STUDI PERBANDINGAN CIPHER BLOK ALGORITMA BLOWFISH DAN ALGORITMA CAMELLIA

Jonathan Marcel T (13507072)

Program Studi Teknik Informatika Institut Teknologi Bandung
Jalan Ganeca 10 Bandung
E-mail: cel_tum@yahoo.co.id

Abstrak *– Masalah keamanan data merupakan salah satu isu terpenting di dalam proses pengiriman data. Salah satu upaya untuk menjamin keamanan data ialah menggunakan algoritma kriptografi. Algoritma kriptografi ialah mengubah tulisan menjadi tidak dapat terbaca dengan menggunakan berbagai algoritma yang dirancang kriptografer. Kriptografi telah berkembang sejak zaman dahulu. Pada saat ini kriptografi terus berkembang sehingga algoritma-algoritma kriptografi yang ada semakin bertambah jumlahnya dan semakin sulit ditembus. Contoh-contoh algoritma kriptografi yang terkenal ialah Blowfish dan Camellia. Makalah ini akan membahas kedua algoritma tersebut secara mendalam, antara lain studi mengenai algoritmanya, analisis kelebihan maupun kekurangan masing-masing algoritma, serta contoh implementasi kedua algoritma tersebut. Pada akhirnya dapat dibandingkan algoritma manakah yang lebih unggul dibandingkan yang lain.*

Kata kunci*: Cipher Blok, Feistel Cipher, Blowfish, Camellia*

Pendahuluan

Pada saat ini keamanan data merupakan isu yang sangat penting dalam proses penukaran data. Masalah keamanan data ini merupakan tujuan utama dari ilmu kriptografi. Berbagai kriptografer membuat algoritma enkripsi/dekripsinya masing-masing. Enkripsi merupakan proses pengubahan plainteks menjadi cipherteks, sedangkan dekripsi merupakan proses pengubahan kembali cipherteks menjadi plainteks. Algoritma kriptografi ini sudah ada sejak zaman dahulu, atau disebut juga algoritma kriptografi klasik. Algoritma kriptografi klasik dianggap sangat tidak aman, karena menggunakan kunci yang mudah ditebak serta mekanisme enkripsi/dekripsi yang sangat sederhana. Pengembangan dari algoritma

kriptografi klasik ialah algoritma kriptografi modern. Algoritma kriptografi modern sangat banyak jenisnya. Ada algoritma yang dianggap lebih kuat dibandingkan yang lain, baik dari segi keamanan maupun kecepatan enkripsi/dekripsi.

Standar dari algoritma kriptografi modern ialah algoritma DES (Data Encryption Standard) yang dikembangkan oleh IBM pada tahun 1972. Namun ternyata algoritma ini menyimpan banyak kelemahan dan dianggap tidak aman lagi untuk pengenkripsian data. Karena itulah, dikembangkan algoritma kriptografi lainnya untuk menggantikan DES, antara lain algoritma Blowfish dan algoritma Camellia. Kedua algoritma inilah yang akan menjadi pembahasan utama dalam makalah ini.

Dasar perbandingan kedua algoritma kriptografi ini adalah kedua algoritma sama-sama merupakan algoritma kriptografi kunci simetri, yaitu algoritma kriptografi yang menggunakan kunci yang sama untuk proses enkripsi dan proses dekripsi. Kedua algoritma ini sama-sama dikembangkan untuk menggantikan algoritma kriptografi standar DES yang dianggap sudah tidak aman. Dalam prakteknya, kedua algoritma kriptografi ini sama-sama digunakan di dalam berbagai bidang keamanan data dan informasi.

Algoritma kriptografi kunci simetri

Algoritma kunci simetri berarti menggunakan kunci yang sama untuk proses enkripsi maupun dekripsi.Ada dua kategori algoritma kriptografi kunci simetri, yaitu: Stream cipher dan Blok cipher. Perbedaan utamanya yaitu, stream cipher beroperasi pada plainteks/cipherteks dalam bentuk bit tunggal, sedangkan blok cipher beroperasi dalam bentuk blok bit. Algoritma Blowfish dan Camellia merupakan algoritma kunci simetri Blok Cipher.

Karakteristik utama dari algoritma blok cipher ialah plainteks akan dibagi menjadi blok-blok bit yang panjangnya sama, misalnya 64 bit. Enkripsi akan dilakukan terhadap masing-masing blok bit dan menghasilkan blok cipherteks yang berukuran sama dengan blok plainteks. Kemudian prosedur dekripsi dilakukan dengan cara yang sama dengan enkripsi. Pada blok cipher terdapat empat buah mode yang lazim diterapkan untuk melakukan enkripsi, yaitu Electronic Code Book (EBC), Cipher Blok Chaining (CBC), Cipher Feedback (CFB), dan Output Feedback (OFB). Pada makalah ini tidak akan dibahas lebih lanjut mengenai keempat mode tersebut.

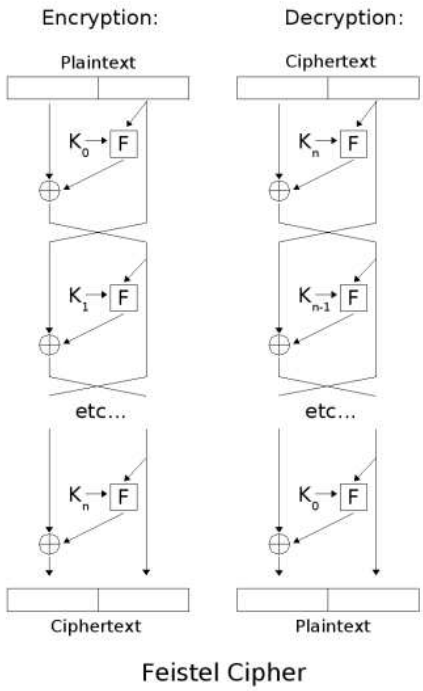
Jaringan Feistel

Jaringan Feistel, atau disebut juga Feistel Cipher merupakan prinsip utama perancangan blok cipher. Hampir semua algoritma enkripsi blok cipher merupakan modifikasi model Feistel Cipher. Feistel Cipher dikembangkan oleh Horst Feistel pada tahun 1970.

