Studi Perbandingan International Data Encryption Algorithm (IDEA) dan The Fast Data Encipherment Algorithm (FEAL)

Andara Livia

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, bandung
e-mail: andara.livia@gmail.com

ABSTRAK

Dalam dunia kriptografi, block cipher adalah cipher dengan kunci simetrik yang terdiri dari kumpulan bit-bit dengan panjang tetap, yang disebut sebagai blok. Algoritma enkripsi dengan menggunakan block cipher menerima masukan n-bit block yang berisi plaintext dan memberikan keluaran n-bit block ciphertext. Transformasi yang terjadi selama proses enkripsi dikontrol oleh masukan kedua, yaitu sebuah kunci rahasia, yaitu k-bit kunci K. Sedangkan untuk proses dekripsi berlaku hal yang sama, yaitu menerima masukan berupa n-bit block ciphertext dan kunci rahasia, kemudian mengeluarkan n-bit block plaintext seperti sedia kala.

Beberapa contoh algoritma enkripsi yang menggunakan block cipher adalah Data Encryption Standard (DES, International Data Encryption Algorithm (IDEA), dan The Fast Data Enchiperment Algorithm (FEAL). Makalah ini akan menitik beratkan terhadap algoritma FEAL dan IDEA. Sehingga di dalam makalah ini terdapat pembahasan mengenai algoritma FEAL dan IDEA, dan terdapat pula pembahasan mengenai perbandingan dari kedua algoritma tersebut.

Kata kunci: block cipher, FEAL, IDEA.

1. PENDAHULUAN

Block cipher dengan menggunakan kunci simetrik adalah cipher yang berperan penting dalam sistem kriptografi. Cipher ini merupakan dasar dari pembentukan pseudorandom number generator, stream cipher, MAC, dan hash function. Lebih jauh lagi, cipher ini merupakan komponen utama pada teknik autentikasi message, mekanisme integritas data, entity authentication protocol, dan skema (symmetric-key) digital signature.

Tidak ada satupun block cipher yg sesuai untuk semua aplikasi, bahkan sekalipun cipher tersebut menawarkan keamanan tingkat tinggi. Hal ini merupakan tradeoff yang tak terelakkan yang selalu terjadi pada aplikasi praktis. Sebagai contoh, efisiensi memiliki keamanan sebagai tradeoff-nya, dengan kata lain, bila menginginkan keamanan maka harus mengesampingkan efisiensi dan begitu pula sebaliknya. Oleh karena itu, memiliki banyak kandidat cipher merupakan suatu keuntungan.

Di antara semua kandidat cipher, terdapat beberapa cipher yang cukup menarik perhatian. Di antaranya yaitu DES, algoritma yang sangat popular digunakan, FEAL yang telah menerima dukungan komersial yang serius dan telah banyak dianalisis secara independen, dan IDEA yang pada awalnya ditujukan untuk menggantikan DES.

Data Encryption Standard (DES) merupakan block cipher yang berbasis algoritma kunci simetrik. DES menggunakan kunci dengan panjang 56-bit. Saat ini DES sudah dianggap tidak aman. Hal ini terkait dengan kunci 56-bit yang digunakan DES dianggap memiliki ukuran yang terlalu kecil. Pada bulan Januari 1999, distributed.net dan Electronic Frontier Foundation bekerja sama untuk memecahkan algoritma DES dalam 22 jam 15 menit.

Terkait dengan keamanan yang telah disebutkan diatas, dan operasi DES yang tergolong lambat saat diimplementasikan pada software, para peneliti pun termotivasi untuk mengajukan sejumlah alternatif desain block cipher. Rancangan-rancangan ini mulai bermunculan pada akhir 1980 dan awal 1990. Sebagai contoh dari alternatif rancangan tersebut antara lain adalah IDEA dan FEAL. Pada bab selanjutnya akan dibahas lebih mendalam mengenai kedua algoritma tersebut.

2. FEAL

Fast Data Encipherment Algorithm (FEAL) adalah block cipher yang dibuat dengan tujuan sebagai alternatif dari Data Encryption Standard (DES) dan didesain agar dapat beroperasi dengan lebih cepat pada perangkat lunak. FEAL termasuk kedalam keluarga algoritma yang memainkan peranan penting dalam perkembangan dan perbaikan dari berbagai macam teknik kriptanalistik, termasuk kriptanalisis linear dan differential. Sama halnya seperti DES, FEAL berbasiskan algoritma Feistel. FEAL pertama kali dipublikasikan pada tahun 1987 oleh Akihiro

Shmizu dan Shoji Miyaguchi dari NTT. Cipher ini rentan terhadap berbagai bentuk kriptanalisis, dan telah berperan sebagai katalis pada penemuan kriptanalisis differential dan linear.

