Bab 7

Kriptografi Block Cipher

Di bab-bab sebelumnya, kita melihat bagaimana enkripsi yang bersifat linear (termasuk *affine transformation*) rentan terhadap analisa statistik. Claude Shannon [sha49], yang dianggap bapak dari teori informasi, dalam *paper* yang diterbitkannya tahun 1949, menganjurkan dua metode untuk mempersulit analisa statistik: *diffusion* dan *confusion*.

Analisa statistik kerap berandalkan pengetahuan mengenai struktur statistik dari naskah asli dan "sisa" dari struktur ini dalam naskah acak. Untuk enkripsi sangat sederhana seperti *Caesar cipher*, "sisa" dari struktur[1] ini masih sangat besar, bahkan masih utuh dan terlokalisir, jadi sangat mudah untuk mencari struktur dalam naskah acak. Efek *diffusion* bertujuan memperlemah "sisa" struktur dengan menyebarnya secara merata ke bagian yang cukup besar dari naskah acak meliputi banyak karakter. Struktur statistik (dalam teori informasi kerap disebut *redundancy*) masih ada tetapi sudah tersebar. Satu-satunya cara menghilangkan struktur tanpa menghilangkan informasi adalah dengan kompresi.

Analisa statistik juga mencari hubungan antara kunci dengan naskah acak. Untuk *simple substitution cipher*, pengetahuan *a priori* mengenai statistik naskah asli memperkecil ruang kunci yang perlu diselidiki secara drastis, karena dapat digunakan untuk menghubungkan kunci dengan struktur statistik dari naskah acak. Efek *confusion* bertujuan untuk menghilangkan atau membuat tidak jelas hubungan antara naskah acak dengan kunci.

Block cipher seperti DES/3DES, CAST, IDEA dan AES menggunakan efek *diffusion* dan *confusion* untuk mempersulit analisa statistik. Ini dilakukan menggunakan apa yang disebut *Feistel network* atau *substitution permutation network*. Walaupun DES sudah dianggap lemah untuk ukuran sekarang, kita akan mulai dengan pembahasan DES karena DES merupakan standard enkripsi

[1] Dalam hal ini struktur statistik didapat dari statistik frekuensi penggunaan huruf.

pertama yang menggunakan konsep *Feistel network* dan algoritma DES masih digunakan oleh 3DES yang dianggap masih cukup kuat untuk penggunaan masa kini.

7.1 DES

DES (Data Encryption Standard) [nis99] pertama dijadikan standard FIPS (Federal Information Processing Standards) oleh NIST (National Institute of Standards and Technology) tahun 1977 untuk digunakan oleh semua instansi pemerintahan Amerika Serikat, dan semua kontraktor dan penyedia jasa untuk pemerintahan Amerika Serikat. DES dirancang oleh tim IBM yang dipimpin Horst Feistel dengan bantuan dari NSA (National Security Agency). DES adalah teknik enkripsi pertama (selain *one-time pad*) yang tahan terhadap *linear cryptanalysis* dan *differential cryptanalysis*.

DES menggunakan kunci sebesar 64 bit untuk mengenkripsi blok juga sebesar 64 bit. Akan tetapi karena 8 bit dari kunci digunakan sebagai *parity*, kunci efektif hanya 56 bit. Gambar 7.1 secara garis besar menunjukkan proses enkripsi DES. Dalam DES, penomoran bit adalah dari kiri kekanan dengan bit 1 menjadi *most significant bit*, jadi untuk 64 bit, bit 1 mempunyai nilai 2^{63}.

Permutasi menggunakan *initial permutation* dilakukan terhadap input sebesar 64 bit. Hasil permutasi dibagi menjadi dua blok L0 dan R0, masing-masing sebesar 32 bit, dimana L0 merupakan 32 bit pertama dari hasil permutasi dan R0 merupakan 32 bit sisanya (bit 33 hasil permutasi menjadi bit 1 R0). Sebanyak 16 putaran enkripsi dilakukan menggunakan fungsi *cipher* f dan setiap putaran menggunakan kunci 48 bit yang berbeda dan dibuat berdasarkan kunci DES. Efeknya adalah setiap blok secara bergantian dienkripsi, masing-masing sebanyak 8 kali.

Pada setiap putaran, blok sebesar 32 bit dienkripsi menggunakan rumus:

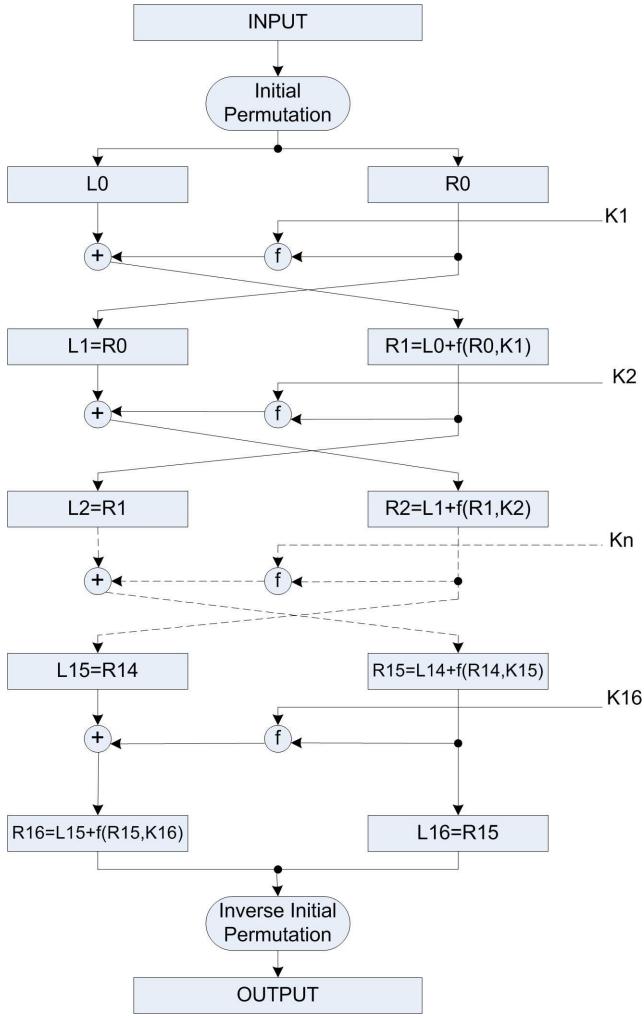
$$R_n = L_{n-1} \oplus f(R_{n-1}, K_n) \tag{7.1}$$