Prinsip Feistel Cipher ialah membagi blok menjadi dua bagian yang sama besar. Misalnya pada blok berukuran 128 bit maka akan dibagi menjadi dua sub-blok yang berukuran sama, yaitu L berukuran 64 bit dan R berukuran 64 bit juga.
Kemudian didefinisikan blok cipher berulang, yaitu hasil putaran ke-i ditentukan dari hasil putaran sebelumnya:

$L_i = R_{i-1}$
$R_i = L_{i-1}$ XOR $f(R_{i-1}, K_i)$

Dimana i merupakan putaran ke-i, K_i merupakan subkunci pada putaran ke-i dan f merupakan fungsi transformasi yang di-desain oleh kriptografer.
Setelah putaran terakhir akan diperoleh L dan R yang apabila digabung menjadi cipherteks dari blok yang dienkripsi tersebut.



Gambar 1 Struktur Feistel Cipher

Sifat dari Feistel Cipher ialah reversible, sehingga memungkinkan kriptografer tidak perlu membuat algoritma dekripsi baru untuk mendekripsikan cipherteks menjadi plainteks. Sifat reversible ini tidak bergantung pada fungsi F, sehingga fungsi F dapat dibuat serumit-rumitnya.

S-box

S-box adalah matriks yang berisi substitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit yang lain. Pada algoritma blok cipher kebanyakan, S-box ini akan memetakan m buah bit input menjadi n buah bit output, sehingga S-box ini dinamakan m*n S-box.

Isu yang paling penting adalah bagaimana kriptografer merancang S-box sedemikian rupa sehingga memiliki kekuatan kriptografi yang baik.

Pada umumnya ada empat pendekatan yang digunakan dalam pengisian S-box, yaitu:

1. Dipilih secara acak
   Cara pengisian ini lebih baik untuk S-box yang berukuran besar.
2. Dipilih secara acak kemudian diuji
   Mirip dengan nomor satu, tetapi nilai yang dibangkitkan secara acak diuji apakah memenuhi sifat tertentu.
3. Dibuat oleh orang
   S-box diisi dengan teknik yang lebih bersifat intuitif.
4. Dihitung secara matematis
   Entri di dalam S-box dibangkitkan melalui perhitungan matematika.

Algoritma Blowfish

Algoritma Blowfish dikembangkan oleh Bruce Scheiner pada tahun 1993. Tujuan beliau mengembangkan algoritma ini ialah untuk menggantikan algoritma DES (Data Encryption Standard). Algoritma ini tidak dipatenkan sehingga semua orang dapat bebas menggunakan algoritma ini untuk mengamankan data.

Ukuran panjang sebuah blok dari algoritma Blowish adalah 64 bit atau 8 byte. Panjang kunci bervariasi mulai dari 32 bit atau 4 byte hingga 448 bit atau 56 byte. Menggunakan Feistel Cipher sebanyak 16 putaran. Algoritma Blowfish memiliki P-array berukuran 18 yang masing-masingnya berisi 32 bit subkunci, serta empat buah S-box dengan 256 entri.

Mekanisme pembangkitan subkunci adalah sebagai berikut:
Inisialisasi P-array[18] dan S-box[256] dengan sembarang nilai. P-array mengandung 18 buah nilai yang masing-masingnya berukuran 32 bit. S-box mengandung 256 buah nilai yang masing-masingnya berukuran 8 bit. Angka yang dibangkitkan harus mengandung digit heksadesimal dari konstanta pi (kecuali angka 3 di depan koma).
Kunci akan di-XOR dengan P-entries secara berurutan, bisa dilakukan pengulangan apabila diperlukan. Operasi XOR akan berlangsung terus menerus hingga P18. Misalnya P1 di-XOR dengan 32 bit pertama kunci, P2 di-XOR dengan 32 bit berikutnya, dan seterusnya.

Sebuah blok berukuran 64 bit yang seluruh nilai bitnya ialah 0 akan dienkripsi dengan algoritma Blowfish menggunakan subkunci yang diperoleh dari hasil sebelumnya. Hasil ciphertextnya akan menggantikan P1 dan P2. Setelah itu ciphertext ini dienkripsi kembali dengan sub-kunci yang berbeda, karena nilai P1 dan P2 telah berubah.

Hasil enkripsi ini kemudian akan menggantikan P3 dan P4. Demikian seterusnya hingga seluruh nilai P-array dan S-box digantikan.Sehingga total diperlukan 521 kali iterasi untuk membangkitkan semua subkunci.

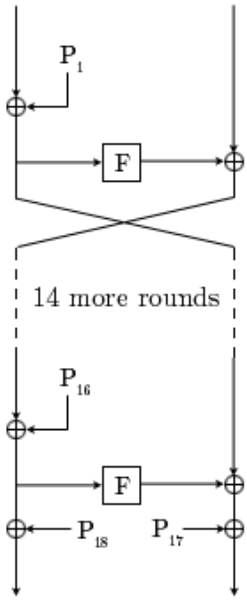
Blowfish sendiri memodifikasi Feistel Cipher. Blok pertama dari plainteks akan dibagi menjadi 2, masing-masing sub-blok berukuran 32 bit. Sub-blok kiri (L) akan di-XOR dengan P1 kemudian masuk ke F-function. Hasil XOR yang tidak dimasukkan ke F-function menjadi subblok kanan (R) pada tahap selanjutnya.

Pseudo-code algoritma enkripsinya adalah sebagai berikut:

```
Bagi blok plainteks x menjadi xL dan xR berukuran 32 bit
For i=1 to i<=16
     xL = xL XOR Pi
     xR = F(xL)XOR xR
```

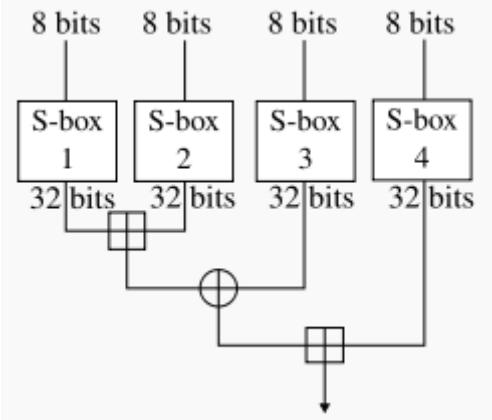
```
     Tukar xL dengan xR
Tukar xL dengan xR /*untuk membatalkan pertukaran terakhir*/
xR = xR XOR P17
xL = xL XOR P18
Gabung xL dengan xR, itulah hasil cipherteks dari blok
plainteks X
```

Agar lebih mudah dipahami silahkan lihat gambar 2:



Gambar 2 Struktur Blowfish Cipher

Adapun F-function untuk Blowfish seperti pada gambar 3:



Gambar 3 Struktur F-function Blowfish Cipher

F-function terdiri dari 4 buah S-box yang masing-masing menerima input 8 bit dan menghasilkan output 32 bit. Jadi 32 ketika bit hasil XOR P1 dan subblok kiri memasuki F-function, akan dipecah menjadi 4 buah bagian yang masing-masing 8 bit. Masing-masing

akan mengalami substitusi dan hasil dari sebuah S-box ialah 32 bit. Hasil dari S-box 1 akan dijumlahkan modular dengan hasil dari S-box 2, kemudian di XOR dengan hasil dari S-box 3, dan terakhir dijumlahkan modular dengan hasil dari S-box 4.

