Blackfish : Block cipher dengan *Key-Dependent* S-Box dan P-Box

Adhitya Ramadhanus / 13511032
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511032@std.stei.itb.ac.id

Farid Firdaus / 13511091
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13511091@std.stei.itb.ac.id

***Abstract*—Pada saat ini, Belum banyak block cipher berbasis key-dependent S-Box dan key-dependent P-Box yang dikembangkan meskipun secara teoritis block cipher jenis ini aman dari linear cryptanalysis dan differential cryptanalysis. Pada Makalah ini dikembangkan sebuah block cipher berbasis feistel network dengan key-dependent S-Box dan key-dependent P-Box yang memiliki ukuran blok 128 bit, ukuran kunci 128 bit dan jumlah ronde 8. Block cipher ini dinamakan Blackfish.**

Keywords—Block cipher; Key-dependent S-Box; Key-Dependent P-Box; Feistel-Network;

I. PENDAHULUAN

Jumlah pertukaran informasi yang ada pada saat ini sangatlah banyak dan jumlah ini terus meningkat mengingat semakin banyaknya media pertukaran informasi seperti media sosial. Hal ini tentunya meningkatkan kebutuhan akan keamanan informasi. Keamanan informasi dapat diperoleh melalui penerapan kriptografi. Kriptografi (dalam konteks ini adalah block cipher) dapat memberikan *security property* seperti confidentiality, integrity, authentication dan non-repudiation.

Komponen utama dari block cipher adalah S-Box, P-Box dan operasi penggabungan key dengan plaintext. S-Box berfungsi seperti cipher substitusi dan P-Box berfungsi seperti transposition cipher. Selain itu, S-Box dan P-Box adalah komponen block cipher yang menghasilkan confusion dan diffusion. Confusion dan diffusion adalah salah satu karakteristik dari suatu block cipher yang memiliki tingkat keamanan yang baik. Block cipher seperti DES[6] dan AES[7] memiliki S-Box dan P-Box yang didesain untuk menghasilkan karakteristik seperti transformasi non-linear agar menghasilkan keamanan yang baik. Block cipher lain seperti Twofish, Blowfish, Khufu, DSDP memiliki S-Box dan/atau P-Box yang dihitung atau dihasilkan berdasarkan kunci yang digunakan untuk enkripsi.

Twofish[1] adalah block cipher berbasis feistel network dengan ukuran blok 128 bit dan ukuran key bervariasi hingga maksimum 256 bit. *Building block* dari Twofish adalah S-Box (Key-dependent) , MDS Matrix (Maximum Distance Separable) , PHT (Pseudo-Hadamard Transform) dan proses Whitening. Twofish didesain untuk memenuhi kriteria AES (Advanced Encryption Standard) seperti ukuran block harus 128 bit, mampu menerima ukuran kunci 128 bit, 192 bit dan 256 bit dan efisien baik dalam implementasi hardware maupun software.

Khufu[2] adalah block cipher yang dikembangkan oleh Ralph C. Merkle dengan *goal design* adalah block cipher yang dapat diimplementasikan pada software secara efisien. Block cipher ini memiliki ukuran blok 64 bit, ukuran kunci 512 dan jumlah ronde yang dapat diatur user (disarankan jumlah ronde yang digunakan adalah kelipatan 8). S-Box yang digunakan dalam block cipher ini adalah key-dependent S-Box yang berubah setiap 8 ronde.

DSDP[3] adalah nama struktur sekaligus nama sebuah block cipher yang menggunakan key-dependent S-Box dan key-dependent P-Box. DSDP memiliki ukuran blok 128 bit , ukuran kunci bervariasi namun umumnya lebih dari 128 bit dan jumlah ronde yang bervariasi. Hal ini membuat block cipher ini cukup fleksibel dalam hal trade-off antara kecepatan dan keamanan. S-Box dan P-Box pada DSDP dihasilkan oleh algoritma key scheduling milik stream cipher RC4.

Pada tahun 1990, E.Biham dan Shamir mempublikasikan suatu teknik kriptanalisis bernama differential cryptanalysis. Differential Cryptanalysis adalah serangan berjenis chosen-plaintext dimana attacker dapat menentukan plaintext yang akan dienkripsi. Setelah Differential cryptanalysis, muncul linear cryptanalysis yang dirilis oleh Matsui. Linear cryptanalysis adalah serangan berjenis known-plaintext dimana attacker memiliki sekumpulan pasangan ciphertext dan plaintext. Kedua serangan ini menitik beratkan pada analisis S-Box sebagai sumber non-linearitas dalam block cipher[4]. Alhasil, dengan S-Box yang tidak diketahui oleh attacker maka kompleksitas serangan berbasis linear cryptanalysis dan differential cryptanalysis akan meningkat atau tidak mungkin dilakukan sama sekali karena cost yang terlalu besar dibandingkan nilai informasi yang didapat.

