembali mengalami operasi add dan xor secara acak, namun kali ini operasinya berganti dibanding operasi yang dilakukan pada operasi

yang pertama. Kali ini *byte* ke-1, 4, 5, 8 yang mengalami operasi add bermodulo 256 dan sisanya mengalami operasi Xor dengan *byte* yang berkorespondensi pada K2i, dimana i adalah urutan ronde fungsi enkripsi tersebut. Setelah keluar dari ketiga lapis transformasi linear, masing-masing *byte* kemudian akan masuk ke fungsi PHT. Pada gambar tertulis 2-PHT untuk menunjukkan dua poin PHT, yaitu PHT yang menerima input dua poin. Urutan pengacakan adalah seperti yang tertera pada gambar, arah panah menunjukkan bahwa hasil dari fungsi PHT kemudian digunakan untuk fungsi PHT untuk yang kedua dan yang ketiga sesuai dengan arah panah dari output masing-masing lapisan. Pengacakan inilah yang menyebabkan cipher dari fungsi enkripsi ini benar-benar menerapkan prinsip diffusion. Setelah melewati tiga lapis PHT, maka hasil ciphertext telah diperoleh.

Dari gambaran fungsi enkripsi ini, sudah jelas bahwa tiap ronde enkripsi membutuhkan dua buah subkey untuk operasi add dan xor secara acak. Dengan dua buah kunci ini, hasil ciphernyamenjadi semakin teracak, dan sulit ditebak.

3 Algoritma Dekripsi SAFER

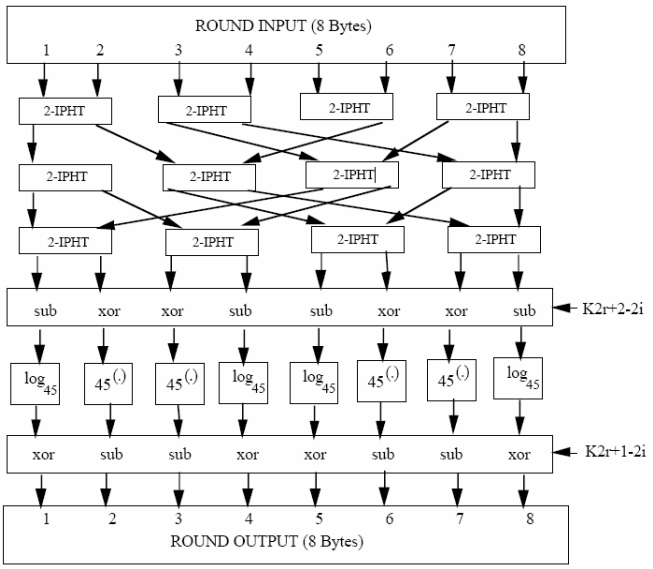


gambar 3. Proses Dekripsi Ciphertext pada Algoritma SAFER

Algoritma dekripsi pada SAFER adalah kebalikan dari algoritma enkripsinya. Masukannya adalah delapan *byte* ciphertext. Text ini lalu mengalami input transformation dalam hal ini, ada pada *byte* ke- 1, 4, 5, 8 mengalami substraction, dan sisanya mengalami operasi Xor. Operasi substraction ini adalah operasi pengurangan interger dari *byte* dan akan ditambah 256 jika nilainya minus.

Setelah mengalami operasi tersebut. Teks memasuki fungsi dekripsi dari ronde ke-1 hingga ronde ke-i, dengan i adalah jumlah ronde pada algoritma SAFER yang mengenkripsi teks. Kunci masukannya pada algoritma dekripsi ini urutannya terbalik dari urutan kunci pada algoritma enkripsi. Setelah menyelesaikan fungsi dekripsi ke i, maka hasil *plaintext* semula telah didapat.

3.1 *Fungsi Dekripsi*



gambar 4. Fungsi Dekripsi pada SAFER

Fungsi dekripsi dari SAFER adalah fungsi kebalikan dari fungsi enkripsi. Terbalik secara urutannya dan fungsinya, terutama pada fungsi PHT. Sedangkan fungsi lainnya tidak berubah fungsinya. Urutan dekripsi dimulai dari invers PHT sebanyak tiga lapis. Fungsi invers PHT menerima input dua buah integer dari *byte* input (b_1,b_2) dan mengeluarkan hasil dua buah integer dari *byte* output (a_1,a_2) Fungsi invers PHT ini adalah sebagai berikut:

$$a_1 = b_1 - b_2$$

$a_2 = -b_1 + 2b_2$.