Secara matematis F-function dapat ditulis sebagai berikut:

$F(xL) = ((S_{1,a} + S_{2,b} \bmod 2^{32}) \text{ XOR } S_{3,c}) + S_{4,d} \bmod 2^{32}$

Hasil dari F-function ini akan di XOR dengan subblok kanan (R) dari blok pertama. Hasilnya akan menjadi subblok kiri untuk kemudian di XOR lagi dengan P2, dan seterusnya.

Setelah diulang sebanyak 16 kali, subblok kiri terakhir di XOR dengan P17, dan subblok kanan terakhir di XOR dengan P18. Hasil inilah yang merupakan ciphertext dari blok pertama.
Selanjutnya dienkripsi blok kedua dan seterusnya dari plainteks dengan algoritma enkripsi yang sama.

Bagaimana proses dekripsi pada algoritma Blowfish? Proses dekripsinya sama seperti ketika mengenkripsi, namun penggunaan subkuncinya dilakukan secara terbalik. Jadi, blok cipherteks pertama-tama akan dibagi menjadi dua kemudian didekripsi dengan subkunci P_{17} dan P_{18}, begitu seterusnya sampai dekripsi dengan P_1 dan P_2.

Algoritma Camellia

Algoritma Camellia dikembangkan secara bersama oleh NTT dan Mitsubishi Electric Corporation pada tahun 2000. Algoritma ini mengadopsi algoritma kriptografi E2 (dikembangkan oleh NTT) dan algoritma kriptografi MISTY (dikembangkan oleh Mitsubishi).

Sebuah blok memiliki ukuran 128 bit. Panjang kunci bervariasi antara 128 bit, 192 bit atau 256 bit. Merupakan modifikasi Feistel Cipher sebanyak 18 putaran (ketika panjang kunci 128 bit) atau 24 putaran (ketika panjang kunci 192 atau 256 bit).

Sebelum membahas lebih lanjut, saya akan membahas tentang mekanisme pembangkitan kunci pada algoritma Camellia.

Pertama-tama dibangkitkan dua buah variabel 128 bit K_A dan K_B. K_B hanya digunakan ketika panjang kunci 192 bit atau 256 bit.Bagaimana cara pembangkitannya? Kunci rahasia dipisahkan menjadi dua bagian K_L dan K_R, masing-masing berukuran 128 bit. Untuk panjang kunci 128 bit, $K_R = 0$. K_L dan K_R di-XOR kemudian dilakukan enkripsi dua tahap menggunakan nilai konstanta $\sum_1$ dan $\sum_2$. Konstanta ini berukuran masing-masing 64 bit. Hasilnya di-XOR dengan K_L kemudian dienkripsi dua tahap lagi dengan menggunakan nilai konstanta $\sum_3$ dan $\sum_4$. Hasilnya ialah K_A. Setelah itu K_A di-XOR dengan K_R dan dienkripsi dua tahap lagi menggunakan nilai konstanta $\sum_5$ dan $\sum_6$.

Adapun $\sum_i$ merupakan representasi nilai kontinyu dari tempat kedua heksadesimal dan tempat ketujuhbelas heksadesimal dari akar kuadrat bilangan prima ke-i. Untuk lebih lengkapnya silahkan lihat tabel 1.

$\sum_1$	0xA09E6673FBCC908B
$\sum_2$	0xB67AE8584CAA73B2
$\sum_3$	0xC6EF372FE94F82BE
$\sum_4$	0x54FF53A5F1D36F1C
$\sum_5$	0x10E527FADE682D1D
$\sum_6$	0xB05688C2B3E6C1FD

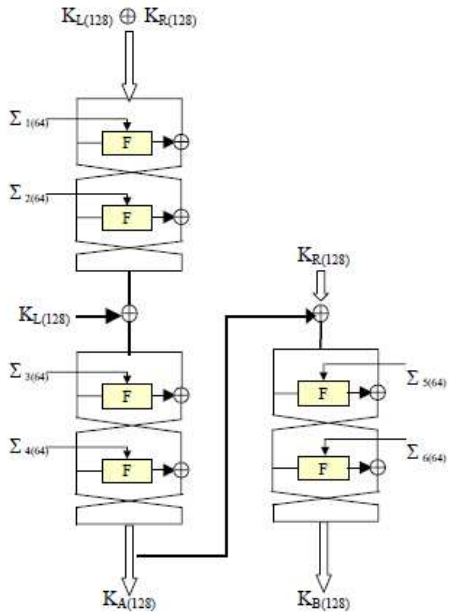
Tabel 1 Bilangan Desimal $\sum_i$

Setelah itu pembangkitan subkunci Kw dan kl dibangkitkan dari nilai hasil rotasi penggeseran K_L, K_R, K_A, dan K_B. <<<n merupakan operasi penggeseran sebanyak n bit. Untuk lebih detailnya bisa dilihat tabel 2. Subkunci kw dan kl masing-masing memiliki panjang 64 bit.