dan blok juga sebesar 32 bit tidak dienkripsi:

$$L_n = R_{n-1} \tag{7.2}$$

dimana
L_{n-1} adalah blok yang sedang giliran tidak dienkripsi,
$\oplus$ adalah operasi *exclusive or* secara bitwise,
f adalah fungsi *cipher* yang akan dijelaskan,
R_{n-1} adalah blok yang sedang giliran dienkripsi, dan
K_n adalah kunci untuk putaran n.

Setelah putaran terahir, kedua blok digabung lagi tetapi bertukaran tempat, jadi R16 menjadi blok pertama dan L16 menjadi blok kedua. Ini dilakukan



Gambar 7.1: Enkripsi DES

untuk menyederhanakan proses dekripsi. Setelah itu permutasi menggunakan *inverse permutation* dilakukan terhadap blok yang sudah digabung menjadi 64 bit memberikan hasil ahir enkripsi DES.

Untuk proses dekripsi, rumus 7.1 dan 7.2 memberikan:

$$\begin{aligned} L_{n-1} &= R_n \oplus f(R_{n-1}, K_n) \\ &= R_n \oplus f(L_n, K_n). \end{aligned}$$

Jadi tanpa harus mengetahui fungsi f, kita tahu bahwa operasi *bitwise exclusive or* R_n dengan $f(R_{n-1}, K_n) = f(L_n, K_n)$ akan mendapatkan kembali L_{n-1} (lihat 2.2 - penjelasan enkripsi *one-time pad*).

Juga karena

$$\begin{aligned} \text{output} &= \text{IP}^{-1}(\text{R16L16}) \text{ dan} \\ \text{L0R0} &= \text{IP}(\text{input}) \end{aligned}$$

dimana IP adalah *initial permutation*, maka

$$\begin{aligned} \text{R16L16} &= \text{IP}(\text{output}) \text{ dan} \\ \text{input} &= \text{IP}^{-1}(\text{L0R0}). \end{aligned}$$

Jadi proses dekripsi DES dapat menggunakan algoritma yang sama dengan enkripsi, asalkan *schedule* kunci dibalik (mulai dari K16 dan berahir dengan K1). Blok yang terahir dienkripsi harus didekripsi pertama, itulah sebabnya proses enkripsi membuat L16 dan R16 bertukaran tempat.

IP

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 7.1: Tabel *Initial Permutation*

Tabel 7.1 adalah tabel untuk *Initial Permutation*. Bit 1 hasil permutasi berasal dari bit 58 input, bit 2 dari bit 50, dan seterusnya hingga bit 64 dari bit 7.

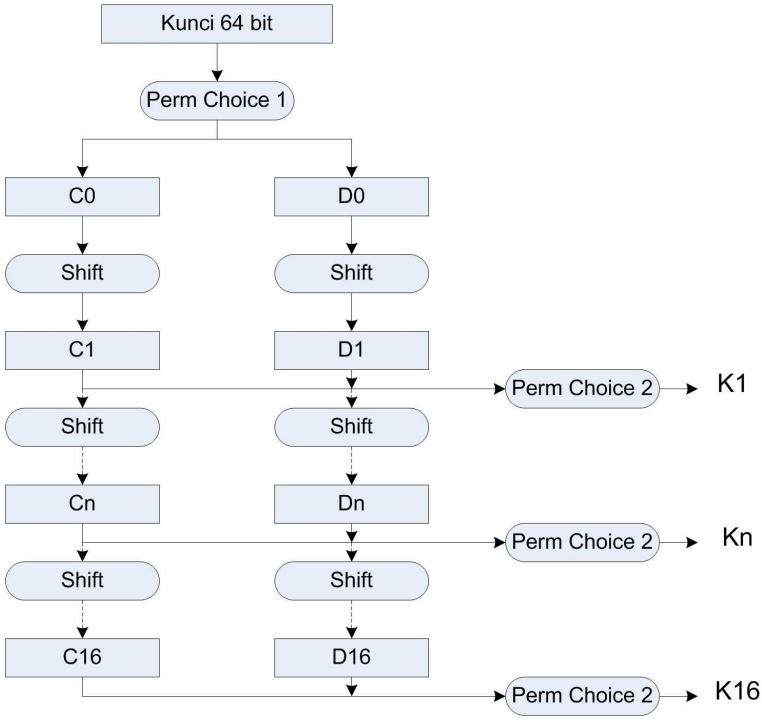
Tabel 7.2 adalah tabel untuk *Inverse Initial Permutation*. Transformasi *Inverse Initial Permutation* adalah *inverse* dari transformasi *Initial Permutation*:

- dalam IP bit 1 berasal dari bit 58, dalam IP^{-1} bit 58 berasal dari bit 1,
- dalam IP bit 2 berasal dari bit 50, dalam IP^{-1} bit 50 berasal dari bit 2,
- dalam IP bit 3 berasal dari bit 42, dalam IP^{-1} bit 42 berasal dari bit 3, dan seterusnya.

