

# **Analisis Sandi Linear Terhadap Ciphertext dengan SPN**

Diajukan sebagai tugas akhir mata kuliah  
**Keamanan Jaringan Lanjut – EC 7010**  
Dosen : **Dr. Ir. Budi Raharjo**

Disusun oleh:  
**DIDIN SAEFUDIN**  
**23203104**



**PROGRAM MAGISTER TEKNIK ELEKTRO**  
**BIDANG KHUSUS TEKNOLOGI INFORMASI-DIKMENJUR**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2004**

# Analisis Sandi Linear Terhadap Ciphertext dengan SPN

## 1. Pendahuluan

Analisis Sandi Linear diperkenalkan oleh Matsui pada Eurocrypt tahun 1993 sebagai *attack* teoritis pada *Data Encryption Standard* (DES) dan kemudian cukup berhasil secara aplikatif pada DES, DES adalah suatu algoritma enkripsi *block cipher* yang didefinisikan dan didukung oleh pemerintah Amerika Serikat, dan pada tahun 1977 dijadikan standar resmi, DES menggunakan kunci *56-bit* untuk me-enkripsi blok data *64-bit*.

Analisis sandi linear ini merupakan *attack* yang paling penting, yang digunakan pada kode blok kunci simetri modern yang berbasis komputer. Memahami *attack* jenis ini sangat penting karena kode Rijndael yang dijadikan standar enkripsi terbaru Amerika Serikat, *Advanced Encryption Standard* (AES) yang merupakan pelanjut DES yang menjadi standar de-facto enkripsi di dunia, menggunakan arsitektur *Substitution-Permutation Network* (SPN) ini. Oleh karena itu analisis sandi linear, pengkajiannya didasarkan pada analisis kode jaringan Substitusi-Permutasi dasar yang sederhana dan terstruktur.

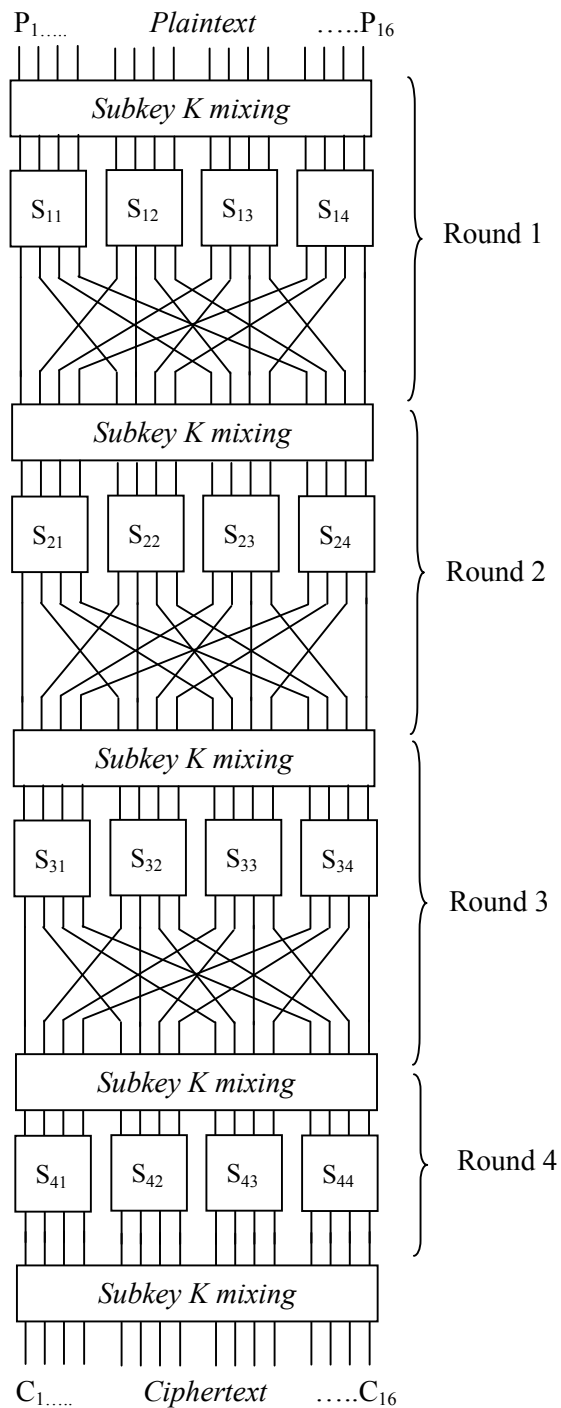
Meskipun pada awalnya analisis sandi ini digunakan untuk menyerang DES (karena menjadi standar internasional), analisis sandi ini juga sangat bermanfaat untuk mengukur kekuatan Algoritma enkripsi lainnya, karena banyaknya calon Algoritma yang didaftarkan untuk menjadi *Advanced Encryption Standard* (AES), didesain untuk menahan serangan analisis sandi ini.