II. DASAR TEORI

A. Block Cipher

Block cipher adalah cipher yang menerima input berupa blok-blok plaintext. Blok-blok ini nantinya akan dienkripsi sesuai dengan transformasi yang didefinisikan pada block cipher tersebut. Transformasi suatu block cipher secara umum

dikendalikan oleh suatu kunci sehingga block cipher juga dapat didefinisikan sebagai pemetaan suatu plaintext pada message space ke suatu ciphertext (ciphertext bisa jadi terdapat pada message space yang sama dengan plaintext) berdasarkan suatu kunci.

Dua prinsip dasar pada desain block cipher ada confusion dan diffusion. Diffusion berarti penyebaran pengaruh suatu bit plaintext pada ciphertext untuk menyembunyikan karakteristik statistic suatu plaintext. Dengan kata lain, perbedaan satu bit input dapat menyebabkan perubahan ciphertext yang signifikan. Cara sederhana yang dapat dilakukan untuk menghasilkan diffusion adalah permutasi plaintext pada level tertentu (byte/bit). Confusion menyembunyikan hubungan antara plaintext dan ciphertext. Confusion dapat dihasilkan melalui transformasi substitusi. Operasi-operasi sederhana seperti substitusi dan permutasi jika dilakukan berkali-kali pada suatu blok plaintext dapat menghasilkan confusion dan diffusion yang baik yang implikasinya adalah tingkat keamanan yang lebih baik untuk block cipher tersebut. Hal inilah yang mendasari desain *iterated cipher* untuk suatu block cipher.

B. Struktur Block Cipher

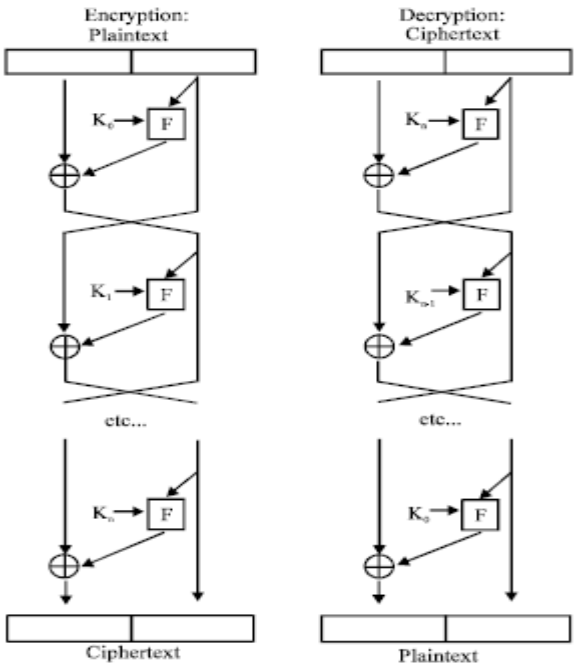
Secara umum, terdapat 2 jenis struktur block cipher yaitu SPN (Substitution Permutation Network) dan Feistel Network. Contoh block cipher berbasis SPN adalah AES dan contoh block cipher berbasis Feistel Network adalah DES,Twofish,Blowfish,Camellia, dll. Pada umumnya, SPN memiliki 4 jenis transformasi yaitu transformasi substitusi, transformasi permutasi, transformasi penggabungan linear dan transformasi penambahan kunci.

Feistel Network adalah salah satu jenis struktur block cipher dimana pada balanced feistel network, plaintext dibagi menjadi 2 bagian sama panjang yang akan diproses secara berbeda. Pada dasarnya, blok plaintext dibagi menjadi dua blok yaitu L dan R. L dan R untuk round ke-i didefinisikan pada persamaan 1 dan 2.

$$L_{i+1} = R_i \quad (1)$$

$$R_{i+1} = L_i \oplus F(R_i,K_i) \quad (2)$$

Setelah itu, Ciphertext didefinisikan sebagai (R_{n+1},L_{n+1}) dengan n adalah jumlah *round* untuk block cipher tersebut. $F(R_i,K_i)$ adalah round function yang didefinisikan oleh suatu block cipher. Feistel network memiliki kelebihan yaitu round function yang digunakan tidak harus *invertible*. Hal ini disebabkan pada proses dekripsi, round function yang digunakan pada saat enkripsi digunakan kembali pada saat dekripsi hanya saja dengan urutan key yang dibalik. Struktur feistel network dapat dilihat pada Gambar II.1.