IP^{-1}

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

Tabel 7.2: Tabel *Inverse Initial Permutation*



Gambar 7.2: Algoritma *Key Schedule* DES

Jadi jika bit 58 dipindahkan oleh IP menjadi bit 1, IP^{-1} akan mengembalikannya ke posisi bit 58, demikian juga bit lainnya.

Gambar 7.2 menunjukkan secara garis besar algoritma *key schedule* pembuatan 16 kunci putaran. Transformasi *permuted choice* 1 membuang 8 bit untuk *parity* dan melakukan permutasi terhadap 56 bit yang tersisa, yang kemudian dibagi menjadi dua blok C0 dan D0, masing-masing 28 bit.

PC1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	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

Tabel 7.3: Tabel *Permuted Choice* 1

Untuk setiap putaran n:

- operasi *shift* dilakukan terhadap blok C_{n-1} dan blok D_{n-1} menghasilkan masing-masing blok C_n dan blok D_n, dan

- operasi *permuted choice* 2 dilakukan terhadap gabungan blok C_n dan blok D_n, membuang 8 dari 56 bit, lalu melakukan permutasi terhadap 48 bit yang tersisa, menghasilkan kunci putaran K_n.

Tabel 7.3 menunjukkan operasi *permuted choice* 1. Tabel mempunyai dua bagian, bagian pertama untuk membuat blok C0 dan bagian kedua untuk membuat blok D0. Jadi bit 1 blok C0 didapat dari bit 57 kunci DES, bit 2 blok C0 didapat dari bit 49 kunci DES, dan seterusnya. Bit 1 blok D0 didapat dari bit 63 kunci DES, bit 2 blok D0 didapat dari bit 55 kunci DES, dan seterusnya. Bit 8, 16, 24, 32, 40, 48, 56 dan 64 kunci DES tidak digunakan.

Operasi *shift* merotasi blok 28 bit satu atau dua posisi kekiri tergantung pada putaran. Merotasi satu posisi kekiri berarti bit 1 menjadi bit 28 (karena bit 1 adalah bit paling kiri), bit 2 menjadi bit 1, bit 3 menjadi bit 2, dan seterusnya. Merotasi dua posisi berarti bit 1 menjadi bit 27, bit 2 menjadi bit 28, bit 3 menjadi bit 1, dan seterusnya.

Tabel 7.4 menunjukkan berapa besar *shift* yang harus dilakukan untuk setiap putaran. Jadi untuk putaran 1, 2, 9 dan 16 blok C dan blok D masing-masing dirotasi 1 posisi, sedangkan untuk putaran 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14 dan 15 besar rotasi adalah 2 posisi.

Putaran	Besar Shift
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

Tabel 7.4: Tabel *Shift*

Tabel 7.5 menunjukkan operasi *permuted choice* 2. Bit 9, 18, 22, 25, 35, 38, 43 dan 54 dibuang, bit 1 kunci putaran didapat dari bit 14 gabungan blok C dan D, bit 2 kunci putaran didapat dari bit 17 gabungan blok C dan D, dan seterusnya.

Komponen terahir dari algoritma DES yang perlu dijelaskan adalah fungsi *cipher f*. Gambar 7.3 menunjukkan fungsi *cipher f* secara garis besar.

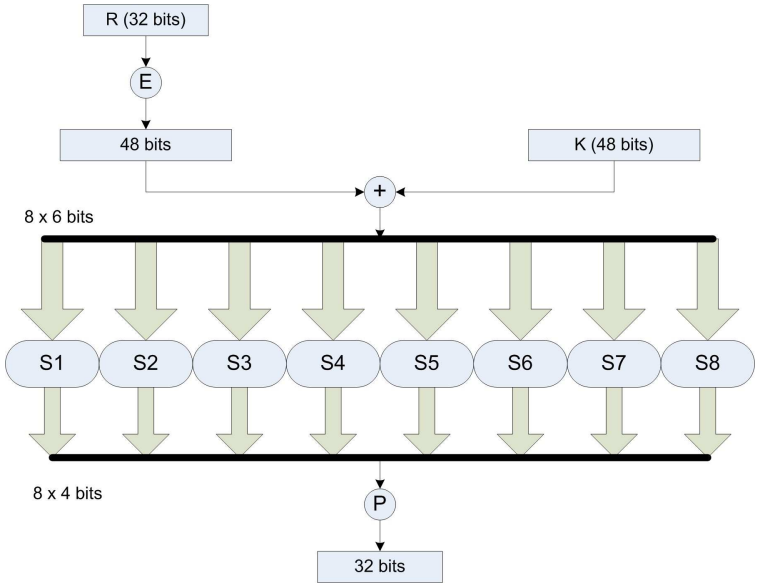
Blok sebesar 32 bit diekspansi menjadi 48 bit menggunakan transformasi E (lihat tabel B.1 di appendix B untuk tabel transformasi E). Operasi *exclusive or* dilakukan terhadap hasil ekspansi dan kunci putaran (yang juga 48 bit). Hasil *exclusive or* lalu dibagi menjadi 8 bagian, masing-masing sebesar 6 bit. Setiap bagian disubstitusi menghasilkan 4 bit, menggunakan fungsi substitusi S1 sampai dengan S8 (lihat tabel B.5 di appendix B). Karena operasi ini merupakan bagian dari fungsi *cipher f*, ini bukan operasi langsung terhadap naskah (lihat gambar 7.1, naskah direpresentasikan oleh L), jadi tidak ada informasi yang hilang dari naskah asli dengan dilakukannya substitusi 6 bit menjadi 4 bit. Gabungan hasil substitusi sebesar 32 bit kemudian dipermutasi menggunakan P yang menghasilkan 32 bit hasil ahir (lihat tabel B.2 di appendix B untuk tabel permutasi P).