### 1.1 Kode Sandi SPN Dasar

Kode rahasia yang akan dibahas menggunakan kode sandi SPN dasar, dimana kode sandi ini memiliki masukan 16 bit *plaintext*, dengan 4 ronde tahapan enkripsi. Setiap ronde terdiri dari :

1. substitusi (kotak  $S_1, \dots, S_{44}$ ),
2. transposisi (permutasi), misalnya bit ke-2 (keluaran dari ronde ke-1) menjadi bit ke-5 (masukan pada ronde ke-2),
3. pencampuran kunci, dimana setiap bit kunci di-XOR-kan dengan *plaintext* dan *ciphertext* sementara (keluaran kotak-S).

Struktur dasar ini diperkenalkan oleh Feistel pada tahun 1973 dan dijumpai pula pada DES dan Rijndael (AES). Untuk jelas struktur dari kode sandi SPN dasar dapat dilihat pada gambar di bawah ini.



**Gambar 1** Kode Sandi SPN Dasar.

## 1.2 Substitusi

*Plaintext* 16-bit yang kita miliki dipecah menjadi empat sub-blok 4-bit. Setiap sub-blok menjadi masukan ke dalam kotak-S 4 x 4 (kotak substitusi dengan masukan 4-bit dan keluaran 4-bit), yang dengan mudah dapat diimplementasikan dengan tabel *lookup* dari 16 nilai 4-bit, yang diberi indek dengan integer yang dinyatakan dengan masukan 4-bit. Sifat yang paling fundamental dari kotak-S adalah memiliki pemetaan yang tidak linear, sehingga, keluaran kotak substitusi tidak dapat dinyatakan sebagai operasi linear dari masukannya.

untuk kode ini, kita menggunakan pemetaan tidak linear yang sama untuk semua kotak-S. Pada DES, ke-8 kotak-S jalan satu ronde berbeda. sedangkan seluruh ronde menggunakan sekumpulan Kotak-S yang sama). Analisis sandi linear dapat digunakan untuk semua kasus ini. Pemetaan kotak-S dicantumkan pada tabel XX, diambil dari kotak-S pertama dan baris pertama milik DES. *Most Significant Bit* (MSB) terletak pada bit paling kiri.

Masukan	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Keluaran	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

**Tabel 1** Pemetaan kotak-substitusi (dalam hexadesimal)

Dari tabel substitusi jelas, bahwa bila masukan ke kotak-S adalah  $F_{16}$  (bit  $1111_2$ ), maka keluaran kotak-S adalah  $7_{16}$  ( $0111_2$ ), dan bila masukannya  $1100_2$  ( $C_{16}$ ), maka keluarannya adalah bit  $0101_2$  ( $5_{16}$ )

## 1.3 Permutasi

Permutasi merupakan transposisi bit-bit atau pertukaran urutan bit. Bila urutan bit-bit semula adalah 1,2,3,...,16 secara berurutan, maka setelah permutasi, urutan bit-bit menjadi 1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16. Bit 1 menempati bit paling kiri. Setelah permutasi, bit ke-2 menjadi bit ke-5, bit ke-3 menjadi bit ke-9 dan seterusnya.

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

**Tabel 2** Permutasi

Dalam Gambar 1. permutasi digambarkan dengan garis keluaran setiap kotak-S yang dihubungkan ke empat kotak-S berikutnya.

### 1.3 Pencampuran Kunci

Untuk memberikan peran kepada kunci dalam proses enkripsi dan dekripsi, maka dapat dilakukan pencampuran kunci yang dalam contoh ini berupa XOR antara bit-bit kunci (biasanya berupa subkey) dengan masukan blok data dalam setiap ronde. Kunci yang terletak pada awal dan akhir ronde diberikan untuk meningkatkan keamanan kode rahasia. Pada umumnya kunci-kunci di atas diturunkan dari kunci master yang lebih pendek melalui proses pengaturan kunci. Pada contoh ini, dianggap tidak ada proses pengaturan kunci, sehingga kunci-kunci di atas dianggap dibuat secara independen dan tidak berkorelasi.

## 2. Analisis Sandi Linear

Pada bagian ini, kita akan membicarakan proses attack kode rahasia menggunakan Analisis Sandi Linear. Analisis Sandi Linear menggunakan tingginya peluang terjadinya ekspresi linear yang mencakup bit-bit plaintext, bit-bit "*ciphertext*" (bukan ciphertext yang sesungguhnya melainkan masukan untuk ronde terakhir), dan bit-bit subkey. Metoda ini merupakan jenis *attack* adalah *known plaintext attack* : sehingga *attacker* harus mengetahui sekumpulan *plaintext* dari *ciphertext* yang berhubungan. Tetapi *attacker* tidak mempunyai cara untuk memilih *plaintext* yang mana, dan *ciphertext* yang berkaitan, yang tersedia. Dalam banyak aplikasi dan skenario sangat masuk akal apabila hal ini terjadi.

Ide dasarnya adalah pendekatan operasi bagian dari sandi dengan ekspresi yang linear. Kelinearan merujuk pada operasi bit modulo-2 (XOR). Ekspresi linear berbentuk :

$$X_{i_1} \oplus X_{i_2} \oplus \dots \oplus X_{i_u} \oplus Y_{j_1} \oplus Y_{j_2} \oplus \dots \oplus Y_{j_v} = 0 \quad \dots \dots \dots (1)$$

Dimana  $X_i$  menyatakan bit masukan ke-i dari  $X = [X_1, X_2, \dots]$  dan  $Y_j$  menyatakan bit ke-j dari keluaran  $Y = [Y_1, Y_2, \dots]$ . Persamaan ini menyatakan *exclusive-OR "sum"* dari bit-bit masukan u dan bit-bit keluaran v.