Gambar II.1 Struktur Feistel Network (Sumber : http://www.medwelljournals.com/fulltext/?doi=ijscomp.2009.131.135)

C. Key-Dependent S-Box dan *Key-Dependent P-Box*

Key-Dependent S-Box adalah S-Box yang digenerate berdasarkan kunci enkripsi, dalam artian proses pembuatan S-Box tersebut dikendalikan oleh kunci enkripsi. Terdapat beberapa cara untuk menghasilkan Key-Dependent S-Box. Pada struktur DSDP[3], S-Box dihasilkan lewat permutasi yang dihasilkan oleh algoritma key scheduling RC4. Stoianov[5] mengusulkan cara pembuatan Key-Dependent S-Box untuk AES dengan melakukan xor setiap elemen pada S-Box AES dengan MSB Key lalu selanjutnya S-Box tersebut digunakan pada setiap enkripsi AES.

Key-Dependent P-Box adalah P-Box yang digenerate berdasarkan kunci enkripsi. Salah satu cara untuk menghasilkan Key-Dependent S-Box adalah dengan menggunakan algoritma key scheduling RC4.Cara ini dilakukan oleh DSDP cipher.

III. DESAIN BLOCK CIPHER BLACKFISH

Blackfish didesain untuk memiliki key-dependent S-Box dan Key-Dependent P-Box untuk menghasilkan ketahanan terhadap linear cryptanalysis dan differential cryptanalysis. Black Fish memiliki ukuran blok 128 bit, ukuran kunci 128 bit dan jumlah ronde 8. Pada setiap ronde, S-Box dan P-Box yang digunakan dalam round function tersebut di*generate* menggunakan kunci yang digunakan pada ronde tersebut.

Black Fish memiliki beberapa komponen utama antara lain:

1. Key-Dependent S-Box
2. Key-Dependent P-Box

3. Pseudo-Hadamard Transformation

Selain itu, digunakan pula operasi rotasi bit dan operasi *addkey* yang melakukan xor byte plaintext dengan key.

A. *Spesifikasi Blackfish*

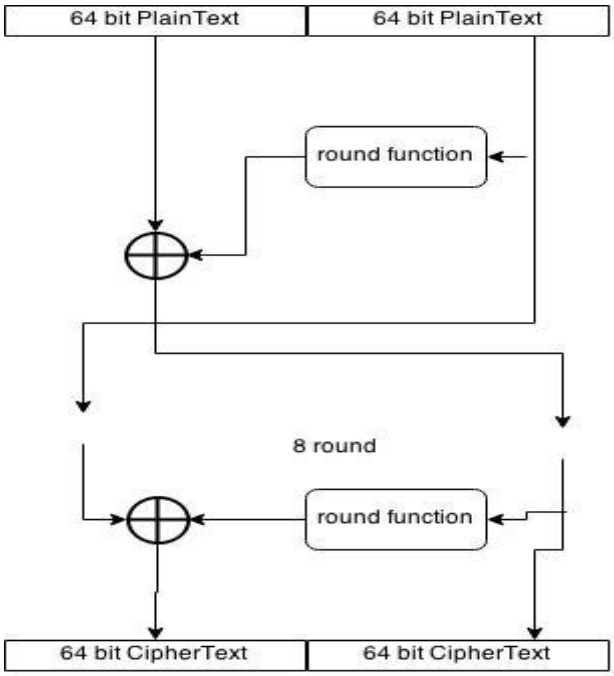
Blackfish memiliki spesifikasi sebagai berikut :

1. Ukuran blok : 128 bit
2. Ukuran key : 128 bit
3. Jumlah round : 8
4. Struktur : Feistel Network

Notasi yang digunakan pada makalah ini untuk mendefinisikan blackfish maupun hal lain adalah sebagai berikut :

1. A $\oplus$ B , operasi exclusive or A dan B
2. A <<< B, operasi rotasi A sebanyak B bit
3. K, KL dan KR, K adalah kunci yang digunakan untuk suatu round, K dibagi menjadi 2 bagian sama panjang (64 bit) yaitu KL dan KR

Struktur blackfish dapat dilihat pada Gambar III.1.



Gambar III. Struktur Blackfish

B. *Key-Dependent S-Box*

Cara yang digunakan untuk menghasilkan Key-Dependent S-Box adalah cara yang diusulkan Stoianov[5] dengan sedikit modifikasi. Key-Dependent S-Box dihasilkan melalui operasi xor byte KL[0] dengan setiap element pada S-Box basis. S-Box digenerate pada setiap round blackfish untuk menghasilkan kombinasi S-Box yang semakin banyak sehingga mempersulit linear cryptanalysis maupun differential cryptanalysis. S-Box dasar yang digunakan pada block cipher ini dapat dilihat pada Tabel III.1. S-Box ini adalah S-Box yang dikembangkan oleh Asim & Jeoti [8] yang berbasis chaotic maps. S-Box ini memiliki differential approximation probability yang cukup kecil dan linear approximation probability yang cukup kecil juga sehingga cukup aman terhadap linear cryptanalysis dan differential cryptanalysis. S-Box ini menerima input 8 bit integer dan menghasilkan 8 bit integer juga. Elemen pada Tabel III.1 berisi nilai hasil substitusi input. Input 0 akan disubstitusi menjadi 50 sedangkan 1 akan disubstitusi menjadi 243, dst.