Enkripsi DES menggunakan permutasi dan substitusi secara berulang untuk mendapatkan efek *diffusion* dan *confusion*. Operasi permutasi lebih mengarah

PC2

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tabel 7.5: Tabel *Permuted Choice* 2



Gambar 7.3: Fungsi *Cipher f*

pada efek *diffusion*, karena permutasi berefek menyebar struktur informasi naskah keseluruh blok sebesar 64 bit. Operasi substitusi dengan *S-box* membuat hubungan antara kunci dengan naskah enkripsi tidak jelas, jadi menghasilkan efek *confusion*.

Meskipun semula ada kecurigaan terhadap ketangguhan enkripsi DES (ada yang berspekulasi bahwa NSA, yang ikut berperan dalam merancang DES, tidak akan memperbolehkan enkripsi tangguh digunakan oleh pihak lain), ter-

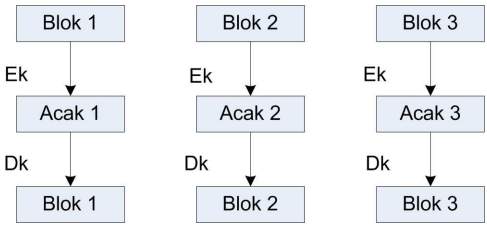
nyata DES cukup tahan terhadap analisa statistik. Masih ada kecurigaan terhadap fungsi *S-box* karena tidak adanya penjelasan mengenai bagaimana *S-box* dirancang. Akan tetapi, kelemahan DES adalah besar kunci efektif hanya 56 bit, yang membuatnya rentan terhadap *brute force attack*. Pada bulan Mei 2005, DES sebagai standard FIPS telah dicabut.

7.2 Mode Operasi DES

Standard FIPS-81, yang pernah dikeluarkan oleh NIST, memberi petunjuk untuk mode operasi DES. Standard tersebut telah dicabut seiring dengan dicabutnya DES sebagai standard FIPS, namun mode operasi dalam standard dapat digunakan untuk *block cipher* lainnya. Mode operasi dalam FIPS-81 adalah:

- Electronic Code Book (ECB).
- Cipher Block Chaining (CBC).
- Cipher Feedback (CFB).
- Output Feedback (OFB).

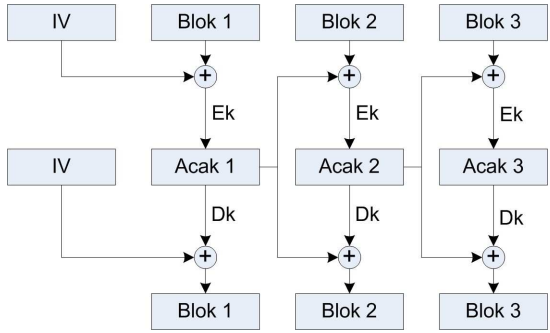
Dengan mode operasi ECB, data dienkripsi secara berurutan per blok menggunakan satu kunci DES. Jika naskah asli berisi blok yang berulang, maka naskah acak juga berisi blok yang berulang di posisi yang sama. Gambar 7.4 menunjukkan penggunaan DES dengan mode operasi ECB.



Gambar 7.4: DES dengan mode ECB

Dengan mode operasi CBC, setiap blok data dikombinasikan dahulu dengan hasil enkripsi blok sebelumnya menggunakan operasi *exclusive or*, lalu dienkripsi menggunakan kunci DES. Untuk blok pertama, blok dikombinasikan dengan *initialization vector* (IV) sebesar 64 bit, yang sebaiknya dibuat secara acak. Gambar 7.5 menunjukkan penggunaan DES dengan mode CBC.

Dengan mode operasi CBC, naskah yang berisi blok yang berulang akan dienkripsi menjadi sesuatu yang tidak berulang[2]. Akibatnya mode CBC mempersulit analisa untuk memecahkan enkripsi. Mode CBC adalah mode terpopuler untuk penggunaan DES dan *block cipher* lainnya karena menghasilkan enkripsi yang teraman diantara semua mode.



Gambar 7.5: DES dengan mode CBC

Mode CFB menghasilkan apa yang dinamakan *stream cipher* (lihat bab 6) dimana operasi *exclusive or* dilakukan terhadap naskah asli dengan *keystream* menghasilkan naskah acak, jadi meniru enkripsi *one-time pad*. Namun naskah acak dijadikan *feedback* untuk ikut menentukan *keystream*, jadi agak lebih aman dibandingkan OFB. Gambar 7.6 menunjukkan penggunaan DES dengan mode CFB.

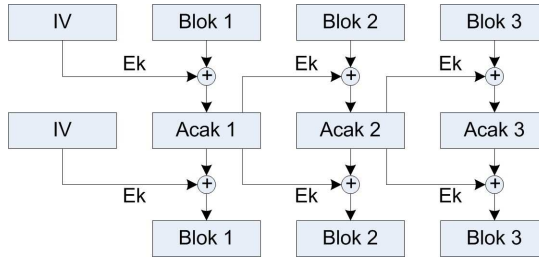
Dengan mode operasi CFB, *keystream* dibuat dengan mengenkripsi IV dan naskah acak menggunakan algoritma enkripsi DES. Jadi untuk blok pertama, bagian *keystream* yang digunakan adalah hasil enkripsi IV, untuk blok kedua, bagian *keystream* yang digunakan adalah hasil enkripsi blok acak pertama, dan seterusnya (naskah acak dijadikan *feedback*). Jadi enkripsi DES dapat dianggap sebagai *pseudo-random number generator* dengan IV atau blok acak sebelumnya sebagai *seed*.