Pendekatan dalam Analisis Sandi Linear bertujuan untuk menentukan ekspresi dalam bentuk di atas yang memiliki peluang kejadian tinggi atau rendah. Jika kode menunjukkan kecenderungan bahwa persamaan (1) di atas memiliki peluang yang sangat tinggi atau rendah, maka hal ini menunjukkan bahwa kemampuan peng-acak-an (perandoman) sandi rendah.

Bila secara acak memilih nilai bit-bit  $u + v$  dan meletakkannya  $p$ , ada persamaan di atas, maka peluang persamaan di atas akan dipenuhi adalah  $\frac{1}{2}$ . Penyimpangan dari peluang  $\frac{1}{2}$  bagi suatu persamaan  $u + v$  di atas, akan menimbulkan kesempatan untuk eksploitasi pada Analisis Sandi Linear. Semakin besar penyimpangannya, semakin besar pula peluang Analisis Sandi Linear akan meng-attack sandi. Besar penyimpangan peluang dari  $\frac{1}{2}$  ini disebut *bias peluang linear*. Jadi, bila ekspresi di atas dipenuhi dengan peluang  $p_L$  untuk sebarang *plaintext* yang dipilih beserta *ciphertext*-nya, maka peluang biasanya adalah  $p_L - \frac{1}{2}$ . Semakin besar nilai mutlak  $|p_L - \frac{1}{2}|$ , semakin baik kemampuan Analisis Sandi Linear untuk memecahkan kunci dengan sedikit *plaintext* yang diketahui.

Terdapat beberapa cara untuk melakukan *attack* dengan Analisis Sandi Linear. Salah satu cara yaitu dengan meneliti konstruksi pendekatan linear yang berisi bit-bit *plaintext* ( $X$ ) dan masukan ke ronde terakhir (atau ekuivalen dengan keluaran dari ronde ke-3 untuk kasus ini) yang dinyatakan dengan  $Y$  pada persamaan (1). Karena *plaintext* acak, maka akan mengakibatkan  $Y$  juga acak.

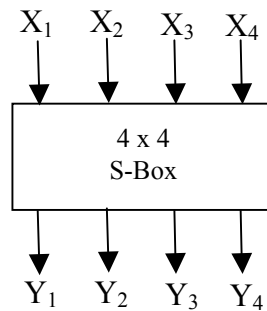
Persamaan (1) dapat ditulis ulang dan ekuivalen dengan penambahan XOR bit-bit sub-kunci di sisi kanannya. Bila pada persamaan (1), tertulis angka 0 pada sisi kanan, maka ini menunjukkan bahwa hasil XOR bit-bit kunci dianggap  $= 0$ , dengan peluang  $p_L$ .

Bila  $p_L = 1$ , maka persamaan (1) merupakan sandi yang memiliki kelemahan yang parah akibat fungsi linear yang sempurna berada dalam algoritma sandi. Bila  $p_L = 0$ , maka persamaan (1) menyatakan hubungan fungsi *affine* dalam sandi, yang juga menunjukkan kelemahan yang fatal. Untuk sistem penjumlahan mod-2, fungsi *affine* merupakan komplemen fungsi linear. Aproksimasi linear dan *affine* yang ditunjukkan oleh  $p_L > 1/2$  dan  $p_L < \frac{1}{2}$ , merujuk pada mudahnya sandi diserang oleh Analisis Sandi Linear. Kita juga dapat menganggap bahwa baik hubungan *affine* maupun linear yang terdapat dalam sandi adalah *linear*.

Pertanyaan berikutnya adalah : bagaimana kita dapat menyusun ekspresi yang sangat linear sehingga dapat kita eksploitasi. Ini dilakukan dengan memperhatikan sifat-sifat komponen *cipher* yang tidak linear dari Kotak-S. Dengan meneliti sifat kotak-S, kita dapat menyusun aproksimasi antara set masukan dan keluaran dalam kotak-S. Kemudian kita dapat menggabungkan aproksimasi linear dari kotak-kotak-S sedemikian sehingga bit-bit antara (bit-bit data yang berada di dalam *cipher*) dapat dihilangkan pengaruhnya, sehingga kita mendapatkan ekspresi linear yang memiliki bias besar dan hanya mengandung bit-bit *plaintext* dan bit-bit masukan ronde terakhir.

## 2.1 Analisis Komponen Cipher

Sebelum mempelajari *attack* lebih detail pada keseluruhan *cipher*, pertama-tama kita memerlukan pengetahuan kelemahan linear kotak-S. Perhatikan diagram blok kotak-S pada Gambar 2 dengan masukan  $X = [X_1, X_2, X_3, X_4]$  keluarannya  $Y = [Y_1, Y_2, Y_3, Y_4]$ .



**Gambar 2** Pemetaan kotak-S

Seluruh aproksimasi linear dapat diperiksa untuk menentukan kegunaannya, dengan menghitung peluang bias masing-masing aproksimasi. Karena itu, kita akan memeriksa seluruh ekspresi dalam bentuk persamaan (1) di mana  $X$  dan  $Y$  adalah masukan dan keluaran kotak-S.