	Subkunci	Nilai
Prewhitening	$Kw_{1(64)}$	$(KL<<<0)_{L(64)}$
	$Kw_{2(64)}$	$(KL<<<0)_{R(64)}$
F (Tahap1)	$k_{1(64)}$	$(KA<<<0)_{L(64)}$
F (Tahap2)	$k_{2(64)}$	$(KA<<<0)_{R(64)}$
F (Tahap3)	$k_{3(64)}$	$(KL<<<15)_{L(64)}$
F (Tahap4)	$k_{4(64)}$	$(KL<<<15)_{R(64)}$
F (Tahap5)	$k_{5(64)}$	$(KA<<<15)_{L(64)}$
F (Tahap6)	$k_{6(64)}$	$(KA<<<15)_{R(64)}$
FL	$kl_{1(64)}$	$(KA<<<30)_{L(64)}$
FL^{-1}	$kl_{2(64)}$	$(KA<<<30)_{R(64)}$
F (Tahap7)	$k_{7(64)}$	$(KL<<<45)_{L(64)}$
F (Tahap8)	$k_{8(64)}$	$(KL<<<45)_{R(64)}$
F (Tahap9)	$k_{9(64)}$	$(KA<<<45)_{L(64)}$
F (Tahap10)	$k_{10(64)}$	$(KL<<<60)_{R(64)}$
F (Tahap11)	$k_{11(64)}$	$(KA<<<60)_{L(64)}$
F (Tahap12)	$k_{12(64)}$	$(KA<<<60)_{R(64)}$
FL	$kl_{3(64)}$	$(KL<<<77)_{L(64)}$
FL^{-1}	$kl_{4(64)}$	$(KL<<<77)_{R(64)}$
F (Tahap13)	$k_{13(64)}$	$(KL<<<94)_{L(64)}$
F (Tahap14)	$k_{14(64)}$	$(KL<<<94)_{R(64)}$
F (Tahap15)	$k_{15(64)}$	$(KA<<<94)_{L(64)}$
F (Tahap16)	$k_{16(64)}$	$(KA<<<94)_{R(64)}$
F (Tahap17)	$k_{17(64)}$	$(KL<<<111)_{L(64)}$
F (Tahap18)	$k_{18(64)}$	$(KL<<<111)_{R(64)}$
Postwhitening	$Kw_{3(64)}$	$(KA<<<111)_{L(64)}$
	$Kw_{4(64)}$	$(KA<<<111)_{R(64)}$

Tabel 2 Nilai Rotasi Penggeseran

Sedangkan mekanisme pembangkitan kunci secara lebih lengkap dijelaskan pada gambar 4:



Gambar 4 Key Scheduling Camellia Cipher

Setelah pembahasan tentang pembangkitan kunci pada algoritma Camellia. Berikutnya adalah pembahasan tentang mekanisme enkripsi dan dekripsi pada algoritma Camellia.

Enkripsi menggunakan kunci 128 bit akan berbeda dengan enkripsi menggunakan kunci 192 bit atau 256 bit.Pada enkripsi dengan kunci 128 bit, struktur Feistel Cipher terdiri atas 18 putaran dengan 2 lapisan fungsi FL/FL^{-1} setelah tahap keenam dan tahap keduabelas.

Pertama-tama blok pertama plainteks dipecah menjadi dua, kemudian masing-masing di-XOR dengan Kw_1 dan Kw_2. Hasilnya ialah L_0 dan R_0. Pada putaran pertama, L_0 dimasukkan ke fungsi F beserta k_1. Hasilnya kemudian di-XOR dengan R_0. Pada putaran kedua, L_0 akan menjadi R_1, sedangkan hasil XOR tadi menjadi L_1. L_1 ini kemudian masuk ke fungsi F lagi beserta k_2. Demikian seterusnya hingga putaran keenam.

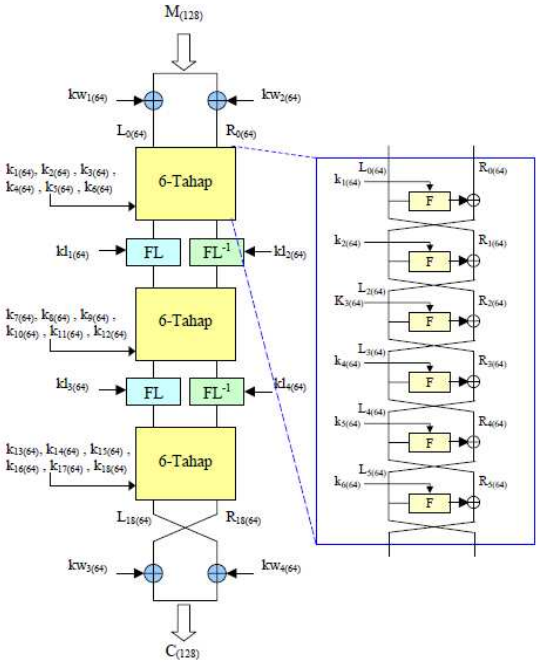
Sebelum putaran ketujuh, L_6 akan memasuki fungsi FL bersama kl_1, dan R_6 akan memasuki fungsi FL^{-1}bersama kl_2. Hasil masing-masing akan kembali dienkripsi dengan cara seperti tadi sebanyak 6 putaran lagi, dan sebelum putaran ketigabelas akan menggunakan fungsi FL/FL^{-1} lagi. Kemudian dienkripsi dengan fungsi F lagi sebanyak 6 putaran. Sehingga total putaran adalah 18. Terakhir L_{18} akan di-XOR dengan kw_4 dan R_{18} akan di-XOR dengan kw_3. Hasilnya kemudian digabung, itulah cipherteks dari blok pertama plainteks.

Apabila ditulis dalam persamaan matematis, hasilnya seperti dibawah ini:

$L_i = R_{i-1}$ XOR $F(L_{i-1}, k_i)$
$R_i = L_{i-1}$

Operasi tersebut dilakukan untuk i=1 sampai i=18, kecuali untuk i=6 dan i=12. Untuk i=6 dan i=12 dilakukan operasi sebagai berikut:

$L'i = R_{i-1}$ XOR $F(L_{i-1}, k_i)$
$R_i = L_{i-1}$
$Li = FL(L'i, kl_{2i/6-1})$
$Ri = FL^{-1}(R'i, kl_{2i/6})$



Gambar 5 Enkripsi Camellia Cipher Mode 128 Bit

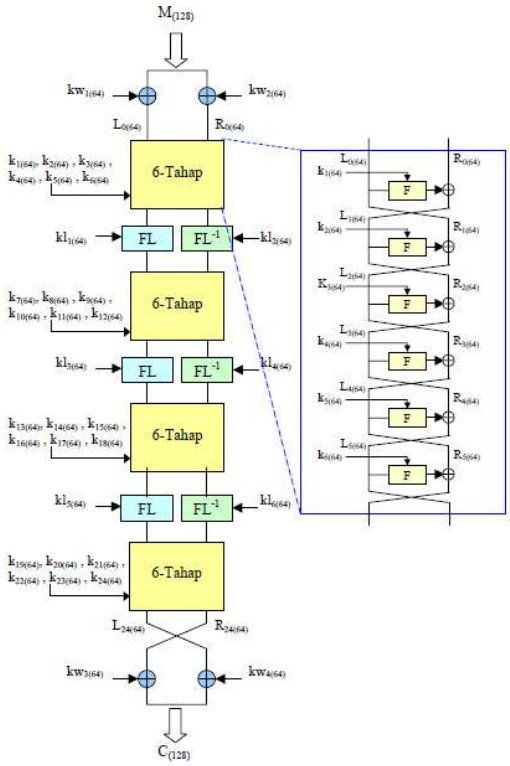
Enkripsi dengan menggunakan kunci 192 bit dan 256 bit memiliki mekanisme yang sedikit berbeda. Pada enkripsi ini diselipkan 3 buah layer fungsi FL/FL^{-1}sebelum putaran ketujuh, ketigabelas, dan kedelapanbelas. Tetapi pada putaran lainnya secara umum prosedur enkripsinya sama dengan enkripsi menggunakan kunci 128 bit. Total putaran pada mekanisme enkripsi ini adalah 24 buah putaran. Apabila ditulis dalam persamaan matematis, hasilnya ialah sebagai berikut:

$L_i = R_{i-1}$ XOR $F(L_{i-1}, k_i)$
$R_i = L_{i-1}$

Sama seperti pada kunci 128 bit, operasi tersebut dilakukan untuk i=1 sampai i=24, kecuali untuk i=6, i=12, dan i=18. Berikut operasi yang dilakukan untuk i=6, i=12, dan i=18.

$L'i = R_{i-1}$ XOR $F(L_{i-1}, k_i)$
$R_i = L_{i-1}$
$Li = FL(L'i, kl_{2i/6-1})$
$Ri = FL^{-1}(R'i, kl_{2i/6})$

Untuk lebih jelasnya silahkan lihat gambar 6:



Gambar 6 Enkripsi Camellia Cipher Mode 192/256 Bit

Bagaimana dengan prosedur dekripsinya? Sama seperti prosedur enkripsi, terdapat sedikit perbedaan pada prosedur dekripsi dengan menggunakan kunci 128 bit dan prosedur dekripsi dengan menggunakan kunci 192 bit atau 256 bit.

Pada dekripsi dengan kunci 128 bit, pertama-tama blok cipherteks akan dibagi menjadi dua kemudian masing-masing di-XOR dengan kw_3 dan kw_4. Setelah itu akan subblok kiri maupun kanan akan didekripsi dengan cara yang sama seperti pada proses enkripsi, namun perbedaannya ialah penggunaan kuncinya berkebalikan, yaitu k_{18}, k_{17}, hinggak$_{13}$. Kemudian, sama seperti proses enkripsi, sebelum putaran berikutnya akan memasuki fungsi FL dan FL^{-1}, namun dengan subkunci yang berkebalikan, yaitu kl_4 dan kl_3. Ingatlah bahwa pada proses enkripsi digunakan kl_1 dan kl_2. Setelah itu kembali akan memasuki Feistel Cipher sebanyak 6 putaran lagi, masing-masing dengan menggunakan subkunci k_{12}, k_{11}, hingga k_7. Kemudian masuk ke fungsi FL dan FL^{-1} menggunakan subkunci kl_2 dan kl_1. Setelah itu kembali didekripsi dengan Feistel Cipher 6 putaran lagi, masing-masing dengan menggunakan subkunci k_6, k_5, hingga k_1. Pada tahap terakhir, subblok kiri akan di-XOR dengan kw_1 dan subblok kanan akan di-XOR dengan kw_2. Hasil penggabungan dua subblok ini merupakan plainteks blok cipherteks. Apabila ditulis dalam persamaan matematis, hasilnya sebagai berikut:

$R_{i-1} = L_i$ XOR $F(R_i, k_i)$

$L_{i-1} = R_i$

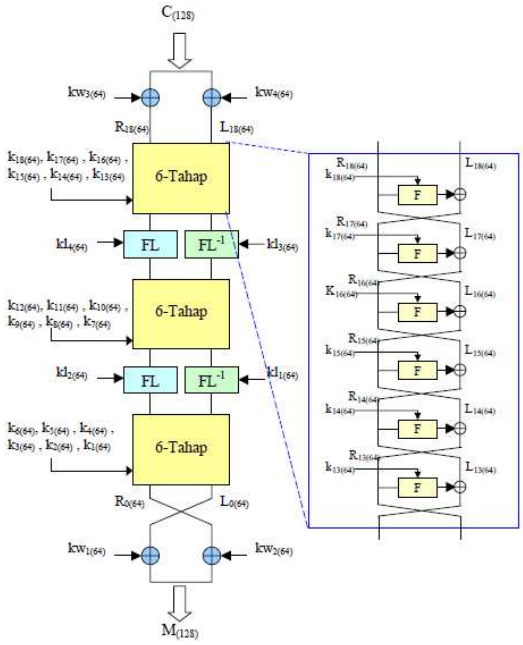
Operasi tersebut dilakukan untuk i=18 sampai i=1, kecuali untuk i=13 dan i=1. Untuk i=13 dan i=1 dilakukan operasi sebagai berikut:

$R'_{i-1} = L_i$ XOR $F(R_i, k_i)$

$L'_{i-1} = R_i$

$R_{i-1} = FL(R'_{i-1}, kl_{2(i-1)/6})$

$L_{i-1} = FL^{-1}(L'_{i-1}, kl_{2(i-1)/6-1})$



Gambar 7 Dekripsi Camellia Cipher Mode 128 Bit

Jadi proses dekripsi algoritma Camellia mirip dengan proses enkripsi, tetapi perbedaannya ialah urutan penggunaan subkunci berkebalikan dengan proses enkripsi.

Bagaimana dengan proses dekripsi dengan menggunakan kunci 192 bit atau 256 bit?
Sama seperti proses dekripsi pada algoritma Camellia dengan kunci 128 bit, proses dekripsi juga menggunakan prosedur yang sama seperti enkripsi algoritma Camellia dengan kunci 192 bit atau 256 bit, namun dengan urutan penggunaan subkunci yang berkebalikan.

