STUDI, IMPLEMENTASI DAN PERBANDINGAN ALGORITMA KUNCI SIMETRI *TRIPLE DATA ENCRYPTION STANDARD* DAN *TWOFISH*

Revi Fajar Marta – NIM : 13503005

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13005@students.if.itb.ac.id

Abstraksi

Makalah ini membahas perbandingan antara algoritma Twofish dan 3DES untuk menyandikan data. Twofish merupakan algoritma yang dirancang oleh sebuah tim dari USA yang diketuai oleh Bruce Schneier. Algoritma ini adalah salah satu kandidat finalis sayembara AES. Twofish mendukung panjang kunci 128-bit, 192-bit, dan 256-bit. Sementara itu 3DES adalah salah satu algoritma enkripsi yang cukup populer dan merupakan perkembangan dari DES. 3DES menggunakan panjang kunci 168 bit.

Algoritma twofish banyak diterapkan pada berbagai aplikasi untuk mengenkripsi data, seperti Safe, XPDFViewer, TrueCrypt, dan lain sebagainya. Kebanyakan apikasi yang menggunakan twofish adalah aplikasi pengenkripsi file. Twofish juga dapat diimplementasikan menggunakan berbagai bahasa, seperti Delphi, C, Java, Perl, VisualBasic, dan lain sebagainya. Sementara itu, 3DES dapat diimplementasikan dalam bentuk perangkat keras, yaitu *chip* mikroprosesor. Salah satu yang cukup populer adalah prosesor IBM yang digunakan untuk mesin ATM.

Secara kinerja twofish tidak berbeda jauh dari 3DES. Metode yang baik diperlukan agar hasil ujinya akurat.

Kata kunci: algoritma kunci simetri, Triple Data Encryption Standard, Twofish Encryption Algorithm.

1. Pendahuluan

Pengiriman dan penyimpanan data melalui media elektronik membutuhkan proses dengan tingkat keamanan data yang tinggi. Untuk menjamin keamanan dan keutuhan data, dilakukan enkripsi dan dekripsi data.

Enkripsi data adalah proses penyandian data atau pengubahan data asli menjadi data rahasia. Sementara dekripsi data adalah proses pengubahan kembali data rahasia menjadi data asli. Data yang dikirimkan melalui media elektronik adalah data rahasia hasil enkripsi. Data asli dapat diketahui oleh penerima menggunakan kunci rahasia. Dengan menggunakan enkripsi dan dekripsi, data yang diolah dapat dihindarkan dari campur tangan pihak-pihak yang tidak berhak atas isi data tersebut.

Salah satu algoritma penyandian yang pernah menjadi standar adalah *Data Encryption Standard* (DES). Sejak tahun 1977, algoritma ini telah dipakai luas. Algoritma ini menggunakan panjang kunci 56-bit. Namun, dengan perkembangan kecepatan perangkat keras algoritma ini terbukti tidak aman dan tidak mencukupi lagi. Kunci DES dapat diketahui menggunakan perangkat keras khusus dalam waktu yang tidak lama. Untuk memperkuat lagoritma DES digunakan 3DES, yaitu algoritma DES yang dipakai 3 kali dalam satu penyandian. Dengan pengulangan 3 kali ini, 3DES lebih kuat menahan serangan *brute force* daripada DES.

Akan tetapi, laju perkembangan perangkat keras tidak akan dapat dihambat oleh algoritma 3DES. Suatu hari 3DES akan dapat pula dipecahkan dalam waktu singkat. Untuk itu diperlukan suatu standar algoritma baru.

Pada tahun 1997, *U.S. National Institute of Standards and Technology* (NIST) mengumumkan sayembara untuk menemukan algoritma penyandian baru yang kelak akan diberi nama *Advanced Encryption Standard* (AES). Setelah melalui seleksi panjang, beberapa algoritma yang menjadi finalis antara lain:

1. Rijndael, olehVincent Rijmen, Joan Daemen dari Belgia
2. Serpent, oleh Ross Anderson, Eli Biham, dan Lars Knudsen, dar Inggris, Israel, dan Norwegia
3. Twofish, oleh tim yang diketuai oleh Bruce Schneier, dari USA
4. RC6, dari laboratorium RSA, USA
5. MARS, dari IBM.