Sebagai contoh, untuk kotak-S yang digunakan dalam *cipher*, perhatikan persamaan linear  $X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0$  atau ekuivalen dengan :  $X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4$

Menggunakan seluruh nilai masukan yang mungkin (16 nilai) untuk  $X$ , dan memeriksa nilai keluaran yang bersesuaian  $Y$ , dapat diamati bahwa untuk 12 kasus dari 16 kemungkinan, persamaan (1) dipenuhi. Karena itu, peluang biasanya adalah  $12/16 - \frac{1}{2} = \frac{1}{4}$ . Ini disajikan dalam Tabel 3.

$X_1$	$X_2$	$X_3$	$X_4$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$X_2 \oplus X_3$	$Y_1 \oplus Y_3 \oplus Y_4$	$X_1 \oplus X_4$	$Y_2$	$X_3 \oplus X_4$	$Y_1 \oplus Y_4$
0	0	0	0	1	1	1	1	0	0	0	1	1	1
0	0	0	1	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	1	1	0	0	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	0	1	1

1	0	1	0	0	1	1	0	1	1	0	0	1	0
1	0	1	1	1	1	0	0	1	1	1	1	0	1
1	1	0	0	0	1	0	1	1	1	0	1	0	1
1	1	0	1	1	0	0	1	1	0	1	0	1	0
1	1	1	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	1	1	1	0	0	1	1	0	1

**Tabel 3** Aproksimasi Linear Kotak-S

Demikian pula untuk persamaan  $X_1 \oplus X_4 = Y_2$ , terlihat bahwa peluang biasanya = 0.

Sedangkan untuk persamaan  $X_3 \oplus X_4 = Y_1 \oplus Y_4$  peluang biasanya  $2/16 - 1/2 = -3/8$ .

Dalam kasus terakhir, aproksimasi terbaik adalah fungsi *affine* (ditunjukkan oleh nilai negatif). Namun, keberhasilan attack tidak tergantung pada positif atau negatifnya nilai, melainkan tergantung pada nilai mutlak bias. Jadi aproksimasi *affine* sama baiknya dengan aproksimasi linear.

		OUTPUT SUM															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
I N P U T  S U M	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	-2	-2	0	0	-2	-6	-2	-2	0	0	-2	-2	0	0
	2	0	0	-2	-2	0	0	-2	-2	0	0	-2	-2	0	0	-6	-2
	3	0	0	0	0	0	0	0	0	-2	0	-2	-2	-2	-2	-2	-2
	4	0	-2	0	-2	-2	-4	-2	0	0	-2	0	-2	-2	-4	-2	0
	5	0	-2	2	0	-2	0	-4	-2	-2	0	-4	-2	0	-2	-2	0
	6	0	-2	2	-4	-2	0	0	-2	0	-2	-2	0	-2	0	0	-2
	7	0	-2	0	-2	-2	-4	-2	0	-2	0	-2	0	-4	-2	0	-2
	8	0	0	0	0	0	0	0	0	-2	-2	-2	-2	0	-2	-2	-6
	9	0	0	2	2	0	0	-2	-2	-4	0	-2	-2	-2	-4	-2	-2
	A	0	-4	2	2	-4	0	-2	-2	-2	-2	0	0	-2	-2	0	0
	B	0	-4	0	-4	-4	0	-4	0	0	0	0	0	0	0	0	0
	C	0	-2	4	-2	-2	0	-2	0	-2	0	-2	-4	0	-2	0	-2
	D	0	-2	2	0	-2	-4	0	-2	-4	-2	-2	0	-2	0	0	-2
	E	0	-2	2	0	-2	-4	0	-2	-2	0	0	-2	-4	-2	-2	0
	F	0	-2	-4	-2	-2	0	-2	0	0	-2	-4	-2	-2	0	-2	0

**Tabel 4** Aproksimasi Linear

Perhitungan satu demi satu secara lengkap dari seluruh aproksimasi linear kotak-S pada *cipher* diberikan pada Tabel 4 aproksimasi linear. Setiap elemen dalam tabel menyatakan jumlah kesesuaian antara persamaan linear yang dinyatakan dalam hexadesimal sebagai *Analisis Sandi Linear*

"jumlah masukan" dan jumlah bit-bit keluaran yang dinyatakan dalam hexadesimal sebagai "jumlah keluaran" dikurangi 8. Karena itu, pembagian suatu nilai elemen dengan 16 akan memberikan peluang bias untuk kombinasi linear tertentu dari bit-bit masukan dan keluaran. Nilai hexadesimal menyatakan jumlah, ketika dipandang sebagai nilai biner menunjukkan variabel yang dilibatkan ke dalam penjumlahan. Untuk kombinasi linear dari variabel masukan yang dinyatakan sebagai  $a_1.X_1 \oplus a_2.X_2 \oplus a_3.X_3 \oplus a_4.X_4$  di mana  $a_i \in \{0,1\}$  dan "." menyatakan AND biner, nilai hexadesimal menyatakan vektor biner  $a_1a_2a_3a_4$  dengan  $a_i$  sebagai MSB. Demikian pula untuk kombinasi linear bit-bit keluaran  $b_1Y_1 \oplus b_2Y_2 \oplus b_3Y_3 \oplus b_4Y_4$  di mana  $b_i \in \{0,1\}$ , nilai hexadesimal menyatakan vektor biner  $b_1b_2b_3b_4$ . Karena itu, bias dari persamaan linear  $X_3 \oplus X_4 = Y_1 \oplus Y_4$  (masuk hex 3 dan keluaran hex 9) adalah  $-6/16 = -3/8$  dan peluang bahwa persamaan linear dipenuhi adalah  $\frac{1}{2} - 3/8 = 1/8$ .