Operasi yang dilakukan untuk i=24 sampai i=1 adalah sebagai berikut:

$R_{i-1} = L_i \text{ XOR } F(R_i, k_i)$

$L_{i-1} = R_i$

Kecuali untuk i=19, i=13, dan i=7 dilakukan operasi matematis sebagai berikut:

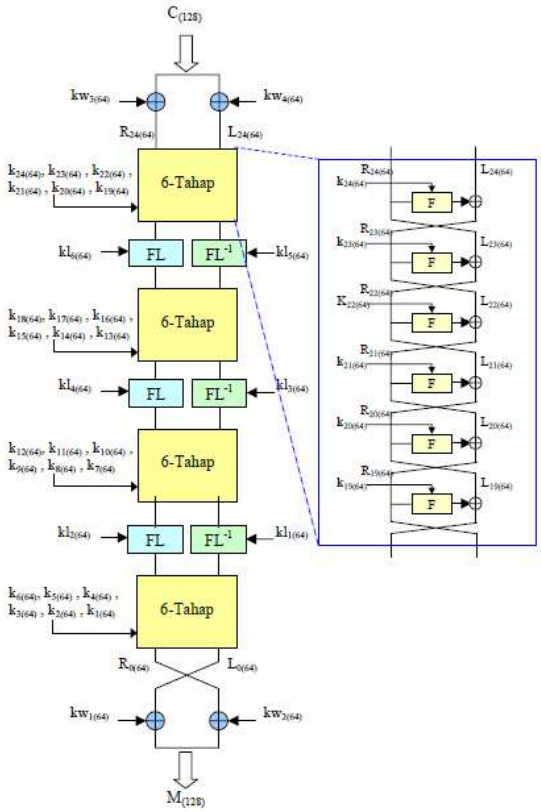
$R'_{i-1} = L_i \text{ XOR } F(R_i, k_i)$

$L'_{i-1} = R_i$

$R_{i-1} = FL(R'_{i-1}, kl_{2(i-1)/6})$

$L_{i-1} = FL^{-1}(L'_{i-1}, kl_{2(i-1)/6-1})$

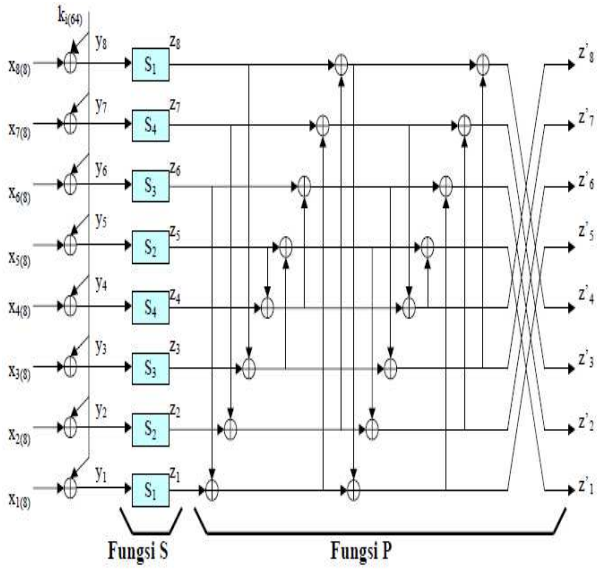
Untuk lebih jelasnya silahkan lihat gambar 8:



Gambar 8 Dekripsi Camellia Cipher Mode 192/256 Bit

Pada algoritma Camellia terdapat tiga buah fungsi internal, yaitu fungsi F, fungsi FL, dan fungsi FL^{-1}.

Fungsi F diperlihatkan pada gambar 9:



Gambar 9 Struktur F-Function Camellia Cipher

F:L*L → L

(X,k) → Y = P(S(X XOR k))

Fungsi S merupakan fungsi yang menggunakan empat buah S-box. Masing-masing S-box ini menerima input 8 bit dan menghasilkan keluaran 8 bit. S-box yang digunakan ialah berukuran 16*16. Masing-masing S-box ini diperlihatkan pada tabel berikut.

112	130	44	236	179	39	192	229	228	133	87	53	234	12	174	65
35	239	107	147	69	25	165	33	237	14	79	78	29	101	146	189
134	184	175	143	124	235	31	206	62	48	220	95	94	197	11	26
166	225	57	202	213	71	93	61	217	1	90	214	81	86	108	77
139	13	154	102	251	204	176	45	116	18	43	32	240	177	132	153
223	76	203	194	52	126	118	5	109	183	169	49	209	23	4	215
20	88	58	97	222	27	17	28	50	15	156	22	83	24	242	34
254	68	207	178	195	181	122	145	36	8	232	168	96	252	105	80
170	208	160	125	161	137	98	151	84	91	30	149	224	255	100	210
16	196	0	72	163	247	117	219	138	3	230	218	9	63	221	148
135	92	131	2	205	74	144	51	115	103	246	243	157	127	191	226
82	155	216	38	200	55	198	59	129	150	111	75	19	190	99	46
233	121	167	140	159	110	188	142	41	245	249	182	47	253	180	89
120	152	6	106	231	70	113	186	212	37	171	66	136	162	141	250
114	7	185	85	248	238	172	10	54	73	42	104	60	56	241	164
64	40	211	123	187	201	67	193	21	227	173	244	119	199	128	158

Tabel 3 S-Box 1

224	5	88	217	103	78	129	203	201	11	174	106	213	24	93	130
70	223	214	39	138	50	75	66	219	28	158	156	58	202	37	123
13	113	95	31	248	215	62	157	124	96	185	190	188	139	22	52
77	195	114	149	171	142	186	122	179	2	180	173	162	172	216	154
23	26	53	204	247	153	97	90	232	36	86	64	225	99	9	51
191	152	151	133	104	252	236	10	218	111	83	98	163	46	8	175
40	176	116	194	189	54	34	56	100	30	57	44	166	48	229	68
253	136	159	101	135	107	244	35	72	16	209	81	192	249	210	160
85	161	65	250	67	19	196	47	168	182	60	43	193	255	200	165
32	137	0	144	71	239	234	183	21	6	205	181	18	126	187	41
15	184	7	4	155	148	33	102	230	206	237	231	59	254	127	197
164	55	177	76	145	110	141	118	3	45	222	150	38	125	198	92
211	242	79	25	63	220	121	29	82	235	243	109	94	251	105	178
240	49	12	212	207	140	226	117	169	74	87	132	17	69	27	245
228	14	115	170	241	221	89	20	108	146	84	208	120	112	227	73
128	80	167	246	119	147	134	131	42	199	91	233	238	143	1	61