Selain menggunakan blok sebesar 64 bit, mode CFB dapat juga digunakan dengan besar blok kurang dari 64 bit (sebut saja k bit dengan $1 \leq k < 64$) sebagai berikut:

- IV sebesar k bit digunakan sebagai *least significant bits* untuk *seed* blok pertama, dengan *most significant bits* diisi 0.
- *Seed* lalu dienkripsi menggunakan DES, dengan k *most significant bits* hasil enkripsi digunakan sebagai bagian *keystream* untuk blok pertama.

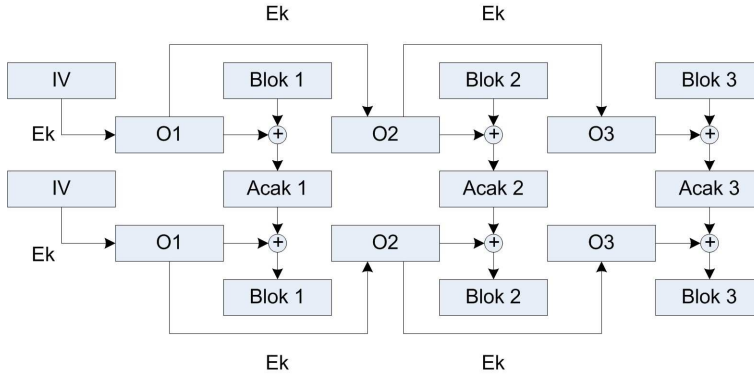
[2]Ini mungkin tidak pasti, namun jika ada kemungkinan blok berulang juga dalam naskah acak, kemungkinan ini sangat kecil.

- Blok acak didapat dari operasi *exclusive or* antara blok asli dengan bagian *keystream* untuk blok.
- Untuk blok-blok berikutnya, *seed* didapat dengan membuang k *most significant bits* dari *seed* blok sebelumnya, menggeser sisa *seed* kekiri k posisi menjadi *most significant bits*, dan mengisi k *least significant bits* dengan blok acak sebelumnya.



Gambar 7.6: DES dengan mode CFB

Mode OFB juga menghasilkan *stream cipher*, tetapi pembuatan *keystream* berbeda dengan mode CFB. *Keystream* hanya ditentukan oleh IV dan kunci enkripsi, jadi OFB lebih lemah dibandingkan CFB. Gambar 7.7 menunjukkan penggunaan DES dengan mode OFB.



Gambar 7.7: DES dengan mode OFB

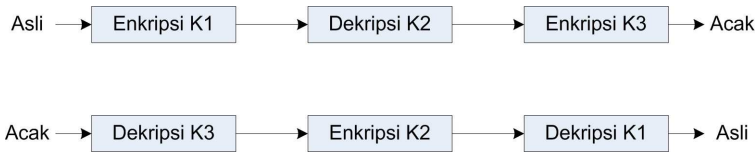
Beda antara OFB dengan CFB terletak pada *feedback*. Dengan OFB, yang dijadikan *feedback* adalah *keystream*, bukan naskah acak. Sama dengan CFB,

selain menggunakan blok sebesar 64 bit, OFB dapat juga digunakan dengan besar blok kurang dari 64 bit (sebut saja k bit dengan $1 \leq k < 64$) sebagai berikut:

- IV sebesar k bit digunakan sebagai *least significant bits* untuk *seed* blok pertama, dengan *most significant bits* diisi 0.
- *Seed* lalu dienkripsi menggunakan DES, dengan k *most significant bits* hasil enkripsi digunakan sebagai bagian *keystream* untuk blok pertama.
- Blok acak didapat dari operasi *exclusive or* antara blok asli dengan bagian *keystream* untuk blok.
- Untuk blok-blok berikutnya, *seed* didapat dengan membuang k *most significant bits* dari *seed* blok sebelumnya, menggeser sisa *seed* kekiri k posisi menjadi *most significant bits*, dan mengisi k *least significant bits* dengan bagian *keystream* blok sebelumnya.

7.3 3DES

Enkripsi dengan DES sudah dianggap tidak memadai karena kunci efektif sebesar 56 bit sudah terlalu kecil. Standard Triple DES menggunakan algoritma DES dengan 3 kunci seperti dalam gambar 7.8.



Gambar 7.8: Enkripsi dan Dekripsi 3DES

Dengan 3 kunci DES K1, K2 dan K3, enkripsi 3DES dilakukan sebagai berikut:

1. Enkripsi DES dengan kunci K1 dilakukan terhadap naskah asli.
2. Dekripsi DES dengan kunci K2 dilakukan terhadap hasil pertama.
3. Enkripsi DES dengan kunci K3 dilakukan terhadap hasil kedua.

Jadi enkripsi 3DES mempunyai rumus:

$$C = E3_{k1,k2,k3}(P) = E_{k3}(D_{k2}(E_{k1}(P))) \tag{7.3}$$

dimana
$E3_{k1,k2,k3}(P)$ adalah enkripsi 3DES terhadap P dengan kunci $k1$, $k2$ dan $k3$,
$E_{kn}(P)$ adalah enkripsi DES terhadap P dengan kunci kn, dan
$D_{kn}(P)$ adalah dekripsi DES terhadap P dengan kunci kn.

Dekripsi 3DES mempunyai rumus:

$$P = D3_{k1,k2,k3}(C) = D_{k1}(E_{k2}(D_{k3}(C))) \tag{7.4}$$

dimana
$D3_{k1,k2,k3}(C)$ adalah dekripsi 3DES terhadap C dengan kunci $k1$, $k2$ dan $k3$.