Pada bulan Oktober 2000, NIST mengumumkan untuk memilih Rijndael menjadi algoritma AES. Saingan algoritma Rijndael yang terbesar salah satunya adalah Twofish. Meskipun tidak terpilih, algoritma ini dapat dikategorikan kuat dan cukup banyak dipakai oleh *software-software* enkripsi data.

2. TDES dan Twofish

2.1 *Triple Data Encryption Standard* (TDES)

Algoritma DES

3DES menggunakan algoritma yang sama dengan DES, hanya saja prosesnya diulang 3 kali.

Skema global dari algoritma DES adalah sebagai berikut:

1. Blok plainteks dipermutasi dengan permutasi awal (IP, *Initial Permutation*).
2. Hasil permutasi awal kemudian di-*enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.
3. Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^{-1}) menjadi blok cipherteks.

Di dalam proses *enciphering*, blok plainteks dibagi manjadi dua bagian, kiri (L) dan kanan (R), yang masing-masing panjangnya 32-bit. Kedua bagian ini masuk ke dalam 16 putaran DES. Pada setiap putaran *i*, blok *R* merupakan masukan untuk fungsi transformasi yang disebut fungsi *f*. Pada fungsi *f*, blok *R* dikombinasikan dengan kunci internal K_i. keluaran dari fungsi *f* di-*XOR*-kan dengan blok *L* untuk mendapatkan blok *R* yang baru. Sedangkan blok *L* yang baru langsung diambil dari blok *R* sebelumnya. Ini merupakan satu putaran *DES*. Secara matematis, satu putaran *DES* dinyatakan sebagai:

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

Satu putaran DES merupakan model jaringan Feistel. Perlu dicatat bahwa jika (L_{16}, R_{16}) merupakan keluaran dari putaran ke-16, maka (R_{16}, L_{16}) merupakan pra-cipherteks (*pre-ciphertext*) dari *enciphering* ini. Cipherteks yang sebenarnya diperoleh dengan melakukan permutasi awal balikan, IP^{-1}, terhadap blok pra-cipherteks. Gambar 1 memperlihatkan skema algoritma DES yang lebih rinci.

Permutasi awal

Sebelum putaran pertama, terhadap blok plainteks dilakukan permutasi awal (*initial permutation* atau IP). Tujuan permutasi awal adalah mengacak plainteks sehingga urutan bit-bit di dalamnya berubah. Pengacakan dilakukan dengan menggunakan matriks permutasi awal, contohnya seperti pada Tabel 1:

Pembangkitan kunci internal

Karena ada 16 putaran, maka dibutuhknan kunci internal sebanyak 16 buah, yaitu K_1, K_2,…,K_{16}. Kunci kunci internal ini dapat dibangkitkan sebelum proses enkripsi atau bersamaan dengan proses enkripsi. Kunci internal dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Kunci eksternal pada DES panjangnya 64-bit atau 8 karakter.

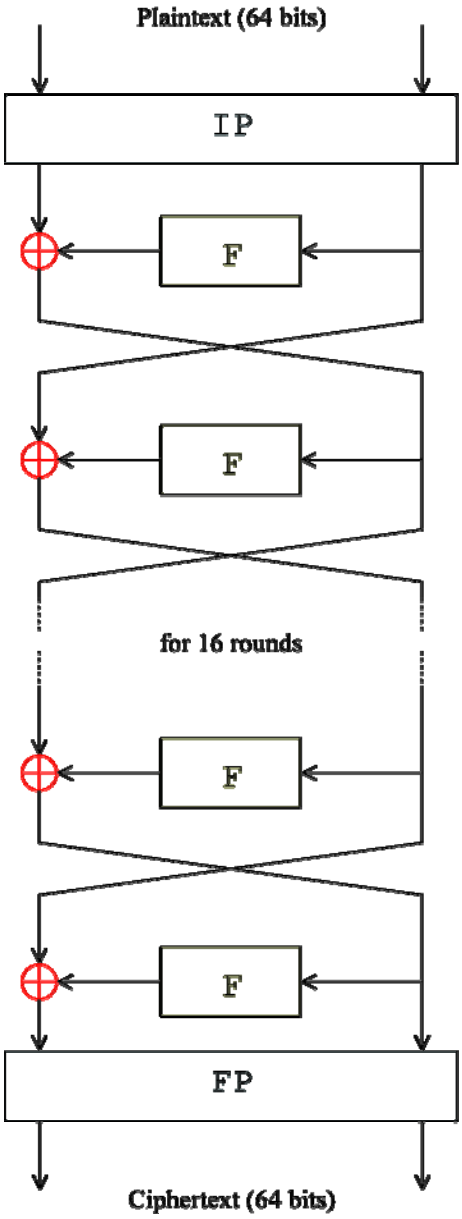
Misalkan kunci eksternal yang tersusun atas 64-bit adalah *K*. Kunci eksternal ini menjadi masukan untuk permutasi dengan menggunakan matriks kompresi PC-1 seperti pada Tabel 2.