Tabel 4 S-Box 2

56	65	22	118	217	147	96	242	114	194	171	154	117	6	87	160
145	247	181	201	162	140	210	144	246	7	167	39	142	178	73	222
67	92	215	199	62	245	143	103	31	24	110	175	47	226	133	13
83	240	156	101	234	163	174	158	236	128	45	107	168	43	54	166
197	134	77	51	253	102	88	150	58	9	149	16	120	216	66	204
239	38	229	97	26	63	59	130	182	219	212	152	232	139	2	235
10	44	29	176	111	141	136	14	25	135	78	11	169	12	121	17
127	34	231	89	225	218	61	200	18	4	116	84	48	126	180	40
85	104	80	190	208	196	49	203	42	173	15	202	112	255	50	105
8	98	0	36	209	251	186	237	69	129	115	109	132	159	238	74
195	46	193	1	230	37	72	153	185	179	123	249	206	191	223	113
41	205	108	19	100	155	99	157	192	75	183	165	137	95	177	23
244	188	211	70	207	55	94	71	148	250	252	91	151	254	90	172
60	76	3	53	243	35	184	93	106	146	213	33	68	81	198	125
57	131	220	170	124	119	86	5	27	164	21	52	30	28	248	82
32	20	233	189	221	228	161	224	138	241	214	122	187	227	64	79

Tabel 5 S-Box 3

112	44	179	192	228	87	234	174	35	107	69	165	237	79	29	146
134	175	124	31	62	220	94	11	166	57	213	93	217	90	81	108
139	154	251	176	116	43	240	132	223	203	52	118	109	169	209	4
20	58	222	17	50	156	83	242	254	207	195	122	36	232	96	105
170	160	161	98	84	30	224	100	16	0	163	117	138	230	9	221
135	131	205	144	115	246	157	191	82	216	200	198	129	111	19	99
233	167	159	188	41	249	47	180	120	6	231	113	212	171	136	141
114	185	248	172	54	42	60	241	64	211	187	67	21	173	119	128
130	236	39	229	133	53	12	65	239	147	25	33	14	78	101	189
184	143	235	206	48	95	197	26	225	202	71	61	1	214	86	77
13	102	204	45	18	32	177	153	76	194	126	5	183	49	23	215
88	97	27	28	15	22	24	34	68	178	181	145	8	168	252	80
208	125	137	151	91	149	255	210	196	72	247	219	3	218	63	148
92	2	74	51	103	243	127	226	155	38	55	59	150	75	190	46
121	140	110	142	245	182	253	89	152	106	70	186	37	66	162	250
7	85	238	10	73	104	56	164	40	123	201	193	227	244	199	158

Tabel 6 S-Box 4

Fungsi FL didefinisikan sebagai berikut:

FL:L*L → L

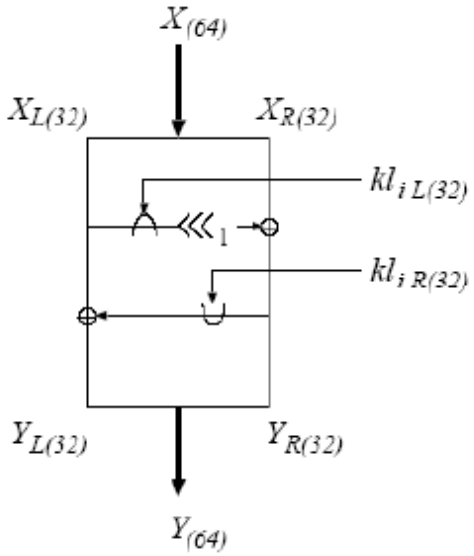
$(X_{L(32)} || X_{R(32)}, kl_{L(32)} || kl_{R(32)}) \rightarrow Y_{L(32)} || Y_{R(32)}$

Dimana

$Y_{R(32)} = (X_{L(32)} \text{ AND} kl_{L(32)}) <<<1 \text{ XOR } X_{R(32)}$

$Y_{L(32)} = (Y_{R(32)} \text{ OR} kl_{R(32)}) \text{ XOR } X_{L(32)}$

Fungsi FL diperlihatkan pada gambar 10:



Gambar 10 Struktur FL Function

Yang terakhir fungsi FL^{-1} didefinisikan sebagai berikut:

FL^{-1}:L*L → L

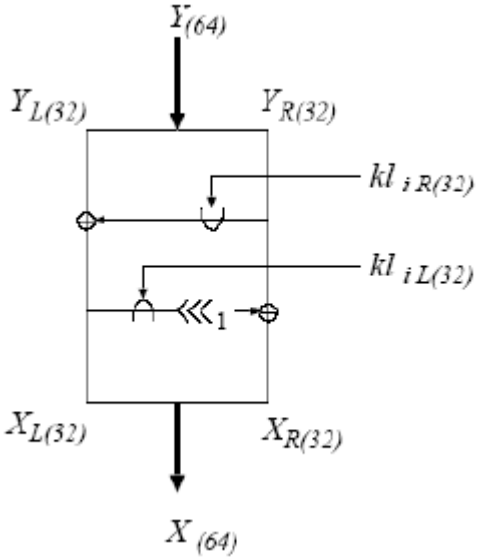
$(Y_{L(32)} || Y_{R(32)}, kl_{L(32)} || kl_{R(32)}) \rightarrow X_{L(32)} || X_{R(32)}$

Dimana

$X_{R(32)} = (X_{L(32)} \text{ AND} kl_{L(32)})<<<1 \text{ XOR } Y_{R(32)}$

$X_{L(32)} = (Y_{R(32)} \text{ OR} kl_{R(32)}) \text{ XOR } Y_{L(32)}$

Fungsi FL^{-1} ini diperlihatkan pada gambar 11:



Gambar 11 Struktur FL-1 Function

Jadi sebenarnya fungsi FL dan FL^{-1} merupakan fungsi yang saling berkebalikan.

Analisis Algoritma Blowfish

Algoritma Blowfish memiliki desain yang sangat sederhana dan mudah dipahami. Algoritma ini mudah untuk diimplementasikan karena menggunakan modifikasi Feistel Cipher serta S-box dan P-array yang sederhana. Penggunaan operasi XOR, penjumlahan, serta substitusi juga sangat efisien ketika diimplementasikan karena tidak membutuhkan resource, dalam hal ini memori komputer, dalam ukuran yang besar.

Pembangkitan sub-kunci algoritma Blowfish juga dianggap cukup aman karena berdasarkan digit konstanta pi, dimana angka-angka dalam bilangan pi merupakan angka-angka yang sangat acak dan tidak memiliki pola.

Penggunaan empat buah S-box juga terbukti meningkatkan segi keamanan algoritma Blowfish. Hal ini dapat menghindari kesimetrisan dua buah input F-function, misalnya ketika input yang satu merupakan permutasi dari input yang lain.