Terdapat sejumlah revisi yang berbeda dari FEAL, meskipun semuanya merupakan cipher Feistel, dan menggunaka dasar fungsi putaran yang sama dan beroperasi pada blok 64-bit. FEAL-N memetakan 64-bit block plaintext menjadi 64-bit block ciphertext dengan menggunakan 64-bit kunci rahasia. Sama halnya dengan DES, FEIL menggunakan N putaran cipher Feistel, namun dengan fungsi-*f* yang jauh lebih sederhana, dan dijumlahkan dengan state awal dan akhir yang melakukan XOR sebagian dari kedua data seperti melakukan XOR subkey langsung dengan sebagian data.

2.1 Proses Enkripsi pada FEAL

Berikut ini akan dijelaskan salah satu algoritma FEAL, yaitu FEAL-8.

- Terdapat fungsi *f f(A,Y)* yang memetakan pasangan masukan 32 x 16 bit menjadi keluaran 32 bit.
- Pada fungsi *f* tersebut terdapat dua byte-oriented data substitusi S0 dan S1 yang masing-masing digunakan dua kali.
- Masing-masing data substitusi tersebut akan memetakan sepasang masukan 8-bit menjadi 8-bit keluaran.
- S_0 dan S_1 menambahkan 1 bit $d \in \{0,1\}$ menjadi 8-bit argument *x* dan *y*, dengan mengabaikan carry dari bit teratas, kemudian hasilnya digeser ke kiri sebanyak 2-bit (ROT2).

$$S(x; y) = ROT\ 2(x + y + d \bmod 256)$$

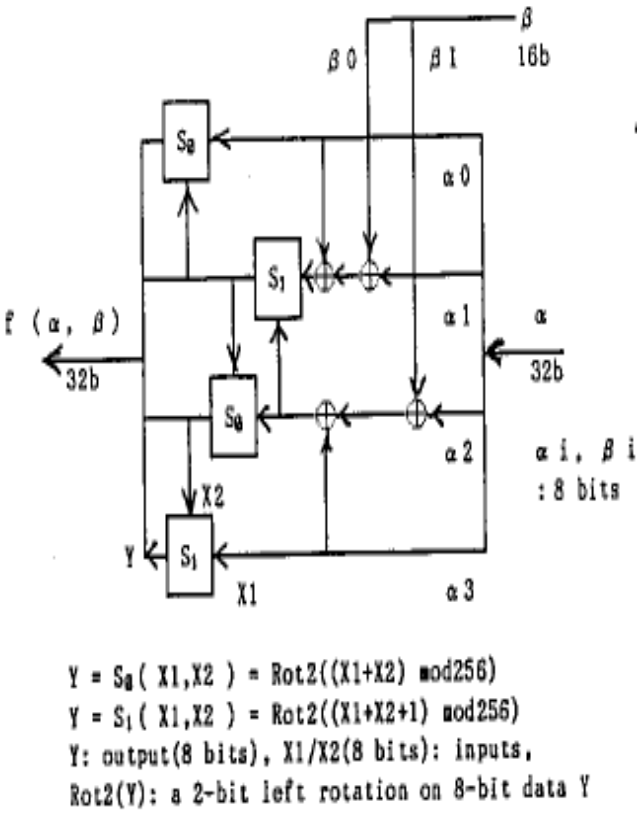
- Terdapat pula fungsi key schedule fK(A,B) yang sama seperti fungsi f, fungsi ini memetakan dua 32-bit input menjadi keluaran 32-bit.
- Pada tabel di bawah ini, A_i, B_i, Y_i, t_i, dan U_i adalah variable berukuran 8-bit.