Triple DES dapat digunakan dengan satu kunci k sebagai berikut:

$$\begin{aligned} E3_{k,k,k}(P) &= E_k(D_k(E_k(P))) \\ &= E_k(P) \text{ dan} \\ D3_{k,k,k}(C) &= D_k(E_k(D_k(C))) \\ &= D_k(C). \end{aligned}$$

Jadi 3DES dengan 1 kunci ekuivalen dengan DES.

Dengan dua kunci $k1$ dan $k2$, penggunaan 3DES adalah sebagai berikut:

$$\begin{aligned} E3_{k1,k2,k1}(P) &= E_{k1}(D_{k2}(E_{k1}(P))) \text{ dan} \\ D3_{k1,k2,k1}(C) &= D_{k1}(E_{k2}(D_{k1}(C))). \end{aligned}$$

Enkripsi 3DES dengan dua atau tiga kunci masih cukup tangguh untuk penggunaan saat ini. Walaupun enkripsi 3DES agak lamban komputasinya dibandingkan dengan enkripsi *block cipher* lainnya yang masih cukup tangguh, 3DES tergolong populer.

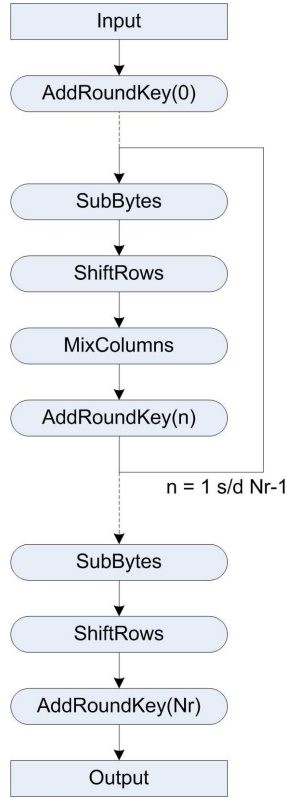
Bagi pembaca yang ingin mendapatkan rekomendasi penggunaan 3DES secara rinci, dipersilahkan untuk membaca [nis08].

7.4 AES

AES (Advanced Encryption Standard) [nis01] adalah teknik enkripsi yang dijadikan standard FIPS oleh NIST tahun 2001. AES dimaksudkan akan, secara bertahap, menggantikan DES sebagai standard enkripsi di Amerika Serikat untuk abad ke 21. (DES sebagai standard FIPS telah dicabut, Mei 2005.)

AES menjadi standard melalui proses seleksi. Dari beberapa teknik enkripsi yang dicalonkan untuk menjadi AES, yang terpilih adalah enkripsi Rijndael. Teknik enkripsi ini termasuk jenis *block cipher* seperti halnya dengan DES. Perbedaan utama antara teknik enkripsi AES dan teknik enkripsi DES adalah AES juga menggunakan substitusi (menggunakan S-boxes) secara langsung terhadap naskah, sedangkan substitusi S-box digunakan DES hanya dalam fungsi

cipher f yang hasilnya kemudian dioperasikan terhadap naskah menggunakan *exclusive or*, jadi DES tidak menggunakan substitusi secara langsung terhadap naskah. AES juga menggunakan kunci enkripsi yang lebih besar yaitu 128 bit, 192 bit, atau 256 bit.



Gambar 7.9: Enkripsi AES

Gambar 7.9 menunjukkan, secara garis besar, enkripsi AES. Input berupa naskah asli sebesar 128 bit, sedangkan output adalah naskah acak sebesar 128 bit. Setiap transformasi dilakukan secara langsung terhadap naskah, mulai dengan transformasi $AddRoundKey(0)$. Jadi setiap transformasi harus mempunyai *inverse* agar naskah acak dapat didekripsi. Pada putaran 1 sampai dengan $Nr - 1$, transformasi *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* dilakukan terhadap naskah. Pada putaran terahir (Nr), transformasi *SubBytes*, *ShiftRows* dan *AddRoundKey* dilakukan terhadap naskah (transformasi *MixColumns* tidak dilakukan). Total ada Nr putaran.

Jumlah putaran (Nr) tergantung pada besar kunci yang digunakan. Tabel 7.6 menunjukkan jumlah putaran (Nr) untuk kunci sebesar 128 bit, 192 bit dan 256 bit. Jadi untuk kunci sebesar 128 bit, besar kunci (Nk) adalah 4 *word* (setiap *word* mempunyai 32 bit), besar blok (Nb) adalah 4 *word*, dan jumlah putaran (Nr) adalah 10.

	Besar Kunci (Nk) dalam *words*	Besar Blok (Nb) dalam *words*	Jumlah Putaran (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabel 7.6: Tabel untuk Jumlah Putaran

Penjelasan AES menggunakan konvensi urutan indeks untuk bit dan byte sebagai berikut:

- Urutan indeks bit sebagai input adalah $input_0$, $input_1$, $input_2$, dan seterusnya.
- Urutan indeks byte sebagai input adalah a_0, a_1, a_2, dan seterusnya.
- Dalam byte, bit menggunakan urutan indeks berlawanan dengan input: b_7, b_6, b_5, dan seterusnya sampai dengan b_0. *Most significant bit* dalam byte adalah b_7.

input	0	1	2	3	4	5	6	7	8	9	...	...
a	0								1			...
b	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	...

Tabel 7.7: Index bit dan byte AES

Tabel 7.7 memperlihatkan konvensi indeks bit dan byte untuk AES. Tabel menunjukkan bahwa untuk byte kedua (a_1), $b_7 = input_8$, $b_6 = input_9$, $b_5 = input_{10}$, dan seterusnya sampai dengan $b_0 = input_{15}$.