C. *Key-Dependent P-Box*

Key-Dependent P-Box dihasilkan dengan menggunakan algoritma key scheduling yang dimiliki oleh stream cipher RC4. Algoritma ini dipilih karena sederhana namun menghasilkan permutasi yang baik. Permutasi ini dikendalikan oleh suatu kunci sehingga P-Box ini termasuk dalam Key-Dependent P-Box. Pseudocode dari algoritma key scheduling RC4 yang digunakan pada Blackfish dapat dilihat pada Gambar III.2.

```
for i from 0 to 8
    S[i] := i
endfor
j := 0
for i from 0 to 8
    j := (j+S[i]+key[i mod 8]) mod 256
    swap values of S[i] and S[j]
endfor
for i from 0 to 8
    ciphertext[i] := plaintext[S[i]]
endfor
```

Gambar III.2 Pseudo-code RC4 Key Scheduling

D. *Pseudo-Hadamard Transformation*

Pada dasarnya, Pseudo-Hadamard Transformation menerima input bit string dengan panjang genap lalu membagi input tersebut menjadi 2 bagian sama panjang dan memprosesnya. Pseudo-Hadamard Transformation pernah digunakan dalam block cipher twofish dan diketahui dapat memberikan *diffusion* pada suatu block cipher. Transformasi ini dipilih karena dapat digunakan sebagai sumber *diffusion* dan sangat sederhana dalam implementasinya. Pseudo-Hadamard Transformation dapat didefinisikan pada persamaan 3 dan 4.

$$a' = a + b \pmod{2^n} \quad (3)$$

$$b' = a + 2b \pmod{2^n} \quad (4)$$

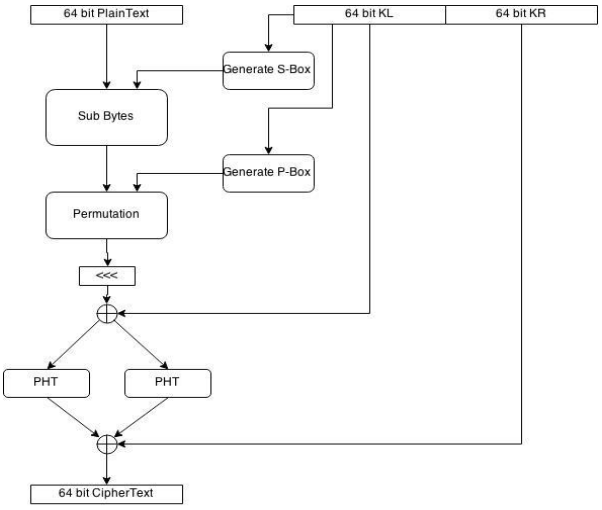
a dan b pada persamaan 3 dan 4 adalah bit string yang memiliki panjang sama yaitu n.

Tabel III.1 S-Box dasar Blackfish

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

E. Round Function

Round function dari blackfish dapat dilihat secara visual pada gambar III.3.



Gambar III.3 Blackfish round function

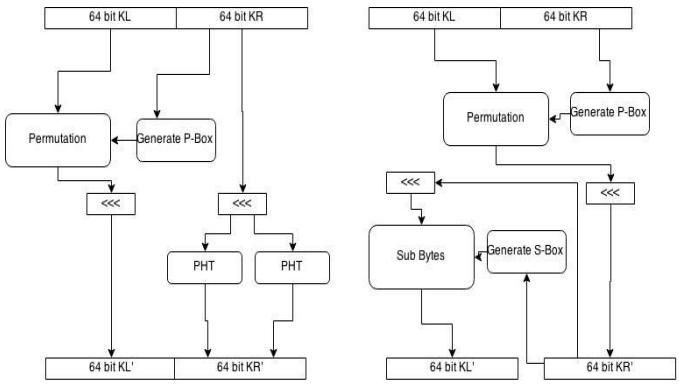
Input yang dibutuhkan oleh round function blackfish adalah plaintext berukuran 64 bit dan kunci berukuran 128 bit. Sebelum memulai proses enkripsi plaintext, digenerate terlebih dahulu S-Box dan P-Box menggunakan KL. Proses generasi S-Box dan P-Box dapat dilakukan didalam round function (dalam loop utama) maupun dilakukan sebelum round function (di luar loop utama) sesaat setelah proses key scheduling blackfish selesai. Setelah proses generasi S-Box dan P-Box selesai maka selanjutnya adalah operasi SubBytes yaitu proses substitusi plaintext menggunakan S-Box dalam level bytes. Pada operasi ini plaintext diproses sebagai array of bytes yang berukuran 8. Proses permutasi adalah proses pengacakan plaintext pada level bytes menggunakan P-Box yang dihasilkan pada proses *generate* P-Box. P-Box dalam block cipher didefinisikan sebagai urutan bytes plaintext setelah pengacakan. Misalkan input proses permutasi adalah (18,0,64,54,79,81,90,108) dan P-Box yang digunakan adalah (4,1,0,7,6,5,3,2) maka ciphertext yang dihasilkan adalah (79,0,18,108,90,81,54,64). Output hasil permutasi selanjutnya akan dirotasi dengan jumlah pergeseran 7 bit. Hal ini bertujuan untuk menghasilkan pengacakan pada level bit. Setelah itu operasi addkey(xor) akan dilakukan pada plaintext dan KL.