Table 1

	$U \leftarrow f(A,Y)$	$U \leftarrow f_K(A, B)$
t_1	$(A_o \oplus A_1) \oplus Y_0$	$A_o \oplus A_1$
t_2	$(A_2 \oplus A_3) \oplus Y_1$	$A_2 \oplus A_3$
U_1	$S_1(t_1, t_2)$	$S_1(t_1, t_2 \oplus B_0)$
U_2	$S_0(t_2, U_1)$	$S_0(t_2, U_1 \oplus B_1)$
U_0	$S_0(A_0, U_1)$	$S_0(A_0, U_1 \oplus B_2)$
U_3	$S_1(A_3, U_2)$	$S_1(A_3, U_2 \oplus B_3)$

Langkah-langkah enkripsi pada algoritma FEAL adalah sebagai berikut:

1. Algoritma ini menerima input berupa 64-bit plaintext $M = m_1 \ldots m_{64}$ dan 64-bit kunci $K = k_{1...} k_{64}$.
2. Sedangkan keluarannya adalah 64-bit block ciphertext $C = c_1 \ldots c_{64}$
3. Dengan fungsi *key schedule* yang akan dijelaskan di bawah, ubah 16-bit subkey menjadi K_i dari *K*.
4. Definisikan $M_L = m_1 \ldots m_{32}$ dan $M_R = m_{33} \ldots m_{64}$.
5. Lakukan *XOR* initial subkey:
   $(L_0, R_0) \leftarrow (M_L, M_R) \oplus ((K_8, K_9),(K_{10}, K_{11}))$.
6. $R_0 \leftarrow R_0 \oplus L_0$.
7. Untuk $i = 1$ hingga $i = 8$, lakukan:
   $L_i \leftarrow R_{i-1}$
   $R_i \leftarrow L_{i-1} \oplus f(R_{i-1}, K_{i-1})$
   Gunakan Table 1 untuk $f(A, Y)$ dengan
   $A = R_{i-1} = (A_0, A_1, A_2, A_3)$ dan
   $Y = K_{i-1} = (Y_0, Y_1)$
8. $L_s \leftarrow L_s \oplus R_s$
9. Lakukan XOR final subkey
   $(R_s, L_s) \leftarrow (R_s, L_s) \oplus ((K_{12}, K_{13}),(K_{14}, K_{15}))$
10. $C \leftarrow (R_s, L_s)$

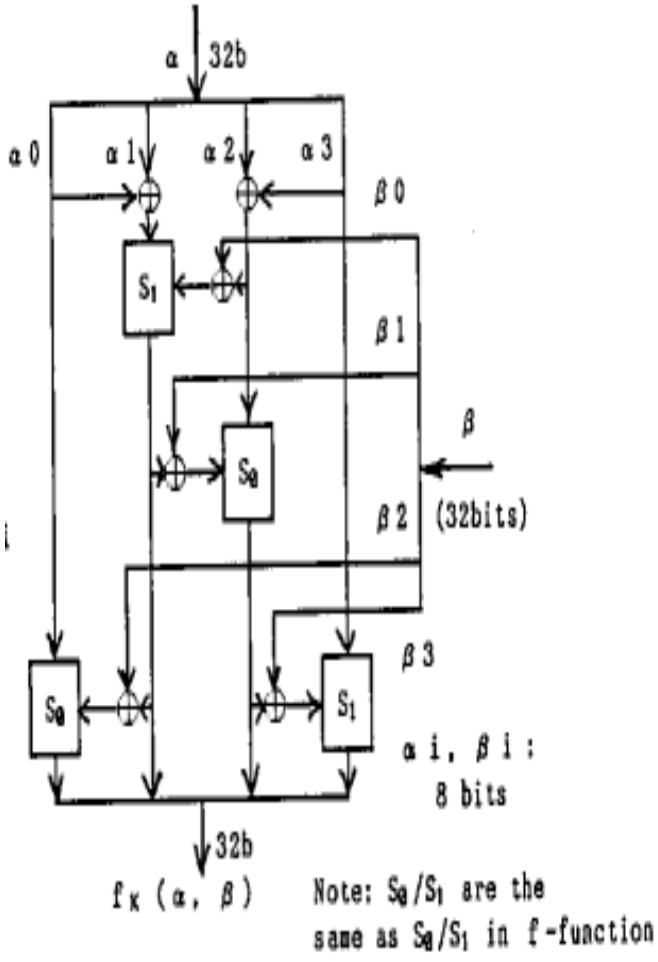


Gambar 1 *f-function*

Fungsi *key schedule*:

1. Fungsi ini menerima masukan 64-bit kunci $K = k_1 \ldots k_{64}$.
2. Sedangkan keluarannya adalah 256-bit extended key (16-bit subkey K_i, $0 \le i \le 15$).
3. Inisialisasi:
   $U^{(-2)} \leftarrow 0$, $U^{(-1)} \leftarrow k_1 \ldots k_{32}$, $U^{(0)} \leftarrow k_{33} \ldots k_{64}$.
4. $U \stackrel{def}{=} (U_0, U_1, U_2, U_3)$ untuk 8-bit U_i.
5. Hitung $K_0, ..., K_{15}$ untuk $i = 1$ hingga $i = 8$:
   (a) $U \leftarrow f_K(U^{(i-2)}, U^{(i-1)}, U^{(i-3)})$
   f_K didefinisikan pada Tabel 1, dengan A dan B menunjukkan 4-byte vector $(A_0, A_1, A_2, A_3), (B_0, B_1, B_2, B_3)$

   (b) $K_{2i-2} = (U_0, U_1)$
   $K_{2i-1} = (U_2, U_3)$
   $U^{(i)} \leftarrow U$.



Gambar 2 f_K - *function*

2.2 Proses Dekripsi pada FEAL

Untuk mendekripsi ciphertext kembali menjadi plaintext, digunakan cara yang sama dengan sewaktu mengenkripsi plaintext menjadi ciphertext, yaitu dengan menggunakan kunci K yang sama dan ciphertext C = (RS, LS) yang menggantikan masukan plaintext. Namun fungsi *key schedule* digunakan secara terbalik.

Untuk lebih spesifik, berikut ini adalah urutan *key schedule* pada saat melakukan dekripsi:

1. Subkey $((K_{12}, K_{13}),(K_{14},K_{15}))$ digunakan untuk XOR initial subkey
2. Sedangkan subkey $((K_8,K_9),(K_{10},K_{11}))$ digunakan untuk XOR final subkey.
3. Dan putaran kunci digunakan dari i=8 hingga i=1 (tahap 7).

2.3 Kekuatan dan Performansi FEAL

FEAL bekerja dengan kunci blok yang berbeda-beda sehingga menjadikannya aman dari beberapa serangan karena proses dari FEAL ini sendiri dikontrol oleh kunci 64-bit , yang mana lebih aman dari pada kunci DES yang berukuran 56-bit.

FEAL adalah algoritma enkripsi yang sesuai untuk implementasi software. FEAL dapat diaplikasikan secara luas untuk skala kecil atau untuk sistem lain yang telah ada namun tidak dapat menggunakan DES karena terkendala cost. Terlebih lagi, FEAL juga cocok untuk implementasi hardware.

FEAL didesain untuk kecepatan dan kesederhanaan, terutama untuk software pada 8-bit microprocessor. FEAL menggunakan operasi berorientasi byte (8-bit addition mod 256, 2-bit left rotation, dan XOR), menghindari permutasi bit dan table look-up, dan menghasilkan kode program dengan ukuran kecil.

FEAL cipher dapat mencapai kecepatan pemrosesan cipher hingga lebih dari 200kbit/s (lebih dari sepuluh kali lipat kecepatan yang dapat dicapai DES) pada microprocessor software berskala paling besar 400bytes. Hal ini merupakan milestone pada pengembangan praktis software berbasis cipher.

Setelah menjadi cipher algoritma public, FEAL membuka jalan untuk penggunaan cipher oleh berbagai pengguna. FEAL dapat digunakan pada perangkat telekomunikasi berbasis microprocessor dengan mudah karena fiturnya yang compact dan berkecepatan tinggi. Selain itu FEAL juga telah digunakan pada smart card dan sistem telekomunikasi lainnya.

Versi pertama yang diajukan secara komersial adalah FEAL yang menggunakan 4 putaran (FEAL-4), ditempatkan sebagai alternatif tercepat dari DES. Namun ternyata FEAL-4 jauh lebih tidak aman dari yang diharapkan. FEAL-8 yang dikembangkan setelahnya juga ternyata kurang aman dari yang direncanakan. Keamanan

yang dimiliki FEAL-16 atau FEAL-32 lah yang dapat disetarakan dengan tingkat keamanan DES, namun sayangnya throughput yang dihasilkan berkurang jika jumlah putaran bertambah. Terlebih lagi, FEAL sulit untuk meningkatkan kecepatannya seperti implementasi DES yang meningkatkan kecepatannya dengan menggunakan tabel lookup yang sangat besar.

2.4 Serangan pada FEAL

FEAL menstimulus perkembangan berbagai teknik kriptanalis yang keberagamannya dan utilitasnya tidak tertandingi. Hal ini karena banyak analis yang tertarik untuk memecahkan FEAL dengan berbagai teknik kriptanalis agar dapat memecahkan FEAL seefisien mungkin.