## 2.2 Menyusun Aproksimasi Linear

Sekali informasi aproksimasi linear kotak-S pada SPN disusun, kita memiliki data untuk diolah dengan menentukan aproksimasi linear dari keseluruhan cipher ke dalam bentuk persamaan (1). Ini dapat dicapai dengan menggabungkan aproksimasi linear kotak-kotak-S yang tepat. Dengan membangun aproksimasi linear yang mencakup bit-bit *plaintext* dan data dari keluaran ronde-terakhir-kedua dari kotak-S, *attack* menjadi sangat dimungkinkan untuk mendapatkan sebagian bit-bit kunci yang terletak setelah ronde terakhir. Contohnya sebagai berikut : Perhatikan suatu aproksimasi yang berisi  $S_{12}$ ,  $S_{22}$ ,  $S_{32}$ , dan  $S_{34}$  seperti yang ditunjukkan pada Gambar 3. Perhatikan bahwa disini, akan membangun ekspresi untuk 3 ronde pertama *cipher* dan bukannya 4 ronde penuh. Kita akan melihat bagaimana hal ini berguna untuk mendapatkan bit-bit kunci setelah ronde terakhir.

Kita menggunakan aproksimasi kotak-S berikut :

$$S_{12} : X_1 \oplus X_3 \oplus X_4 = Y_2 \quad \text{dengan peluang } 12/16 \text{ dan bias } +1/4$$

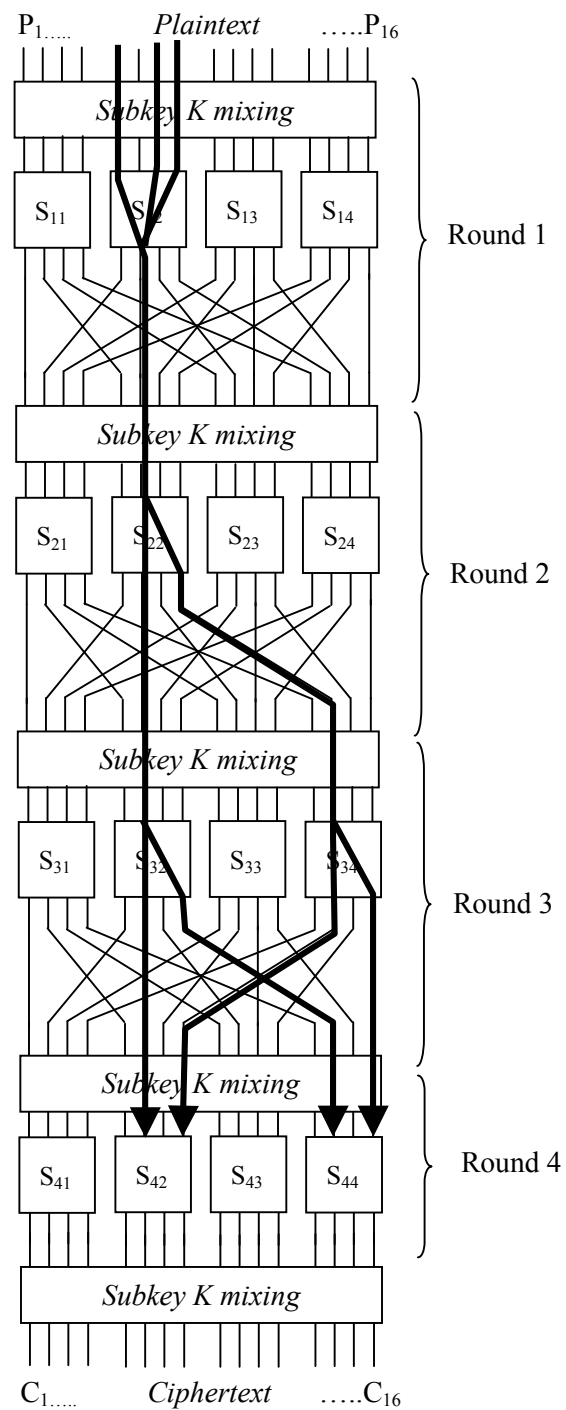
$$S_{22} : X_2 = Y_2 \oplus Y_4 \quad \text{dengan peluang } 4/16 \text{ dan bias } -1/4$$

$$S_{32} : X_2 = Y_2 \oplus Y_4 \quad \text{dengan peluang } 4/16 \text{ dan bias } -1/4$$

$$S_{34} : X_2 = Y_2 \oplus Y_4 \quad \text{dengan peluang } 4/16 \text{ dan bias } -1/4.$$

Ambil  $U_i(V_i)$  menyatakan blok 16-bit pada masukan(keluaran) dari kotak-S ronde ke-  $i$  sedangkan  $U_{i,j}(V_{i,j})$  menyatakan bit ke- $j$  dari blok  $U_i(V_i)$  (di mana bit-bit diberi nomor dari 1 hingga 16 dari kiri ke kanan pada gambar *cipher*). Kemudian ambil  $K_i$  yang

menyatakan bit-bit subkey blok yang di-XOR-kan pada masukan ke ronde  $i$ , dengan perkecualian  $K_5$  yang di-XOR-kan dengan keluaran ronde 4.