Algoritma enkripsi AES beroperasi terhadap *state* dari naskah yang dipandang sebagai matrik terdiri dari 16 byte. Setiap kolom dari *state* merepresentasikan satu *word*, dengan $s_{0,i}$ sebagai *most significant byte* untuk kolom i. Tabel 7.8 menunjukkan bagaimana *state* didapat dari input dan bagaimana *state* dijadikan output. Transformasi *AddRoundKey*, *SubBytes*, *ShiftRows* dan *MixColumns* semua dilakukan terhadap *state*.

input bytes

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_10	in_{14}
in_3	in_7	in_11	in_{15}

⇓

state

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

⇓

output bytes

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_10	out_{14}
out_3	out_7	out_11	out_{15}

Tabel 7.8: Input, State dan Output

Beberapa transformasi yang dilakukan dalam enkripsi AES memperlakukan byte seolah *polynomial* dalam *polynomial field* $\mathbf{GF}(2^8)$ (lihat bagian 5.7). Setiap bit dalam byte merepresentasikan koefisien suku *polynomial* sebagai berikut:

- bit b_7 merupakan koefisien untuk x^7,
- bit b_6 merupakan koefisien untuk x^6,
- dan seterusnya sampai dengan bit b_0 yang merupakan koefisien untuk x^0 (konstan).

Operasi *inverse polynomial* terhadap byte dapat dilakukan (dalam $\mathbf{GF}(2^8)$) jika byte $\neq 0$. *Irreducible polynomial* yang digunakan AES untuk $\mathbf{GF}(2^8)$ adalah

$$x^8 + x^4 + x^3 + x + 1$$

AES juga melakukan transformasi *affine* (dalam $\mathbf{GF}(2)$) terhadap byte menggunakan rumus

$$b'_i = b_i + b_{i+4 \bmod 8} + b_{i+5 \bmod 8} + b_{i+6 \bmod 8} + b_{i+7 \bmod 8} + c_i \qquad (7.5)$$

dengan $0 \leq i \leq 7$ dan byte $c = 01100011$. Jika dijabarkan, rumus menjadi

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Operasi *SubByte* (substitusi S-box) terhadap byte terdiri dari operasi *inverse polynomial* (jika byte $\neq 0$) diikuti oleh transformasi *affine* menggunakan rumus 7.5. Jika byte $= 0$ maka hanya transformasi *affine* yang dilakukan, menghasilkan 01100011 (63 dalam notasi hexadecimal). Operasi *SubByte* (substitusi S-box) dan *inverse* operasi *SubByte* dalam bentuk tabel kami berikan di appendix C.

Operasi *AddRoundKey* melakukan *bitwise exclusive or* menggunakan kunci putaran sebesar 128 bit terhadap *state* (yang juga 128 bit). Untuk setiap putaran, kunci putaran berbeda dari *key schedule* yang didapat dari ekspansi kunci enkripsi digunakan. Total ada $Nr+1$ kunci putaran yang digunakan, satu untuk operasi awal sebelum putaran pertama, dan satu untuk setiap putaran. Jadi ekspansi kunci menghasilkan $Nb(Nr+1) = 4(Nr+1)$ *words* menggunakan algoritma sebagai berikut:

1. $i \leftarrow 0$,
2. $w[i] \leftarrow [k[4i], k[4i+1], k[4i+2], k[4i+3]]$,
3. $i \leftarrow i+1$, jika $i < Nk$ kembali ke langkah 2,
4. $t \leftarrow w[i-1]$,
5. jika $i \bmod Nk = 0$ maka $t \leftarrow SubWord(RotWord(t)) \oplus Rcon[i/Nk]$, kalau tidak, jika $Nk > 6$ dan $i \bmod Nk = 4$ maka $t \leftarrow SubWord(t)$,
6. $w[i] \leftarrow w[i-Nk] \oplus t$,
7. $i \leftarrow i+1$, jika $i < Nb(Nr+1)$ kembali ke langkah 4

dimana

- k adalah kunci enkripsi,
- w adalah *array* terdiri dari $Nb(Nr+1)$ *words* dengan indeks mulai dari 0,

- $\oplus$ adalah operasi *bitwise exclusive or*,
- *SubWord* adalah operasi terhadap *word* dimana substitusi *SubByte* dilakukan terhadap setiap byte dalam *word*,
- *RotWord* adalah operasi terhadap *word* dimana urutan byte dalam *word* diubah sebagai berikut:

$$[a_0, a_1, a_2, a_3] \Rightarrow [a_1, a_2, a_3, a_0],$$

dan

- *Rcon*[*j*] adalah word sebagai berikut:

$$[x^{j-1}, 0, 0, 0]$$

dimana $x^0 = 00000001$, $x = 00000010$, $x^2 = 00000100$ dan seterusnya sampai dengan $x^7 = 10000000$, dan untuk x dengan pangkat > 7 aritmatika *polynomial field* digunakan.

Jadi Nk *words* pertama untuk w didapatkan langsung dari kunci enkripsi, sedangkan *words* selanjutnya untuk w ditentukan oleh *word* 1 posisi sebelumnya dan *word* Nk posisi sebelumnya. Kunci putaran diambil dari w sebagai berikut:

$$k_i \leftarrow [w[4i], w[4i+1], w[4i+2], w[4i+3]]$$

dengan $0 \leq i \leq Nr$. Transformasi *AddRoundKey*(i) terhadap *state* adalah sebagai berikut:

$$state' \leftarrow state \oplus k_i.$$

Transformasi *AddRoundKey* mempunyai *inverse* yaitu transformasi *AddRoundKey* sendiri (ingat sifat operasi *exclusive or*).