Proses terakhir sebelum addkey terakhir adalah Pseudo-Hadamard Transformation. Pada proses ini, plaintext dibagi menjadi 2 buah bit string sama panjang berukuran 32 bit yang masing-masing string tersebut akan menjadi input pseudo-hadamard transformation. Setelah itu proses addkey terakhir adalah proses addkey menggunakan KR. Proses round function blackfish secara keseluruhan dapat dilihat dalam pseudo-code berbasis bahasa c++ berikut:

```
//Black fish round function , PlainText is 8
bytes (64 bit) and Key is 16 bytes (128 bit)
CipherText : Array of unsigned char
KL,KR : Array of unsigned char
CipherL,CipherR : Array of unsigned char
NewBox : Array of unsigned char

//Generate New S-Box
NewBox := GenerateNewSBox(KL[0])
//Substitution
CipherText := SubBytes(PlainText,NewBox)
//Permutation
CipherText := RC4KeySched (CipherText,KL)
//Bit rotation
CipherText := RotLeft(CipherText,7)
//AddKey
CipherText = AddKey(CipherText,KL)
//Pseudo-Hadamard Transformation
CipherL := CipherText[0..3]
CipherR := CipherText[4..7]

CipherL := PHT(CipherL)
CipherR := PHT(CipherR)

CipherText := (CipherL || CipherR) //concat

CipherText := AddKey(CipherText,KR)
```

F. Key Scheduling

Key scheduling yang digunakan blackfish memanfaatkan operasi-operasi maupun komponen yang ada pada blackfish seperti operasi rotasi, pseudo-hadamard transformations maupun sub bytes. Key scheduling pada blackfish berfungsi untuk menghasilkan 128*8 bit kunci yang dibutuhkan pada proses enkripsi. Oleh karena itu, Key Scheduling ini teridiri atas 8 ronde. Key scheduling blackfish dapat dilihat pada Gambar xx.



(a) even key (b) odd key

Key scheduling pada blackfish dapat dibedakan menjadi 2 jenis yaitu key scheduling untuk odd key (key untuk round ganjil) dan key scheduling untuk even key (key untuk round genap). Pada key scheduling untuk even key, KR digunakan untuk menghasilkan P-Box yang digunakan pada proses permutasi. Selanjutnya output hasil permutasi tadi dirotasi sebanyak n-1 bit , 2<=n<=8 dimana n adalah ronde key scheduling , hasil rotasi ini adalah KL'. Untuk menghasilkan KR', pertama-tama KR dirotasi sebanyak n-1 bit lalu KR dibagi menjadi 2 buah bit string sama panjang yang ditransformasi menggunakan pseudo-hadamard transformations.

Pada key scheduling untuk odd key, KR digunakan untuk menghasilkan P-Box yang digunakan pada proses permutasi KL. Hasil permutasi akan dirotasi sebanyak n-1 bit untuk menghasilkan KR'. KR' akan digunakan untuk menghasilkan S-Box yang digunakan apda proses substitusi bytes. Input substitusi bytes adalah KR' yang dirotasi sejauh n-1 bit. Hasil substitusi adalah KL'.

Proses key scheduling blackfish secara keseluruhan dapat dilihat dalam pseudo-code berbasis bahasa c++ berikut :

```
Keys,MasterKey,KL,KR : array of unsigned char
NewBox : array of unsigned char
Keys[0..15] := MasterKey[0..15]
For i:=1 to 7 do
   KL := Keys[(i-1)*16..(i-1)*16+8]
   KR := Keys[(i-1)*16+8..(i-1)*16+16]
   If (i is odd)
       KL := RotLeft(RC4KeySched(KL,KR),i)
       KR := (PHT(KR[0..3]) || PHT(KR[4..7]))
   Else
       KR := RotLeft(RC4KeySched(KL,KR),i)
       NewBox := GenerateNewSBox(KR[0])
       KL := SubBytes(RotLeft(KR,i),NewBox)
   Keys[i*16..i*16+16] := KL || KR
endfor
```

Perlu diperhatikan, pada pseudo-code diatas, iterasi dimulai saat variabel i bernilai 1 hal ini berarti saat variabel i bernilai ganjil maka proses key scheduling yang digunakan adalah key scheduling untuk even key. Hal ini dilakukan dengan tujuan mempermudah pemindahan kunci hasil key scheduling ke kunci utama yang nantinya akan digunakan pada enkripsi.