Masukan *byte* adalah sesuai urutan panah pada gambar. Urutan *byte* harus benar, agar menghasilkan teks yang sama seperti sebelum proses enkripsi.

Setelah melewati tiga lapis IPHT, masing-masing *byte* lalu memasuki operasi subs dan add secara acak. Seperti yang terlihat pada gambar, *byte* ke-1, 4, 5, 8 akan mengalami operasi subs terlebih dahulu. Dan sisanya akan mengalami operasi Xor dengan setiap *byte* yang berkorespondensi pada K2r+2-2i dengan r adalah jumlah ronde dan i adalah urutan ronde fungsi dekripsi. Setelah melewati lapisan operasi subs dan xor secara acak yang pertama, masing-masing *byte* kemudian akan dimasukkan pada fungsi $\log_{45}$ dan $45^{(.)}$. Dimana *byte* yang mengalami operasi subs sebelumnya, sekarang dimasukan pada fungsi $\log_{45}$ dan *byte* yang mengalami operasi Xor sebelumnya, sekarang dimasukan ke fungsi $45^{(.)}$. Setelah itu, setiap *byte* mengalami operasi subs dan xor secara acak kembali, dimana kali ini *byte* ke-1, 4, 5, 8 mengalami operasi xor dengan setiap *byte* yang berkorespondensi pada K2r+1-2i dengan r adalah jumlah ronde dan i adalah urutan ronde fungsi dekripsi, dan setiap *byte* sisanya mengalami operasi subs.

Demikianlah algoritma dekripsi dari SAFER, sangat sederhana, karena hanya melibatkan operasi aritmatika dan manipulasi bit.

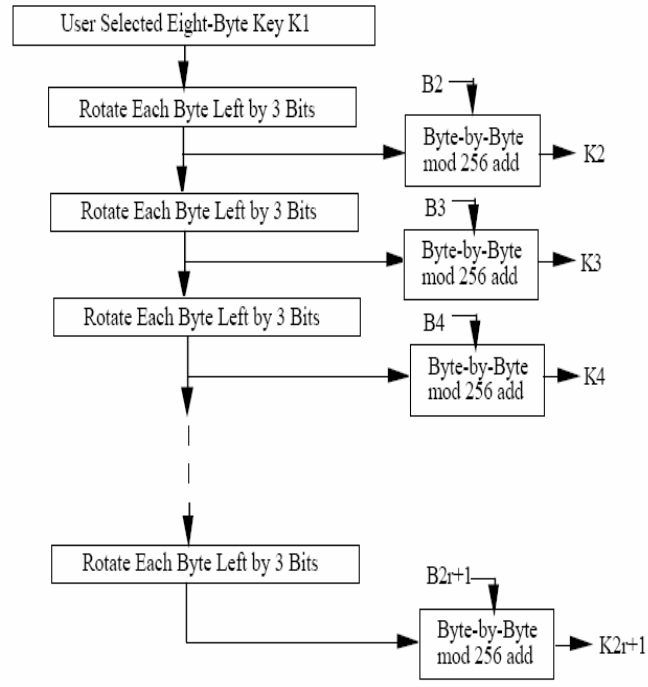
4. Penjadwalan Kunci pada SAFER

Penjadwalan kunci (*key scheduler*) adalah sebuah mekanisme untuk membuat subkey pada algoritma SAFER, mulai dari K2, K3, K4, …, K2r+1. Dari masukan kunci K1 yang dimasukan oleh pengguna. Pada penjadwalan kunci ini, terdapat sebuah fungsi B yang menghasilkan sebuah kunci pengacak(*bias key*) yang dapat memastikan bahwa setiap *byte* pada kunci-kunci berikutnya telihat "acak" dan tampak tidak ada pengaruh sama sekali dari kunci yang sebelumnya. Fungsi tersebut adalah sebagai berikut:

b[i,j] = 45**[45**(9i+j) mod 257] mod 257

Dimana *byte* ke-j dari kunci bias ke-i. Hal ini berarti bahwa, setiap *byte* pada kunci bias akan tampak random dibanding *byte* sebelumnya, tidak tampak ada keterkaitan satu sama lain. Hal inilah yang membuat algoritma SAFER dapat menerapkan prinsip diffusion.