Selain itu penggunaan Feistel Cipher sebanyak 16 putaran juga menyebabkan tiap-tiap bit akan sangat mempengaruhi bit-bit lainnya. Subkunci algoritma Blowfish juga saling mempengaruhi satu sama lain karena dibangkitkan dengan menggunakan algoritma Blowfish itu sendiri. Selain itu, apabila dibandingkan dengan algoritma DES, algoritma Blowfish memiliki banyak keunggulan, antara lain panjang kunci yang lebih bervariasi dan jumlah subkunci yang lebih banyak.

Namun, panjang blok algoritma Blowfish dianggap terlalu pendek karena hanya berukuran 64 bit. Panjang blok 64 bit ini dianggap tidak aman karena akan melemahkan informasi plainteks ketika enkripsi lebih dari 2^{32} blok data. Pada beberapa operasi juga blok data yang terlalu pendek ini mudah diserang dengan *birthday attack.*

Akan tetapi, di dalam disertasi Vincent Rijmen, disebutkan bahwa metode *second-order differential attack* maupun *known-plaintext attack* hanya dapat memecahkan empat buah putaran Blowfish. Hingga saat ini belum ada serangan yang cukup efektif untuk memecahkan algoritma Blowfish selain dengan menggunakan *Exhaustive Search Attack*.

Analisis Algoritma Camellia

Algoritma Camellia memiliki desain yang rumit dan tidak mudah dipahami. Algoritma ini dirancang untuk tujuan kriptografi serius, karena itu operasi-operasi yang terjadi di dalamnya sangatlah rumit apabila dibandingkan algoritma Blowfish.

Algoritma Blowfish beroperasi di dalam mode blok 128 bit, sehingga proses enkripsi/dekripsi relatif akan lebih cepat dibandingkan algoritma lain yang beroperasi dalam blok 32 bit atau 64 bit.

Jumlah sub-kunci algoritma Camellia sangat banyak. Total ada 34 buah sub-kunci untuk kunci 192 bit atau 256 bit, serta 26 buah untuk kunci 128 bit. Jumlah sub-kunci yang sangat banyak ini menyebabkan algoritma Camellia sangat sukar untuk ditembus.

Algoritma Camellia juga menggunakan operasi XOR, penjumlahan, dan substitusi yang lebih kompleks apabila dibandingkan dengan algoritma Blowfish, sehingga algoritma ini membutuhkan memori yang lebih besar.

Modifikasi Feistel Cipher dengan 18 putaran (untuk kunci 128 bit) serta 24 putaran (untuk kunci 192 bit atau 256 bit) menyebabkan bit-bit yang ada benar-benar diacak dan memiliki keterkaitan yang erat antara satu bit dengan bit lainnya.

Hal yang merupakan keunggulan utama dari algoritma Camellia adalah memiliki tiga buah fungsi internal F-function, FL, dan FL^{-1}. Masing-masing fungsinya juga memiliki kompleksitas yang cukup rumit. Misalnya pada F-function input akan dipecah menjadi delapan bagian kemudian masing-masing akan mengalami substitusi dengan empat macam S-box yang berbeda dan setelah itu akan mengalami operasi XOR yang cukup rumit satu sama lain.

Sekuritas algoritma Camellia telah diakui oleh masyarakat kriptografi internasional. Algoritma Camellia diyakini merupakan alternatif pengganti algoritma AES (Advanced Encryption Standard). Salah satu penggunaan umum algoritma Camellia ialah sebagai algoritma pengamanan Transport Layer pada model OSI sistem terdistribusi. Implementasi

algoritma ini misalnya pada browser Mozilla Firefox versi 3.0 keatas yang telah mendukung SSL.

Analisis perbandingan kedua algoritma

Pada bagian ini saya akan mencoba menganalisis perbandingan algoritma yang satu dibandingkan algoritma yang lain.

Kelebihan Algoritma Blowfish:

- Desain algoritma Blowfish mudah untuk dipahami dan digunakan.
- Ketika diimpelementasikan, algoritma Blowfish tidak menggunakan ruang memori dalam jumlah besar.
- Pembangkitan subkunci benar-benar acak karena menggunakan digit konstanta pi dalam heksadesimal.
- Sekuritas cukup tinggi karena menggunakan P-array untuk membangkitkan subkunci dan 4 buah S-box untuk operasi substitusi.
- *Free license.*

Kekurangan Algoritma Blowfish:

- Panjang blok dianggap terlalu pendek, yaitu hanya 64 bit.
- Semakin berkembangnya teknologi prosesor menyebabkan Algoritma Blowfish rentan terhadap serangan kriptanalis karena desainnya terlalu sederhana.

Kelebihan Algoritma Camellia:

- Desain yang sangat kompleks.
- *Throughput* yang tinggi, karena algoritma Camellia beroperasi pada mode blok 128 bit, jauh lebih banyak dibandingkan algoritma Blowfish.
- Sekuritas sangat tinggi, karena mennggunakan banyak sekali operasi XOR, penjumlahan, dan penggeseran.
- Memiliki tiga buah fungsi internal yaitu F-function, FL function, dan FL^{-1} function.
- *International recognized*, sudah banyak diimplementasikan salah satunya untuk keamanan *transport layer* pada sistem terdistribusi

- Penelitian membuktikan algoritma Camellia memiliki resistansi yang cukup tinggi terhadap berbagai serangan kriptanalis, antara lain *Differential and Linear Cryptanalysis, Truncated Differential and Linear Cryptanalysis, Boomerang Attack, Square Attack*, dan lain-lain.

Kekurangan Algoritma Camellia:

- Desain algoritma Camellia sukar untuk dipahami.
- Sukar untuk diimpelementasikan, terutama oleh masyarakat awam.
- Kompleksitas tinggi menyebabkan algoritma ini menggunakan ruang memori yang lebih besar.

Kesimpulan

1. Algoritma Camellia memiliki sekuritas yang lebih baik serta kompleksitas yang lebih tinggi dibandingkan algoritma Blowfish.
2. Panjang blok yang lebih besar menyebabkan *throughput* algoritma Camellia lebih baik dan tidak mudah ditembus oleh kriptanalis.
3. Algoritma Blowfish mudah untuk diimplementasikan karena memiliki desain yang lebih sederhana.
4. Algoritma Blowfish lebih cocok untuk keperluan kriptografi yang tidak serius karena mudah untuk dipahami.

REFERENSI

Kazumaro Aoki, dkk. 2002. *Camellia: A 128 bit Block Cipher Suitable for Multiple Platforms*.

http://info.isl.ntt.co.jp/camellia diakses tanggal 20 Mei 2011 pukul 22.00 WIB

http://www.schneier.com/blowfish.html diakses tanggal 20 Mei 2011 pukul 21.00 WIB