IV. PERFORMANSI DAN ANALISIS HASIL

Implementasi blackfish dilakukan menggunakan bahasa c++ dengan compiler g++ (tdm-1) 4.7.1. Pengujian algoritma dilakukan pada laptop ASUS A46CB dengan spesifikasi :

1. Operating System : Windows 7
2. Processor : Intel® Core ™ i3-3217 CPU @ 1.80 GHz
3. Memory : 4096 MB RAM

Pengujian dilakukan dengan menggunakan 3 buah plaintext yang berbeda ukuran yaitu 1 blok (16 byte), 10000 blok (157 KB) dan 1000000 blok (1.565 KB). Selain itu pengujian terdiri atas 3 jenis yaitu pengujian untuk mode ECB, CBC dan CFB 8-bit. Cakupan pengujian adalah waktu eksekusi enkripsi dalam milliseconds. Detil Pengujian dapat dilihat pada Tabel IV.1.

Tabel IV.1 Hasil Pengujian Blackfish

Mode	1 Blok (milliseconds)	10000 Blok (milliseconds)	1000000 Blok (milliseconds)
ECB	1	2045	20081
CBC	1	2346	20074
CFB 8-bit	4	32423	325521

Dapat dilihat pada Tabel IV.1, waktu eksekusi blackfish ECB dan CBC tidak jauh berbeda karena enkripsi setiap blok dilakukan secara sekuensial sedangkan CFB 8-bit menghasilkan waktu eksekusi yang jauh lebih lama karena CFB mengenkripsi plaintext dalam satuan 8 bit (1 bytes) namun tetap membutuhkan waktu enkripsi seperti 1 blok (16 bytes) sehingga secara teori CFB menghasilkan waktu 16 kali lebih lama dibanding mode ECB dan CBC. Perlu dicatat, implementasi ini mungkin bukan implementasi blackfish yang paling efisien dari segi memori maupun waktu eksekusi karena banyak menggunakan fitur bahasa tingkat tinggi milik c++ dan banyak melakukan operasi penyalinan vector pada c++ sehingga hal ini dapat berpengaruh pada waktu eksekusi blackfish.

Setelah itu, Analisis selanjutnya adalah pada algoritma generasi Key-Dependent S-Box. S-Box yang dapat digunakan pada suatu block cipher adalah S-Box yang memiliki karakteristik invertability. Artinya, misalkan X_1=X_2 adalah input untuk suatu S-Box dan menghasilkan Y_1 dan Y_2 maka Y_1 harus sama dengan Y_2. Hal ini untuk memastikan bahwa tidak ada 2 atau lebih input yang menghasilkan output yang sama. Algoritma yang diusulkan Stoianov[5] dapat menghasilkan S-Box yang berfungsi dengan baik karena fakta jika X_1 tidak sama dengan X_2 dan B_1 sama dengan B_2 maka X_1' tidak sama dengan X_2' dengan $X_1' = X_1 \oplus B_1$ dan $X_2' = X_2 \oplus B_2$.

Beberapa contoh hasil enkripsi blackfish dapat dilihat pada Tabel IV.2.

Tabel IV.2 Hasil enkripsi Blackfish

PlainText (16 bytes Hex)	CipherText (16 bytes Hex)
31 32 33 34 35 31 32 33 34 35 31 32 33 34 35 31	dc 7e d3 d f0 3 15 6e d9 e c2 bd 87 a9 4 e0
31 32 33 34 35 31 32 33 34 35 31 32 33 34 35 30	80 d9 df 9a 52 4c 9 33 53 2c 69 15 42 18 2d f6
74 59 38 76 76 66 35 72 44 42 72 59 31 42 52 30	f8 99 5b 41 38 75 92 e 77 c2 59 a3 e2 17 f0 33
54 47 54 75 35 43 63 47 77 36 57 53 7a 38 54 52	56 c5 b0 88 cd c8 e3 4e 5d 5b 1a d9 2d b5 c2 b7

Dapat dilihat pada Tabel IV.1, Blok pertama dan blok kedua hanya berbeda 1 byte yaitu pada byte terakhir namun menghasilkan ciphertext yang jauh berbeda. Hal ini menandakan blackfish mampu memberikan diffusion yang baik pada ciphertext karena setidaknya terdapat 3 proses transformasi yang berfungsi memberikan diffusion yaitu permutasi dengan P-Box, rotasi bit dan Pseudo-Hadmard Transformation . Seluruh blok input yang terdapat pada Tabel IV.2 dienkripsi dengan menggunakan kunci yang sama.