Untuk transformasi *SubBytes*, substitusi *SubByte* dilakukan terhadap setiap byte dalam *state*. Untuk *inverse* transformasi *SubBytes*, substitusi dilakukan menggunakan *inverse SubByte* (lihat appendix C).

Transformasi *ShiftRows* menggeser byte dalam baris *state* sebagai berikut:

- baris 0 tidak digeser,
- baris 1 digeser 1 posisi kekiri dengan byte terkiri menjadi byte terkanan,
- baris 2 digeser 2 posisi kekiri, jadi dua byte sebelah kiri ditukar dengan dua byte sebelah kanan, dan
- baris 3 digeser 3 posisi kekiri, yang efeknya sama dengan menggeser 1 posisi kekanan.

state

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$\Downarrow$

state'

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,1}$	$s_{1,2}$	$s_{1,3}$	$s_{1,0}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

Tabel 7.9: ShiftRows

Tabel 7.9 memperlihatkan efek *ShiftRows* terhadap *state*. Tidak terlalu sulit untuk menunjukkan bahwa transformasi *ShiftRows* mempunyai *inverse*.

Transformasi terahir yang perlu dijelaskan untuk enkripsi AES adalah transformasi *MixColumns*. Untuk *MixColumns*, kolom dalam *state* (yang merupakan *word*) diperlakukan sebagai *polynomial* dengan koefisien dalam $\mathbf{GF}(2^8)$. *Word*

$$[a_0, a_1, a_2, a_3]$$

diperlakukan sebagai *polynomial*

$$a_3x^3 + a_2x^2 + a_1x + a_0.$$

Pertambahan *polynomial* dilakukan sebagaimana pertambahan dalam *polynomial ring*. Agar hasil tetap sebesar *word*, perkalian dilakukan modulo *polynomial*

$$g(x) = x^4 + 1.$$

Tidak terlalu sulit untuk menunjukkan bahwa

$$x^i \bmod (x^4 + 1) = x^{i \bmod 4} \tag{7.6}$$

karena $-1 = 1$ dalam $\mathbf{GF}(2^8)$. Karena $x^4 + 1$ bukan merupakan *irreducible polynomial* dalam $K[x]$ dimana $K = \mathbf{GF}(2^8)$, maka $K[x]/g(x)K[x]$ bukan merupakan *field*: tidak semua *polynomial* mempunyai *inverse*. Akan tetapi *polynomial*

$$a(x) = 03x^3 + 01x^2 + 01x + 02$$

mempunyai *inverse*

$$a^{-1}(x) = 0\mathrm{b}x^3 + 0\mathrm{d}x^2 + 09x + 0\mathrm{e}.$$

Transformasi *MixColumns* mengalikan (modulo *polynomial* $g(x)$) setiap kolom dalam *state* (diperlakukan sebagai *polynomial*) dengan *polynomial* $a(x)$. Transformasi *MixColumns* terhadap *state* dapat dirumuskan efeknya terhadap setiap kolom c sebagai berikut:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}.$$

Tidak terlalu sulit untuk meyakinkan kebenaran rumus ini menggunakan rumus 7.6. Transformasi *MixColumns* mempunyai *inverse* yaitu mengalikan setiap kolom dalam *state* dengan $a^{-1}(x)$ modulo *polynomial* $g(x)$.

Karena semua transformasi yang dilakukan dalam enkripsi AES mempunyai *inverse*, dekripsi dapat dilakukan menggunakan *inverse* transformasi dengan urutan kebalikan dari urutan transformasi enkripsi. Proses dekripsi dimulai dengan transformasi *AddRoundKey*(Nr) terhadap naskah acak, diikuti dengan transformasi *inverse ShiftRows*, *inverse SubBytes* dan seterusnya.

AES telah menggantikan DES sebagai standard enkripsi untuk keperluan instansi pemerintahan Amerika Serikat. Ada yang meragukan ketangguhan AES karena semua transformasi yang dilakukan dalam enkripsi AES mempunyai rumus aljabar yang elegan. Akan tetapi rumus yang elegan tidak berarti parameter kunci dapat dipecahkan dengan mudah dari persamaan. Kriptografi *public key* menggunakan rumus aljabar yang elegan, namun parameter kunci privat sulit untuk dipecahkan. Kembali ke AES, karena enkripsi adalah manipulasi berbagai bit, pemecahan parameter kunci dapat dirumuskan sebagai masalah SAT dalam aljabar Boolean, namun masalah SAT adalah masalah yang bersifat NP-*complete* yang pada umumnya tidak dapat dipecahkan secara efisien (untuk pemecahan kunci enkripsi AES, ruang pencarian terlalu besar sehingga dalam prakteknya pemecahan tidak dapat dilakukan).

7.5 Ringkasan

Dalam bab ini kita telah bahas berbagai enkripsi *block cipher* antara lain DES dan AES. Meskipun dianggap sudah tidak memadai untuk ukuran sekarang, DES dibahas karena merupakan *block cipher* pertama yang menjadi standard internasional dan merupakan contoh dari enkripsi yang tahan terhadap *differential cryptanalysis* dan *linear cryptanalysis*. Mode penggunaan DES juga dibahas, dengan mode CBC (*cipher block chaining*) sebagai mode yang direkomendasikan, sedangkan CFB dan OFB adalah mode-mode penggunaan DES sebagai *building block* dari suatu *stream cipher*. DES juga digunakan sebagai *building block* untuk 3DES (*triple*-DES). AES dibahas karena merupakan standard baru yang telah menggantikan DES. Berbeda dengan *block cipher*

lainnya, AES menggunakan transformasi *affine* (tetapi dalam suatu *polynomial field*) untuk fungsi S-box.