Tabel 1 Matriks Permutasi Awal

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Tabel 2 Matriks Permutasi Kompresi

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	30	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4



Gambar 1 Algoritma Enkripsi dengan DES

Dalam permutasi ini, tiap bit kedelapan (*parity bit*) dari delapan *byte* kunci diabaikan. Hasil permutasinya adalah sepanjang 56-bit, sehingga dapat dikatakan panjang kunci DES adalah 56-bit. Selanjutnya, 56-bit ini dibagi menjadi 2 bagian, kiri dan kanan, yang masing-masing panjangnya 28-bit, dan masing-masing disimpan di dalam C_0 dan D_0:

C_0 : berisi bit-bit dari K pada posisi
57, 49, 41, 33,…,26, 18
10, 2, 59, 51,…,44, 36

D_0 : berisi bit-bit dari K pada posisi
63, 55, 47, 39,…,30, 22
14, 6, 61, 53,…,12, 4

Selanjutnya, kedua bagian digeser ke kiri (*left shift*) sepanjang satu atau dua bit bergantung pada tiap putaran. Operasi pergeseran bersifat *wrapping* atau *round-shift*. Jumlah pergeseran pada tiap putaran ditunjukkan pada Tabel 3.

Tabel 3 Jumlah Pergeseran Bit Pada Tiap Putaran

Putaran, *i*	Jumlah pergeseran bit
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Misalkan (C_i, D_i) menyatakan penggabungan C_i dan D_i. (C_{i-1}, D_{i-1}) diperoleh dengan menggeser C_i dan D_i satu atau dua bit. Setelah Pergeseran bit, (C_i, D_i) mengalami permutasi kompresi dengan menggunakan matriks PC-2 seperti pada Tabel 4.

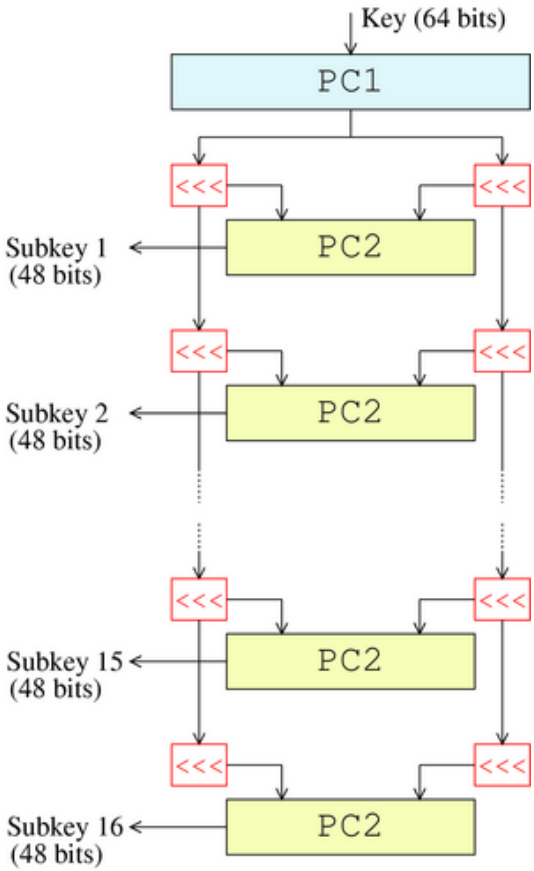
Dengan permutasi ini, kunci internal K_i diturunkan dari (C_i, D_i) yang dalam hal ini K_i merupakan panggabungan bit-bit C_i pada posisi:

14, 17, 11,…,21, 10
23, 19, 12,…,13, 2

Dengan bit-bit D_i pada posisi:

41, 52, 31,…,33, 48
44, 49, 39,…,29, 32

Jadi, setiap klunci internal K_i mempunyai panjang 48-bit. Proses pembangkitan kunci-kunci internal ditunjukkan pada Gambar 2. Bila jumlah pergeseran bit-bit pada Tabel 3 dijumlahkan semuanya, maka jumlah seluruhnya sama dengan 28, yang sama dengan jumlah bit pada C_i dan D_i. Karena itu, setelah putaran ke-16 akan didapatkan kembali $C_{16} = C_0$ dan $D_{16} = D_0$.



Gambar 2 Proses Pembangkitan Kunci-Kunci Internal DES.

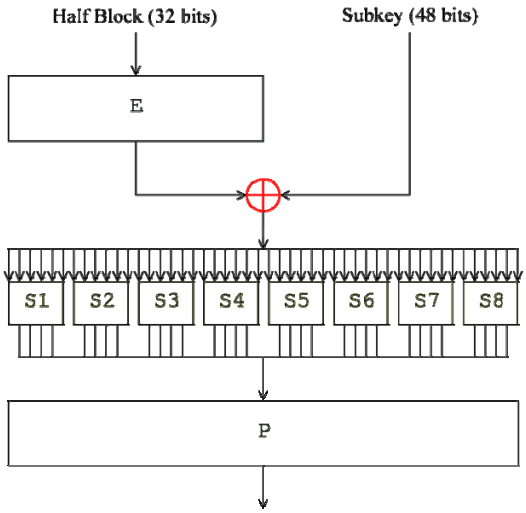
Enciphering

Proses *enciphering* terhadap blok plainteks mengalami 16 kali putaran *enciphering*. Setiap putaran merupakan jaringan Feistel yang secara matematis dinyatakan sebagai

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

Diagram komputasi fungsi f dapat dilihat pada Gambar 3.

E adalah fungsi ekspansi yang memperluas blok R_{i-1} yang panjangnya 32-bit menjadi blok 48-bit. Fungsi ekspansi direalisasikan dengan matriks permutasi ekspansi seperti pada Tabel 4.



Gambar 3 Rincian Komputasi Fungsi *f*

Selanjutnya, hasil ekspansi, yaitu E(R_{i-1}), yang panjangnya 48-bit di-XOR-kan dengan K_i yang panjangya 48-bit menghasilkan vektor A yang panjangnya 48-bit

$$E(R_{i-1}) \oplus K_i = A$$

Vektor *A* dikelompokkan menjadi 8 kelompok, masing-masing 6-bit, dan memjadi masukan bagi proses substitusi. Proses substitusi dilakukan dengan menggunakan delapan buah kotak-*S* (*S-box*), S_1 sampai S_8. Setiap kotak-*S* menerima masukan 6-bit dan menghasilkan keluaran 4-bit. Kelompok 6-bit pertama menggunakan S_1, kelompok 6-bit kedua menggunakan S_2, dan seterusnya.

Keluaran proses substitusi adalah vektor *B* yang panjangnya 48-bit. Vektor *B* menjadi masukan untuk proses permutasi. Tujuan permutasi adalah untuk mengacak hasil proses substitusi kotak-*S*. Permutasi dilakukan dengan menggunakan matriks permutasi *P* (*P-box*) yang digambarkan pada Tabel 5.

Bit-bit *P*(*B*) merupakan keluarandari fungsi *f*. Akhirnya, bit-bit *P*(*B*) di-XOR-kan dengan L_{i-1} untuk mendapatkan R_i.

Tabel 4 Matriks Permutasi Ekspansi

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	30	32	1

Tabel 5 Kotak-S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabel 6 Kotak-S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tabel 7 Kotak-S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tabel 8 Kotak-S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tabel 9 Kotak-S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	3	1	5	0	15	10	3	9	8	16
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tabel 10 Kotak-S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	7	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tabel 11 Kotak-S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tabel 12 Kotak-S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

$$R_i = L_{i-1} \oplus P(B)$$

Keluaran dari putaran ke-*I* adalah

$$(L_i, R_i) = (R_{i-1}, L_{i-1} \oplus P(B))$$

Permutasi terakhir (*Inverse Initial Permutation*)

Permutasi terakhir dilakukan setelah 16 kali putaran terhadap gabungan blok kiri dan blok kanan. Proses permutasi menggunakan matriks permutasi awal balikan (*inverse initial permutation* atau IP^{-1}) seperti pada Tabel 13.