Contoh key scheduling pada blackfish dapat dilihat pada output program berikut :

```
Master Key     : 0 1 2 3 4 5 6 7 8 9 a b c d e f

Expanded Key : 0 2 a e 6 c 4 8 24 28 38 3e 34 38 50 56
Expanded Key : 4a 19 fa e0 a 57 fd bf 38 30 20 28 0 8 18 10
Expanded Key : ff ef 7 d0 ca b8 52 55 c2 c1 c4 1 c0 c1 81 42
Expanded Key : ec 74 a f6 e3 64 88 75 fd 0 7e f5 2b 85 5c af
Expanded Key : 91 e 81 4e be dc 7d 8c 7e b4 5d 59 6 aa 9c a9
Expanded Key : 9 72 c8 40 35 b6 8 52 53 b7 3 af a0 63 24 5f
Expanded Key : 4 a0 64 29 1a 84 5b 39 b3 51 8b 21 61 3b 90 e4
```

```
Master Key : 14 f9 70 1a e3 5f e2 8c 44 a df 4d 4e a9 c0 26

Expanded Key : 35 19 c7 f2 29 c4 be e0 46 af 5 49 1d 9f 9d eb
Expanded Key : d3 57 f6 f5 5c 2f c fd f8 64 a7 83 10 d7 cb 1e
Expanded Key : 99 78 62 bf b7 af ea e6 ff 3d 3b 55 df b5 38 ac
Expanded Key : de 12 c0 72 51 a3 4f ff 6a fb 76 27 8e a9 9b fe
Expanded Key : 49 f4 62 5b df f8 a 2e 24 5f e9 50 55 0 d4 cd
Expanded Key : 75 8e 48 9a 56 1d 37 43 12 42 8b b7 fe 18 96 fd
Expanded Key : 9b cd 24 2b 21 8e c7 3a fd 44 d9 43 8a d4 9 5d
```

V. ANALISIS KEAMANAN

Pada bagian ini akan dianalisis tiga buah serangan yang dapat dilakukan pada suatu block cipher yaitu brute force, linear cryptanaysis dan differential cryptanalysis. Serangan-serangan ini dipilih karena cukup populer.

A. Brute Force Attack

Pada skenario terburuk, Serangan brute force atau exhaustive key search membutuhkan setidaknya 2^{128} percobaan kunci untuk menebak kunci yang digunakan pada enkripsi. Hal ini tentunya menyebabkan serangan ini tidak dapat dilakukan karena membutuhkan setidaknya 1 milyar milyar tahun untuk mencoba seluruh dengan menggunakan supercomputer tercepat dan dengan asumsi 1000 flops (floating operation per seconds) dibutuhkan untuk pengujian satu kombinasi kunci[9]. Dapat disimpulkan bahwa penggunaan kunci yang berukuran 128 bit memberikan keamanan yang cukup bagi blackfish. Hal ini karena cost yang dibutuhkan untuk melakukan serangan brute force terhadap black fish sangat besar dalam hal waktu maupun sumber daya lain.

B. Linear Cryptanalysis

Salah satu langkah penting dalam linear cryptanalysis adalah analisis S-Box untuk melakukan serangan terhadap suatu block cipher. S-Box dalam block cipher mampu

menghasilkan sifat non-linear pada block cipher tersebut, hal ini menyebabkan S-Box sangat berguna dalam proses enkripsi.

Linear cryptanalysis mencoba mencari suatu persamaan linear yang memliki bias tinggi yang menghubungkan bit-bit ciphertext, plaintext dan key. Persamaan linear dalam konteks linear cryptanalysis adalah persamaan yang terdiri atas variable biner dan operator xor (exclusive or). Suatu persamaan linear dianggap memiliki bias tinggi jika probabilitas kebenarannya jauh menyimpang dari 0.5.

S-Box yang digunakan blackfish pada setiap round enkripsi berbeda satu sama lain. Terdapat 256 kemungkinan S-Box yang digunakan dalam suatu ronde. Secara keseluruhan terapat 256x8 kombinasi S-Box yang digunakan dalam enkripsi suatu blok. Misalkan suatu persamaan linear seperti pada persamaan 5digunakan untuk memodelkan S-Box.

$$aX_1 \oplus bX_2 \oplus cX_3 \oplus dX_4 = eY_1 \oplus fY_2 \oplus gY_3 \oplus hY_4 \qquad (5)$$

Pada persamaan 1, a,b,c,d dan e,f,g,h adalah bitmask untuk input dan output X,Y sehingga jika abcd = 0010 dan efgh = 1010 maka persamaan linear yang dianalisis adalah seperti persamaan 6.

$$X_3 = Y_1 \oplus Y_3 \qquad (6)$$

Terdapat sekitar 256x256 persamaan linear yang harus dianalisis untuk setiap S-Box sehingga secara keseluruhan terdapat 256x256x256x8 atau 227 kombinasi persamaan linear untuk setiap S-Box. P-Box juga perlu dianalisis dalam linear cryptanalysis karena P-Box blackfish tidak diketahui oleh attacker. Hal ini berbeda dengan block cipher yang memiliki P-Box statis. Jumlah kombinasi P-Box yang mungkin dalam suatu proses enkripsi adalah 40320 (8!) x 8. Hal ini menyebabkan linear cryptanalysis sulit untuk dilakukan terhadap blackfish.