Serangan-serangan yang pernah dilakukan terhadap FEAL adalah sebagai berikut:

- Pada 1987, Den Boer menemukan beberapa kerentanan pada FEAL-4. Hal ini mengakibatkan Shimizu dan Miyaguchi, yang merupakan pengembang awal FEAL-4, meningkatkan putaran pada FEAL menjadi delapan putaran pada pemrosesan final.
- tahun 1988 den Boer menunjukkan kerentanan FEAL-4 terhadap adaptive chosen plaintext attack dengan menggunakan 100 hingga 10000 plaintext.
- tahun 1990, Gilbert dan Chasse merancang sebuah chosen plaintext attack pada FEAL-8 dengan menggunakan 10000 pasang plaintext.
- Dengan menggunakan fungsi G milik den Boer, yang mengekspresikan kelinearan pada fungsi f milik FEAL, Murphy berhasil menyerang FEAL-4 dengan 20 chosen plaintext dalam waktu kurang dari 4 jam.
- Metode statistic yang dikembangkan oleh tarfy-corfdir dan gilbert kemudian memungkinkan known plaintext attack pada FEAL-4 dengan menggunakan 1000 teks, dan pada FEAL-6 dengan menggunakan 10000 pasang teks, dengan melibatkan pendekatan linear dari FEAL S-box.
- Linear cryptanalis (LC) versi pertama diperkenalkan oleh Matsui dan Yamaishi. LC ini memungkinkan known plaintext attack terhadap FEAL-4 (dengan menggunakan 5 teks dalam waktu 6 menit pada 25MHz 68040 prosesor), FEAL-6 (dengan menggunakan 100 teks dalam 40 menit), dan FEAL-8 (288 teks dalam waktu yang equivalent dengan exhaustive search pada kunci 50-bit).
- Bilman dan Shamir menggunakan 238 teks untuk FEAL-8 pada konversi known plaintext di Differential Cryptanalisis (DC). Bilman dan Shamir kemudian mengimplementasikan DC chosen plaintext attak dan berhasil menemukan FEAL-8 key hanya dalam dua menit.
- Bilham kemudian menggunakan LC untuk menyerang FEAL-8 dengan 224 known plaintext dalam 10 menit.

3. IDEA

International data encryption (IDEA) adalah symmetric block cipher yang didesain oleh James Massey dari ETH Zurich dan Xuejia Lai. IDEA pertama kali dipublikasikan pada tahun 1991. IDEA dirancang untuk menggantikan DES. IDEA sendir sebenarnya adalah bentuk revisi kecil dari PES(Proposed Ecryption Standard). Pada awalnya IDEA disebut sebagai IPES(Improved PES). IPES kemudiana dikomersialkan dengan nama IDEA dan dipatenkan.

IDEA (International Data Encryption Algorithm) mengenkripsi 64-bit plaintext menjadi 64-bit block ciphertext, menggunakan masukan 128-bit kunci K. Berdasarkan bagian dari struktur Feistel, IDEA terdiri dari 8 putaran komputasi yang identik yang diikuti dengan output transformation.

Putaran r menggunakan enam 16-bit subkey $K_i^{(r)}$, dimana $1 \leq i \leq 6$. Putaran ini bertujuan untuk mentransformasikan 64-bit masukan X menjadi keluaran berupa block berukuran 16-bit, yang kemudian akan men jadi masukan untuk putaran berikutnya. Keluaran dari putaran kedelapan akan menjadi masukan untuk output transformation. Output transformation menggunakan empat subkey tambahan Ki(9), dimana $1 \leq i \leq 4$, untuk menghasilkan ciphertext $Y = (Y_1 , Y_2 , Y_3 , Y_4)$. Dan semua subkey berasal dari K.

Konsep desain utama dari IDEA, yang mendasari keamanan pada IDEA, adalah pencampuran operasi dari tiga kelompok aljabar yang berbeda, yang terdiri dari 2n elemen. Operasi-operasi tersebut adalah sebagai berikut:

- Addition modulo 2^{16}
- Multiplication modulo $2^{16}+1$, dimana semua zero-word (0x0000) ditafsirkan sebagai 2^{16}
- bitwise XOR.

Semua operasi tersebut bekerja pada block sepanjang 16-bit.

Grup operasi yang sesuai pada sub-block a dan b dari panjang bit n = 16 adalah bitwise XOR ($a \oplus b$), addition mod 2^n ($a \boxplus b$), dan multiplication mod 2^n+1 ($a \odot b$).