Tabel 13 Matriks Permutasi Terakhir

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Dekripsi

Proses dekripsi terhadap cipherteks merupakan kebalikan dari proses enkripsi. DES menggunakan algoritma yang sama untuk proses enkripsi dan dekripsi. Jika pada proses enkripsi urutan kunci internal yang digunakan adalah K_1, K_2,…, K_{16}, maka pada proses dekripsi urutan

kunci yang digunakan adalah K_{16}, K_{15},…, K_1. Untuk tiap putaran 16, 15,…, 1, keluaran pada setiap putaran *dechipering* adalah

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

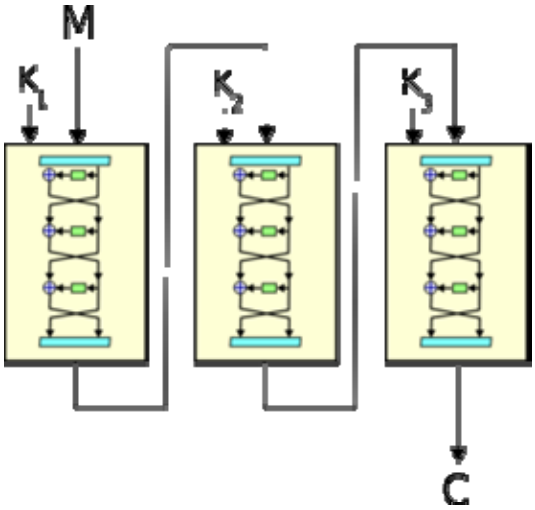
Yang dalam hal ini, (R_{16}, L_{16}) adalah blokj masukan awal untuk *deciphering*. Blok (R_{16}, L_{16}) diperoleh dengan mempermutasikan chiperteks dengan matriks permutasi IP^{-1}. Pra-keluaran dari *deciphering* adalah (L_0, R_0). Dengan permutasi awal IP akan didapatkan kembali blok plainteks semula.

Tinjau Gambar 2. Selama *deciphering*, K_{16} dihasilkan dari (C16, D16) dengan permutasi PC-2. (C16, D16) tidak dapat diperoleh langsung pada permulaan *deciphering*. Tetapi karena (C16, D16) = (C0, D0), maka K16 dapat dihasilkan dari (C0, D0) tanpa perlu lagi melakukan pergeseran bit.

2.2 Triple DES

Triple DES didapat dengan melakukan 3 kali algoritma DES. Urutan langkahnya adalah enkripsi-dekripsi-enkripsi (*EDE*, *Encrypt-Decrypt-Encrypt*). Proses ini dapat dilakukan menggunakan 2 buah kunci atau 3 buah kunci. Urutan kunci TDES menggunakan 2 kunci baik pada enkripsi maupun dekripsi adalah K1-K2-K1. Sementara, pada TDES 3 kunci, urutan kunci pada proses enkripsi adalah K1-K2-K3 dan urutan kunci pada proses dekripsi dibalik menjadi K3-K2-K1. Pada makalah ini yang dibahas adalah TDESm menggunakan 3 kunci.

Proses enkripsi TDES 3 kunci dapat dilihat pada Gambar 4.



Gambar 4 Diagram enkripsi TDES yang menggunakan 3 buah kunci.

Panjang kunci dan blok *cipher*

Jika sebuah algoritma DES menggunakan kunci sepanjang 56-bit, maka TDES dengan 3 kunci yang berbeda menggunakan tiga kali panjang kunci DES, yaitu 168-bit. Sementara ukuran blok *cipher*-nya sama, yaitu 64-bit.