**Gambar 3** Aproksimasi Linier

Jadi,  $U_1 = P \oplus K_1$  di mana P menyatakan blok dari 16 bit plaintext sedangkan " $\oplus$ " menyatakan operasi XOR per bit. Dengan aproksimasi linear dari ronde pertama, kita memiliki

$$\begin{aligned} V_{1,6} &= U_{1,5} \oplus U_{1,7} \oplus U_{1,8} \dots\dots\dots(2) \\ &= (P_5 \oplus K_{1,5}) \oplus (P_7 \oplus K_{1,7}) \oplus (P_8 \oplus K_{1,8}) \end{aligned}$$

dengan peluang  $\frac{3}{4}$ .

Untuk aproksimasi pada ronde kedua, kita memiliki  $V_{2,6} \oplus V_{2,8} = V_{2,6}$

Dengan peluang  $\frac{1}{4}$ . Dan karena  $U_{2,6} = V_{1,6} \oplus K_{2,6}$ , maka kita mendapatkan aproksimasi dalam bentuk  $V_{2,6} \oplus V_{2,8} = V_{1,6} \oplus K_{2,6}$  dengan peluang  $\frac{1}{4}$ . Dan dengan mengkombinasikan persamaan ini dengan persamaan (2) yang memiliki peluang  $\frac{3}{4}$ , maka kita memiliki

$$V_{2,6} \oplus V_{2,8} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} = 0 \dots\dots\dots(3)$$

yang memiliki peluang  $\frac{1}{2} + 2(\frac{3}{4} - \frac{1}{2})(\frac{1}{4} - \frac{1}{2}) = \frac{3}{8}$  (Jadi memiliki bias  $-\frac{1}{8}$ ). Perhatikan bahwa kita menganggap aproksimasi kotak-kotak-S adalah saling bebas, yang meskipun tidak benar sepenuhnya, bekerja baik pada kebanyakan *cipher*.

Untuk ronde 3, kita mencatat bahwa

$$V_{3,6} \oplus V_{3,8} = U_{3,6}$$

dengan peluang  $\frac{1}{4}$ , dan

$$V_{3,14} \oplus V_{3,16} = U_{3,14}$$

dengan peluang  $\frac{1}{4}$ . Jadi, karena  $U_{3,6} = V_{2,6} \oplus K_{3,6}$  dan  $U_{3,14} = V_{2,8} \oplus K_{3,14}$ , maka

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus K_{3,6} \oplus V_{2,8} \oplus K_{3,14} = 0 \dots\dots\dots(4)$$

dengan peluang  $\frac{1}{2} + 2(\frac{1}{4} - \frac{1}{2})^2 = \frac{5}{8}$  (yang memiliki bias  $+\frac{1}{8}$ ).

Sekarang kombinasikan persamaan (3) dan (4) untuk menyertakan keempat aproksimasi kotak-S :

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus V_{2,6} \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus V_{2,8} \oplus K_{3,14} = 0$$

Perhatikan bahwa karena  $U_{4,6} = V_{3,6} \oplus K_{4,6}$  ;  $U_{4,8} = V_{3,14} \oplus K_{4,8}$  ;  $U_{4,14} = V_{3,8} \oplus K_{4,14}$  dan  $U_{4,16} = V_{3,16} \oplus K_{4,16}$ , maka kita dapat menuliskan persamaannya

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus a_k = 0 \dots\dots\dots(5)$$

Di mana  $a_k = K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus V_{3,24} \oplus K_{4,6} \oplus K_{4,8} \oplus K_{4,14} \oplus K_{4,16}$

Dan nilai  $a_k$  ini adalah 0 atau 1, tergantung pada kunci *cipher*. Persamaan di atas dipenuhi dengan peluang  $\frac{1}{2} + 2^3(\frac{3}{4}-\frac{1}{2})(\frac{1}{4}-\frac{1}{2})^3 = \frac{15}{32}$  (yang memiliki bias  $-\frac{1}{32}$ )

Perhatikan bahwa  $U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$

harus dipenuhi dengan peluang  $15/32$  atau  $(1-15/32)= 17/32$ , tergantung pada  $a_k = 0$  atau  $1$ . Dengan kata lain, kita memiliki aproksimasi linear dari tiga ronde pertama dari *cipher* dengan nilai mutlak bias sebesar  $1/32$ . Sekarang kita akan membahas bagaimana bias semacam ini dapat digunakan untuk menentukan beberapa bit kunci.