3.1 Proses Enkripsi pada IDEA

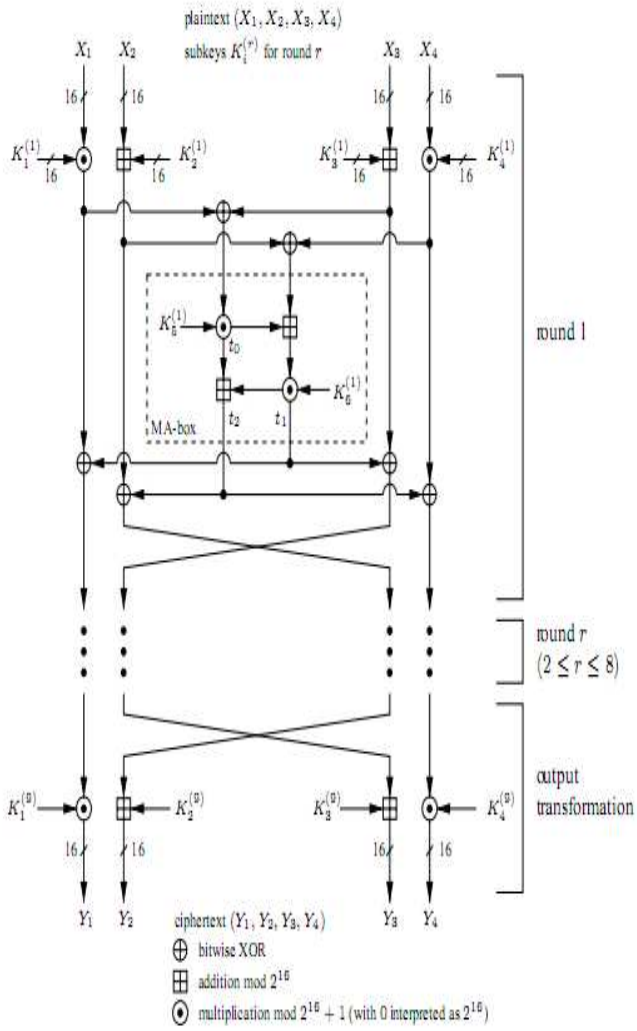
Langkah-langkah enkripsi pada IDEA adalah sebagai berikut:

1. Masukan pada fungsi enkripsi berupa 64-bit plaintext $M = m_1 \ldots m_{64}$ dan 128-bit kunci $K = k_1 \ldots k_{128}$.

2. Keluaran fungsi ini yaitu 64-bit block ciphertext $Y=(Y_1, Y_2, Y_3, Y_4)$
3. Dengan fungsi *key schedule* yang akan dijelaskan di bawah, ubah 16-bit subkey $K_1^{(r)}, ..., K_6^{(r)}$, untuk $1 \leq r \leq 8$, dan $K_1^{(9)}, ..., K_4^{(9)}$untuk output transformation.
4. $(X_1, X_2, X_3, X_4) \leftarrow (m_1 \ldots m_{16}, m_{17} \ldots m_{32}, m_{33} \ldots m_{48}, m_{49} \ldots m_{64})$, dimana X_i adalah data store berukuran 16-bit.
5. Untuk putaran r dari 1 hingga 8, lakukan:
   (a) $X_1 \leftarrow X_1 \odot K_1^{(r)}$
   $X_4 \leftarrow X_4 \odot K_4^{(r)}$
   $X_2 \leftarrow X_2 \boxplus K_2^{(r)}$
   $X_3 \leftarrow X_3 \boxplus K_3^{(r)}$.
   (b) $t_0 \leftarrow K_5^{(r)} \odot (X_1 \oplus X_3)$
   $t_1 \leftarrow K_6^{(r)} \odot (t_0 \boxplus (X_2 \oplus X_4))$
   $t_2 \leftarrow t_0 \boxplus t_1$.
   (c) $X_1 \leftarrow X_1 \oplus t_1$
   $X_4 \leftarrow X_4 \oplus t_2$
   $a \leftarrow X_2 \oplus t_2$
   $X_2 \leftarrow X_3 \oplus t_1$
   $X_3 \leftarrow a$.
6. Output transformation:
   $Y_1 \leftarrow X_1 \odot K_1^{(9)}$
   $Y_4 \leftarrow X_4 \odot K_4^{(9)}$
   $Y_2 \leftarrow X_3 \boxplus K_2^{(9)}$
   $Y_3 \leftarrow X_2 \boxplus K_3^{(9)}$.