Secara keseluruhan, proses pembuatan kunci pada tiap ronde tampak seperti yang ada pada gambar. Masukan pertama adalah K1 yaitu kunci masukan dari user. Kemudian kunci tersebut mengalami perputaran tiga bit ke kiri. Hasil ini kemdian akan digunakan untuk membangkitkan kunci pada ronde berikutnya, dan juga digunakan untuk langsung membangkitkan kunci pada ronde tersebut. Sebelum menjadi kunci pada ronde tersebut, hasil perputaran bit itu mengalami operasi add bermodulo 256 dengan kunci bias pada ronde tersebut. Dan hasil penjumlahan itu lalu menghasilkan kunci ronde pada ronde tersebut. Demikian seterusnya, hingga kunci ronde yang dihasilkan sejumlah 2r+1, dimana r adalah jumlah ronde pada algoritma SAFER.



gambar 5. Proses Penjadwalan Kunci Internal

5. Keamanan Algoritma SAFER

Algoritma SAFER memiliki kelemahan pada penjadwalan kuncinya. Beberapa jurnal mengatakan ada kemungkinan bahwa algoritma SAFER dapat dipecahkan dengan metode *Related Key Choosen Plaintext attack*. Metode

ini membantu mempercepat pemecahan algoritma SAFER dibandingkan dengan metode *brute force*. Dengan demikian, meskipun efek yang ditimbulkan oleh metode ini tidak begitu besar, namun hal ini membuktikan bahwa algoritma SAFER tidak sempurna dari segi keamanannya. Dan harus diperbaiki lagi agar dapat berjalan dengan baik.

6. Improvisasi Algoritma SAFER

Algoritma SAFER dapat ditingkatkan keamanannya dengan:

1. Menambahkan jumlah ronde enkripsi.
   Masih ada banyak subkey yang dapat dibangkitkan dari algoritma SAFER, sehingga jumlah ronde tiap fungsi enkripsi dapat ditambah. Secara logika, dengan menambahkan jumlah ronde putaran fungsi enkripsi, dapat menyebabkan hasil ciphertext menjadi lebih sulit ditebak.

2. Merubah cara penjadwalan kunci.
   Seperti yang sudah disebutkan sebelumnya bahwa algoritma SAFER memiliki kelemahan pada penjadwalan kunci. Untuk meningkatkan keamanan dari algoritma SAFER, kita dapat memodifikasi penjadwalan kunci. Sehingga lebih sulit dilacak. Dan hasil ciphertext dapat benar-benar menerapkan prinsip confussion dan diffusion.

7. Kesimpulan

Kesimpulan yang didapat dari studi algortima SAFER K-64 dan keamanannya adalah :

1. Algoritma SAFER K-64 adalah algoritma block cipher dengan panjang blok 64 bit dan panjang kunci 64 bit.
2. Algoritma ini diciptakan oleh James L. Massey pada tahun 1973.
3. Pada umumnya iterasi untuk mengenkripsi pesan pada algoritma SAFER adalah enam ronde. Namun untuk keamanan, dapat juga memilih jumlah ronde yang lebih banyak.
4. Untuk mengenkripsi dan mendekripsi teks, algoritma ini hanya menggunakan fungsi aritmatika seperti modulo, logaritma, dan eksponen. Serta operasi manipulasi bit seperti pergeseran, penjumlahan, Xor, dan pengurangan.
5. Kunci internal dibangkitkan dari kunci eksternal yang dimasukan oleh pengguna. Dengan melakukan modifikasi pergeseran bit. Dan dengan menggunakan kunci bias sehingga hasil kuncinya sangat baik dalam menerapkan prinsip difusi.
6. Keamanan dari algoritma SAFER masih diragukan, karena ada serangan yang mampu menebak kunci dari enkripsi algoritma SAFER yang dipakai yaitu *Related Key Choosen Plaintext Attack.*
7. Peningkatan keamanan algoritma SAFER dapat dilakukan dengan menambah jumlah ronde enkripsi dan juga dengan mendesain penjadwalan kunci yang baru

DAFTAR PUSTAKA

[1] James L. Massey. SAFER K-64: A *Byte*.Oriented Block-Ciphering Algorithm. *https://eprints.kfupm.edu.sa/63253/1/63253.pdf. Tanggal akses: 1 April 2009.*

[2] Lars. R. Knudsen. Key Schedule Weakness in SAFER K-64. http://portal.acm.org/citation.cfm?id=706014. Diakses : 1 April 2009.

[3] Munir, Rinaldi. (2004). Bahan Kuliah IF5054 Kriptografi. Departemen Teknik Informatika, Institut Teknologi Bandung.

[4] Snighda Jain, Shweta Jain. Implementation of SAFER K-64(2004), *http://www.mcs.csuhayward.edu/~pwong/cs6520_sum04/sec2/safer_k64.pdf*. Tanggal akses : 1 April 2008.