### 2.3 Mencari Bit-bit Kunci

Dengan aproksimasi linear (R-1) ronde dari *cipher* R ronde dengan bias peluang linear yang cukup besar, kita dapat melakukan *attack* terhadap *cipher* untuk mendapatkan bit-bit subkey setelah ronde terakhir. Untuk contoh, dimungkinkan mendapatkan bit-bit subkey  $K_5$  dari aproksimasi linear 3 ronde. Kita namakan bit-bit yang dicari dari subkey terakhir sebagai *target partial subkey*. Lebih khusus lagi, bit-bit *target partial subkey* adalah bit-bit yang berasal dari subkey terakhir yang berhubungan dengan kotak-S di ronde terakhir yang dipengaruhi oleh bit data yang dimasukkan ke dalam aproksimasi linear.

Proses berikutnya mencakup dekripsi parsial terhadap ronde terakhir *cipher*. Untuk seluruh nilai *target partial subkey* yang mungkin, bit-bit *ciphertext* di-XOR-kan dengan bit-bit *target partial subkey* dan hasilnya dibalikkan ke kotak-S yang bersesuaian. Hal ini dilakukan untuk semua sampel *plaintext / ciphertext*, dan setiap nilai *target partial subkey* diberi *counter*. Penambahan counter dilakukan untuk nilai target partial subkey tertentu, yaitu ketika ekspresi linear dipenuhi untuk bit-bit yang masuk ke dalam kotak-S ronde terakhir (yang ditentukan oleh dekripsi parsial) dan bit-bit *plaintext* yang diketahui. Nilai *target partial subkey* yang memiliki *counter* (penghitung) dengan beda terbesar dari setengah jumlah pasangan *plaintext/ciphertext*, dianggap menyatakan nilai bit-bit target *partial subkey*. Hal ini dianggap benar karena diasumsikan bahwa nilai *partial subkey* yang benar akan menghasilkan aproksimasi linear dipenuhi dengan peluang yang paling jauh dari  $1/2$  (tidak peduli apakah berada di atas atau di bawah  $1/2$  tergantung pada apakah ekspresi-nya *linear* atau *affine* dan ini tergantung pada nilai-nilai subkey yang belum dapat diketahui, yang secara implisit telah dimasukkan ke dalam persamaan linear. Subkey yang tidak tepat dianggap menghasilkan terkaan acak relatif pada bit-bit yang memasuki kotak-S pada ronde terakhir, dan sebagai akibatnya, ekspresi linear akan dipenuhi dengan peluang mendekati  $1/2$ .

Sekarang kita akan menggunakan konsep di atas untuk contoh Ekspresi linear persamaan (5) mempengaruhi masukan ke kotak-S,  $S_{42}$  dan  $S_{44}$  pada ronde terakhir. Untuk setiap

pasangan *plaintext/ciphertext*, kita mencoba seluruh nilai (256 kemungkinan) *target partial subkey*  $[K_{5,5}...K_{5,8} \ K_{5,13}...K_{5,16}]$ . Untuk setiap nilai *target partial subkey*, kita akan menaikkan *counter* apabila persamaan (5) dipenuhi, di mana kita menentukan nilai  $[U_{4,5} \dots \dots U_{4,8} \ , \ U_{4,13} \dots \dots U_{4,16}]$  dengan menjalankan *ciphertext* terbalik mundur melalui *target partial subkey* dan kotak-S  $S_{24}$  dan  $S_{44}$ . Counter yang memiliki deviasi terbesar dari setengah jumlah pasangan *plaintext/ciphertext*, diasumsikan merujuk kepada nilai yang benar. Deviasi positif atau negatif akan tergantung kepada nilai bit-bit subkey yang berada di dalam  $a_k$ . Apabila  $\sum a_k = 0$ , aproksimasi linear dari persamaan (5) akan memberi peluang  $< \frac{1}{2}$ , dan bila  $a_k = 1$ , maka persamaan (5) akan dipenuhi dengan peluang  $> \frac{1}{2}$ .

Simulasi *attack* terhadap *cipher* dilakukan dengan membangkitkan 10000 nilai *plaintext/ciphertext* yang diketahui dan mengikuti proses analisis sandi di atas untuk mendapatkan nilai subkey  $[K_{5,5}...K_{5,8}] = [0010]$  (hex 2) dan  $[K_{5,13}...K_{5,16}] = [0100]$  (hex 4). Dan seperti yang diharapkan, counter yang memiliki bias paling besar dari 5000, menunjukkan nilai target partial subkey  $[2,4]_{hex}$  dan sekaligus menunjukkan bahwa *attack* berhasil mendapatkan bit-bit kunci yang dicari, label 5 memperlihatkan ringkasan data yang diturunkan dari *counter subkey*. Data lengkap berisi 256 entri, dan satu entri untuk setiap nilai target partial subkey. Nilai dalam tabel menunjukkan besaran simpangan (bias) yang diturunkan dari  $|bias| = |count-5000| / 10000$

di mana count adalah penghitung yang berkaitan dengan *nilai target partial subkey*.

Sebagaimana dilihat dari hasil parsial ini, simpangan terbesar terjadi pada nilai *subkey parsial*  $[K_{5,5}...K_{5,8} \ ; \ K_{5,13}...K_{5,16}] = [2,4]_{hex}$ , jadi sudah 8 bit kunci yang dapat diperoleh dan *attack* ini dapat dilanjutkan untuk mencari sisa bit-bit kunci lainnya ( $80-8 = 72$  bit).