Fungsi *key schedule* untuk enkripsi*:*

1. Fungsi ini menerima masukan 128-bit kunci $K = k_1 \ldots k_{128}$.
2. Sedangkan keluarannya yaitu 52 buah kunci sub-block berukuran 16-bit, $K_i^{(r)}$ untuk 8 putaran *r* dan output transformation.
3. Susnlah subkey-subkey menjadi $K_1^{(1)}. \ . \ . \ K_6^{(1)}, K_1^{(2)} \ . \ . \ . \ K_6^{(2)}, \ . \ . \ . \ , K_1^{(8)} \ . \ . \ . \ K_6^{(8)}, K_1^{(9)} \ . \ . \ . K_4^{(9)}$.
4. Partisi *K* menjadi delapan block yang masing-masing berukuran 16-bit. Masukkan nilai-nilai ini ke delapan subkey pertama.
5. Ulangi aktivitas pada tahap (5) ini hingga ke-52 subkey telah memiliki nilai.
   - Geser *K* kekiri sebanyak 25 bit secara siklik.
   - Partisi hasil pergeseran di atas menjadi 8 block yang masing-masing berukuran 16-bit.
   - Masukkan nilai blok-blok ini ke 8 subkey berikutnya.



Gambar 3 Proses Enkripsi pada IDEA

3.2 Proses Dekripsi pada IDEA

Proses dekripsi dilakukan dengan menggunakan metode yang sama dengan proses enkripsi. Proses dekripsi menerima ciphertext *Y* yang menggantikan plaintext *M* pada proses enkripsi, dan kunci enkripsi *K* yang sama. Namun untuk proses dekripsi digunakan key schedule yang berbeda. Fungsi key schedule pada proses dekripsi adalah sebagai berikut:

- Pertama-tama gunakan *K* untuk mendapatkan semua subkey enkripsi $K_i^{(r)}$.
- Dari subkey tersebut, hitung subkey dekripsi $K'_i{}^{(r)}$ yang dapat dilihat pada Tabel 2.
- Gunakan $K'_i{}^{(r)}$ untuk menggantikan $K_i^{(r)}$ pada proses dekripsi (langkah-langkah pada proses dekripsi sama dengan langkah-langkah pada proses enkripsi).

Table 2 Subkey dekripsi pada IDEA

Putaran ke-r	$K'_1{}^{(r)}$	$K'_2{}^{(r)}$	$K'_3{}^{(r)}$
$r=1$	$(K_1^{(10-r)})-1$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$
$2 \le r \le 8$	$(K_1^{(10-r)})-1$	$-K_3^{(10-r)}$	$-K_2^{(10-r)}$
$r=9$	$(K_1^{(10-r)})-1$	$-K_2^{(10-r)}$	$-K_3^{(10-r)}$

Putaran ke-r	$K'_4{}^{(r)}$	$K'_5{}^{(r)}$	$K'_6{}^{(r)}$
$r=1$	$(K_4^{(10-r)})-1$	$K_5^{(9-r)}$	$K_6^{(9-r)}$
$2 \le r \le 8$	$(K_4^{(10-r)})-1$	$K_5^{(9-r)}$	$K_5^{(9-r)}$
$r=9$	$(K_4^{(10-r)})-1$	-	-

Pada tabel di atas $-K_i$ melambangkan additive inverse (mod 216) dari K_i, yaitu suatu bilangan integer u, dimana $u = (216 - K_i)$ AND 0xFFFF, $0 \le u \le 216-1$.

Sedangkan $K_i{}^{-1}$ melambangkan multiplicative inverse (mod $2^{16}+1$) dari K_i, termasuk pada {0, 1, . . . , 216-1} , yang nilainya dapat diturunkan dari perlausan algoritma Euclidean.

3.3 Kekuatan dan Performansi IDEA

Desainer mendesain IDEA dengan analisis yang cermat untuk mengukur kekuatannya melawan differential cryptanalysis dan menyimpukan bahwa IDEA kebal terhadap beberapa asumsi. Tidak ada kelemahan linear atau algebraic yang pernah dilaporkan. Pada tahun 2007, serangan terbaik yang diaplikasikan terhadap seluruh kunci dapat membobol pertahanan IDEA pada putaran ke 6 (IDEA memiliki 8.5 total putaran).