C. Differential Cryptanalysis

Differential cryptanalysis menyerang suatu cipher dengan mencari differential output yang sering muncul berdasarkan differential input tertentu. Differential input didefinisikan sebagai ΔX dimana $\Delta X = X' \oplus X''$, X' dan X'' adalah dua input plaintext. Secara teoritis, untuk differential input sembarang, probabilitas suatu differential output adalah $1/2^n$[4] namun pada kenyataannya terdapat pasangan differential input dan output tertentu yang memiliki probablitas kemunculan yang cukup tinggi. Hal inilah yang dimanfaatkan oleh differential cryptanalysis.

Sama halnya dengan linear cryptanalysis, hal yang perlu dianalisis pada differential crytpanalysis adalah S-Box. Pada 8-bit S-Box terdapat 256 differential input yang harus dianalisis dan setiap differential input tersebut menghasilkan 256 differential output. S-Box yang digunakan pada Blackfish adalah S-Box yang dihasilkan dengan menggunakan metode yang diusulkan oleh Asim dan Jeoti [8]. S-Box ini memiliki nilai probabilitas differential maksimum 10/256[8]. Artinya, terdapat minimal satu pasang differential input (ΔX,ΔY) yang memiliki frekuensi kemunculan 10 pada S-Box tersebut. Oleh karena terdapat 256 S-Box yang digunakan Blackfish maka terhadap 256 S-Box tersebut dilakukan kalkulasi maksimum probabilitas differential dengan persamaan 7[8].

$$DP^S(\Delta x \rightarrow \Delta y) = \left(\frac{\#\{x \in X \mid S(x) \oplus S(x \oplus \Delta x) = \Delta y\}}{2^m} \right) \qquad (7)$$

Hasil kalkulasi menunjukkan nilai maksimal probabilitas differential adalah 10/256 dari 256 S-Box yang mungkin dihasilkan dari S-Box dasar Blackfish. Dengan nilai maksimal probabilitas differential dan perlunya analisis P-Box yang jumlahnya tidak sedikit maka differential cryptanalysis sulit untuk dilakukan terhadap blackfish.

VI. KESIMPULAN DAN SARAN

Pada makalah ini telah dikembangkan block cipher yang didesain untuk memiliki ketahanan terhadap linear cryptanalysis dan differential cryptanalysis dengan menggunakan key-dependent S-Box dan key-dependent P-Box. Implementasi yang menggunakan terlalu banyak instruksi penyalinan vector dan desain fungsi yang kurang efisien pada output,parameter maupun algoritma dalam implementasi pada makalah ini bisa jadi menyebabkan lambannya waktu eksekusi blackfish. Selain itu, Banyaknya transformasi (terdapat 7 transformasi yang dilakukan pada round function blackfish) yang ada pada round function blackfish juga sedikit banyak berpengaruh pada waktu eksekusi blackfish namun hal ini tentunya dilakukan dengan tujuan meningkatkan keamanan block cipher ini..

Referensi

[1] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall dan N. Ferguson, "Twofish : A 128-bit Block Cipher", AES Submission.

[2] R. C. Merkle, "Fast Software Encryption Functions", Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'90, pp 476-501, 1990.

[3] R. Zhang, L. Chen, "A Block Cipher using Key-Dependent S-Box and P-Boxes", Industrial Electronics, ISIE 2008. IEEE International Symposium, pp. 1463-1468, 2008.

[4] H. Heys, "A tutorial on linear and differential cryptanalysis".

[5] N. Stoianov, E. Altimirski, "A new approach of generating key-dependent S-Boxes in AES".

[6] FIPS PUB 46-3, "Data Encryption Standard (DES)", National Institute of Standards and Technology, U.S. Department of Commerce, 1999. http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[7] FIPS PUB 197, Advanced Encryption Standard (AES), National Institute of Standards and Technology, U.S. Departement of Commerce, 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] M. Asim, V. Jeoti, "Efficient and simple method for designing chaotic S-Boxes", ETRI J. 30, 170-172, 2008.

[9] M. Arora, " How secure is AES against brute force attacks?", http://www.eetimes.com/document.asp?doc_id=1279619

[10] B. Schneier, "Applied Cryptography (2nd ed.) : protocols, algorithms, and source code in C",Wiley, 1996.