Bandingkan bila kita harus melakukan *brute force attack* terhadap *cipher* untuk mendapatkan bit-bit kuncinya.

Nilai bias yang ditentukan secara eksperimental sebesar 0,0336 sangat mendekati nilai yang diharapkan yaitu  $1/32 = 0,03125$ . Perhatikan bahwa, meskipun *target partial subkey* yang benar memiliki bias terbesar, nilai bias lain yang besar menunjukkan bahwa pengujian *target partial subkey* yang salah tidaklah tepat ekuivalen dengan data acak yang dimasukkan ke dalam *expresi linear* (di mana bias diharapkan sangat mendekati nilai nol). Ketidak-konsisten-an dalam eksperimen bias dapat terjadi karena beberapa sebab termasuk karena pengaruh sifat kotak-S yang mempengaruhi dekripsi parsial untuk nilai subkey parsial yang berbeda, ketidak-tepat-an asumsi bahwa hubungan antar kotak-S adalah saling bebas.

### 3. Kompleksitas Serangan

Merujuk pada kotak-S yang disertakan ke dalam aproksimasi linear sebagai *kotak-S aktif*. Pada Gambar 3, empat kotak-S pada ronde 1 sampai 3 yang diberi garis tebal merupakan kotak-S aktif. Peluang bahwa ekspresi linear dipenuhi akan dipengaruhi oleh bias peluang linear dalam kotak-S aktif dan sejumlah kotak-S aktif lainnya. Secara umum, semakin besar bias dalam masing-masing kotak-S, semakin besar pula bias ekspresi secara keseluruhan. Dan semakin sedikit kotak-S yang aktif, semakin besar pula bias ekspresi linear secara keseluruhan.

Anggap  $e$  menyatakan bias dari peluang  $\frac{1}{2}$ , bahwa ekspresi linear untuk *cipher* lengkap dipenuhi. Menurut Matsui bahwa jumlah *plaintext* yang diketahui yang dibutuhkan dalam *attack*, sebanding dengan  $e^{-2}$  dan bila  $N_L$  menyatakan jumlah *plaintext* yang diketahui dan yang diperlukan untuk *attack*.

Secara praktis, cukup layak untuk mengharapkan sejumlah kelipatan dari  $e^{-2}$  *plaintext* yang diketahui dan diperlukan, meskipun kompleksitas analisis sandi diukur dalam domain **waktu** dan **ruang**, kita merujuk pada data yang dibutuhkan untuk melakukan *attack* ketika membicarakan kompleksitas analisis sandi. Jadi, dapat diasumsikan bahwa apabila kita dapat memperoleh  $N_L$  *plaintext*, maka kita akan dapat memprosesnya. Pendekatan umum untuk memberikan ketahanan terhadap analisis sandi linear telah difokuskan pada optimasi kotak-S (yaitu meminimalkan bias terbesar) dan mencari struktur untuk memaksimalkan jumlah kotak-S yang aktif. Prinsip desain AES merupakan contoh yang istimewa dalam kasus ini.

Harus diperhatikan, konsep "terbukti" aman melawan Analisis Sandi Linear, biasanya didasarkan pada tidak ditemukannya aproksimasi linear yang memiliki peluang bias tinggi. Tetapi komputasi peluang aproksimasi linear semacam itu didasarkan pada asumsi bahwa setiap kotak-S saling bebas dan berdasar asumsi bahwa suatu skenario aproksimasi linear (yaitu sekumpulan kotak-S yang aktif) sudah cukup untuk menentukan *ekspresi linear* terbaik antara bit-bit *plaintext*, dan bit-bit data pada masukan ke ronde terakhir. Kenyataannya adalah bahwa aproksimasi kotak-S tidaklah saling bebas dan ini memberi pengaruh penting terhadap perhitungan peluang. Juga skenario aproksimasi linear yang memasukkan *plaintext* yang sama dan bit-bit masukan ronde terakhir namun menggunakan set kotak-S aktif yang berbeda, dapat memberi kombinasi yang mengakibatkan peluang linear yang lebih tinggi dari pada yang diperkirakan untuk satu set kotak-S yang aktif. Konsep ini dinamakan *linear hull*. Sebagai contoh, sejumlah aproksimasi linear mungkin

memberikan bias yang sangat kecil, sehingga *cipher* nampak kebal terhadap *attack linear*. Namun, ketika mereka digabungkan, ekspresi linear gabungan dari *plaintext* dan masukan ronde terakhir menghasilkan bias yang sangat besar. Meskipun demikian, pendekatan yang disebutkan dalam tulisan ini, cenderung bekerja dengan baik untuk banyak *cipher* karena asumsi saling bebas merupakan aproksimasi yang layak dan ketika satu skenario aproksimasi linear dari sekumpulan kotak-S aktif memiliki bias yang tinggi, maka skenario ini akan cenderung mendominasi *linear hull*.

---

#### DAFTAR PUSTAKA

- [1] Schneier B, *Applied Cryptography*, John Wiley & Sons, Inc, 1996
- [2] Stallings, William, *Data & Computer Communications 6th Edition*, Prentice-Hall, inc, 1996
- [3] Steffen, Andreas, *Secure Network Communication Part II Public Key Cryptography*, Zucher Hochschule Wintethur, 2000-2001