Key schedule yang sangat sederhana membuat IDEA memiliki kunci yang lemah. Beberapa kunci terdiri dari sejumlah besar bilangan 0 bit sehingga mengakibatkan kunci tersebut menjadi lemah (weak key). Namun telah dilakukan sedikit modifikasi pada key schedule untuk memusnahkan weak key tersebut. Seorang analis bernama Meier mengatakan bahwa tidak ada serangan yang mampu melawan 8 putaran penuh IDEA. Hal ini mendukung pernyataan bahwa IDEA tampak aman melawan DC setelah dilakukan 4 putaran dari keseluruhan 8 putaran.

Untuk IDEA dengan 8 putaran penuh, selain serangan terhadap weak key, belum ada serangan lain yang telah dipublikasikan yang lebih baik dari pada exhaustive search pada kunci 128-bit. Saat ini keamanan IDEA masih memiliki kelemahan karena panjang block yang relative kecil (64 bit).

Pada tahun 1999, IDEA bukan lagi termasuk algoritma yang direkomendasikan karena telah banyak algoritma lain yang lebih cepat, beberapa perkembangan dari isu kriptanalisis, dan isu mengenai hak paten.

IV. KESIMPULAN

Dari penjelasan pada bab-bab sebelumnya mengenai FEAL dan IDEA, maka didapatkan beberapa perbedaan mendasar antara FEAL dan IDEA.

- FEAL menerima masukan berupa 64-bit plaintext dan 64 bit kunci, dan menghasilkan keluaran berupa 64-bit ciphertext. Sedangkan IDEA menerima masukan berupa 64-bit plaintext dan 128-bit kunci, dan menggunakan subkey 16-bit. Keduanya sama-sama beroperasi dalam 64-bit block.
- FEAL memiliki beberapa variasi jenis yang menentukan banyak jumlah putaran pada proses enkripsi-dekripsi. Variasi tersebut antara lain FEAL-4, FEAL-6, dan FEAL-8. Sedangkan IDEA terdiri dari 8 putaran identik dan sebuah output transformation.
- FEAL memiliki fungsi f yang sederhana, dengan menggunakan operasi XOR di dalamnya. Hal ini yang membuat FEAL menjadi algoritma yang sederhana dan dapat beroperasi dengan cepat, sehingga algoritma ini cocok diimplementasikan pada perangkat lunak. Sedangkan IDEA memiliki fungsi enkripsi yang kuat dan aman. Namun kesederhanaan pada key schedule yang dimiliki IDEA mengakibatkan IDEA memiliki kunci yang lemah.
- Proses enkripsi pada IDEA menggunakan campuran operasi dari tiga kelompok aljabar, yaitu addition modulo, multiplication modulo, dan bitwise XOR. Campuran ketiga operasi inilah yang menjamin keamanan IDEA.
- Keamanan pada FEAL berasal dari panjang kunci yang digunakan yaitu 64-bit dan pada prosesnya, block kunci yang digunakan akan berubah-ubah. Namun sebenarnya FEAL cenderung mudah untuk diserang. Namun hal ini justru mengakibatkan lahirnya berbagai teknik kriptanalisis baru karena banyak analis yang tertarik untuk memecahkan FEAL dengan berbagai teknik kriptanalisis agar dapat memecahkan FEAL seefisien mungkin.
- Proses dekripsi yang terjadi pada FEAL menggunakan proses yang sama dengan proses enkripsi. Namun dengan urutan penggunaan subkey pada key schedule yang dibalik. Sedangkan proses dekripsi pada IDEA menggunakan subkey dekripsi yang didapatkan dari subkey enkripsi.

Bila dibandingkan antara algoritma FEAL dan IDEA, masing-masing memiliki kelebihan dan kekurangannya masing-masing. FLEA dengan kecepatannya dan IDEA dengan keamanannya. Sehingga tidak bijaksana untuk mengatakan algoritma mana yang lebih baik. Seperti yang telah dikatakan di awal pembahasan, tidak ada satupun

block cipher yg sesuai untuk semua aplikasi, bahkan sekalipun cipher tersebut menawarkan keamanan tingkat tinggi.Oleh karena itu, berbagai cipher diciptakan untuk melengkapi satu sama lainnya.

REFERENSI

[1] http://en.wikipedia.org/wiki/FEAL
[2]http://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm
[3] http://www.cacr.math.uwaterloo.ca/hac/about/chap7.pdf
[4] Munir, Rinaldi, *Diktat Kuliah IF3054 Kriptografi*, Bandung
[5] http://www.quadibloc.com/crypto/co040302.htm