2.2. Twofish

Panjang kunci dan blok *cipher*

Sebagai salah satu kandidat *Advanced Encryption Standard* (AES), Twofish harus memenuhi berbagai kriteria dari *National Institute of Standards and Technology* (NIST). Kriteria-kriteria tersebut antara lain:

1. Algoritma yang ditawarkan termasuk ke dalam kelompok algoritma kriptografi kunci simetri berbasis *cipher* blok.
2. Seluruh rancangan algoritma harus publiki (tidak dirahasiakan).
3. Panjang kunci fleksibel: 128, 192, 256-bit.
4. Ukuran blok yang dienkripsi adalah 128-bit.
5. Algoritma dapat diimplementasikan baik sebagai perangkat lunak maupun perangkat keras.

Dengan demikian jelas bahwa Twofish dapat melakukan *encphering* dan *deciphering* dengan panjang kunci 128, 192, atau 256-bit. Sementara

ukuran blok *cipher* yang diproses oleh Twofish adalah 128-bit.

3.2 Algoritma Twofish

4. Implementasi TDES dan Twofish

4.1 Implementasi TDES

Seperti halnya DES, TDES dapat diterapkan baik sebagai perangkat lunak maupun perangkat keras. Dalam prakteknya, TDES lebih mudah dijumpai dalam bentuk perangkat keras. Salah satu contohnya adalah mesin ATM. Pada mesin ini algoritma TDES diimplementasikan ke dalam bentuk *microprocessor chip*.

Implementasi TDES dalam perangkat lunak dapat dilakukan dalam berbagai bahasa, salah satunya adalah C. Akan tetapi, rumit dan panjangnya implementasi algoritma tersebut menyebabkan *source code*-nya tidak dapat dituliskan dalam makalah ini.

4.2 Implementasi Twofish

Meski tidak terpilih menjadi algoritma AES, Twofish cukup banyak diimplementasikan ke dalam perangkat-perangkat lunak penyandi data. Beberapa perangkat lunak yang mengimplementasikan Twofish antara lain:

1. Away32 Deluxe and Away IDS Deluxe
   Pengenkripsi *file* dan folder untuk Windows.
2. Bassline WinPopUp
   Program *Internet Messenger*
3. Cryptix
   Ekstensi kriptografi untuk Java
4. Crypto++
   Pustaka kelas C++ untuk Twofish.
5. CuteZIP
   Kompresi *file* dan *audio ripping* menggunakan Twofish 128-bit.
6. Foxtrot
   HTTP Server didesain sebagai *application server* profesional. Foxtrot memfasilitasi eksekusi *query* SQL langsung dari *Address Line* browser serta *secure transaction* menggunakan Twofish
7. Password Creator Professional
   Mengimplementasikan Twofish untuk memunculkan *password* secara acak.
8. XPDFViewer
   Menggunakan Twofish untuk mengenkripsi *file*.

Implementasi Twofish dalam bahasa C disediakan oleh pembuatnya, Bruce Scheneier, dalam situsnya, http://www.schneier.com.

5. Perbandingan Kinerja

Bila dibandingkan TDES dan Twofish tidak akan jauh berbeda. Akan tetapi Twofish mengungguli TDES dalam hal kerumitan algoritma. Algoritma Twofish dapat dinilai lebih mangkus dan sangkil dalam mengenkripsi dan mendekripsi data dengan tingkat keamanan tinggi. Selain itu algoritma Twofish dapat bertahan lebih kuat dari serangan *brute force* daripada TDES. Hal ini dapat dikatakan wajar mengingat Twofish merupakan salah satu saingan terkuat Rijndael, algoritma AES yang kini menggantikan TDES.

6. Kesimpulan

1. Algoritma TDES cukup aman dalam mengenkripsi data dan lebih kuat dibandingkan pendahulunya, DES.
2. Algoritma Twofish merupakan salah satu algoritma yang dinilai kuat dan dapat bersaing dengan AES.
3. TDES dan Twofish dapat diimplementasikan ke dalam bentuk perangkat lunak dan perangkat keras.
4. Twofish dapat mengungguli TDES dalam desain algoritmanya yang lebih mangkus dan sangkil.
5. Twofish dapat tahan terhadap serangan daripada algoritma TDES.

Daftar Pustaka

Schneier, Bruce & Doug Whiting, A Performance Comparison of The Five AES Finalist, 2000

Munir, Rinaldi, Kriptografi, 2006

http://www.schneier.com, diakses selama September 2006.

http://en.wikipedia.org, diakses selama September 2006.

http://www.quadibloc.com, diakses selama September 2006.