MENGUNGKAP LINEAR CRYPTANALYSIS PADA DES

Ginanjar Pramadita
NIM 13506014
Teknik Informatika Institut Teknologi Bandung
Jalan Ganesha10, Bandung
2008
e-mail: if16014@students.if.itb.ac.id

ABSTRAK

Makalah ini memfokuskan pembahasan mengenai bagaimana konsep *linear cryptanalysis* digunakan dalam men-kriptanalisis DES menggunakan *known plaintext attack*. Dalam pembahasannya, dijelaskan bagaimana menemukan satu atau lebih bit-bit kunci yang didapat melalui berbagai perhitungan probabilistik antara pasangan *plaintext-ciphertext*. Setelah didapat teknik menentukan bit-bit kunci, selanjutnya dijelaskan algoritma untuk memecahkan keseluruhan ciphertext.

Kata Kunci: *DES*, *linear cryptanalysis*, *differential cryptanalysis*, *known plaintext attack*, kunci simetri, enkripsi, dekripsi, kriptografi

1. Pendahuluan

Salah satu contoh *block cipher* yang sempat menjadi standar enkripsi yang sangat powerfull adalah DES (*Data EncryptionAlgorithm*). *Block cipher* ini dikembangkan oleh IBM pada tahun 1972 yang berdasarkan pada algoritma *Lucifer* yang dibuat oleh Horst Feistel. Namun demikian, sekuat apa pun algoritma enkripsi, selalu ada yang dapat mengkriptanalisisnya. Salah satu metode yang dapat digunakan untuk mengkriptanalisis DES, adalah dengan menggunakan *Linear Cryptanalysis*.

Linear Cryptanalysis adalah metode yang kriptanalisis yang *powerfull* yang diperkenalkan oleh Matsui pada tahun 1993. Sekilas, metode ini mirip dengan *differential cryptanalysis*. Metode ini dikategorikan sebagai *known plaintext attack*. Secara umum, metode ini menggunakan aproksimasi linear terhadap bit-bit pariti dari *plaintext*, *ciphertext*, dan *secret key*.

Berdasarkan percobaan yang Matsui lakukan (lihat [5]), didapatkan hasil sebagai berikut:

- 8 putaran DES dapat dipecahkan dengan 2^{21} *known-plaintext*
- 12putaran DES dapat dipecahkan dengan 2^{33} *known-plaintext*
- 16 putaran DES dapat dipecahkan dengan 2^{47} *known-plaintext*

Berikut ini notasi yang akan digunakan (berdasarkan notasi Matsui)

Tabel 1.1 notasi Matsui

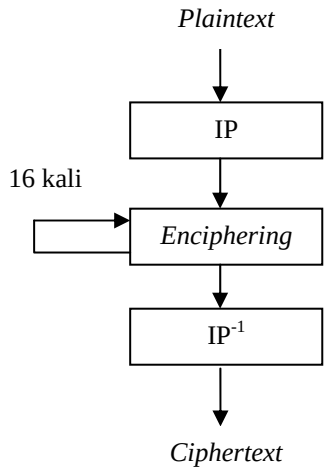
Notasi	Keterangan
P	64 bit *plaintext*
C	64 bit *chipertext* yang berkoresponden
P_H	32 bit P bagian kanan
P_L	32 bit P bagian kiri
C_H	32 bit C bagian kanan
C_L	32 bit C bagian kiri
X_i	32 bit nilai *intermediate* pada putaran ke-*i*
K_i	48 bit upa kunci pada putaran ke-*i*
$F_i(X_i, K_i)$	Fungsi *f* pada putaran ke-*i*
$A[i]$	Bit ke-*i* dari A
$A[i, j, ...,k]$	$A[i] \oplus A[j] \oplus ... \oplus A[k]$

2. DES (*Data Encryption Standard*)

DES (lihat detailnya pada [1]) adalah suatu standar encripsi yang digolongkan ke dalam kriptografi kunci simetri dan termasuk *block cipher* dengan panjang blok 64 bit.

Ada tiga langkah umum pada DES, yaitu *Initial Permutation* yitu proses pengacakan urutan bit-bit

pada *plaintext*, 16 putaran *enciphering*, dan _inverse *Initial Permutation*. (lihat **Gambar 1.1**)



Gambar 1.1 Skema Global DES

2.1 Initial Permutation

Initial Permutation (IP) digunakan untuk mengacak urutan bit-bit *plaintext* sesuai dengn matriks permutasi (lihat **Apendiks A.1**)

Caranya adalah dengan memindahkan bit sesuai dengan angka pada tiap elemen matriks ke posisi urutan elemen matriks. Contohnya pada elmen ke-2 (tertulis 50) artinya memindahkan bit ke-50 ke posisi bit ke-2.

2.2 Pembangkitan Kunci Internal

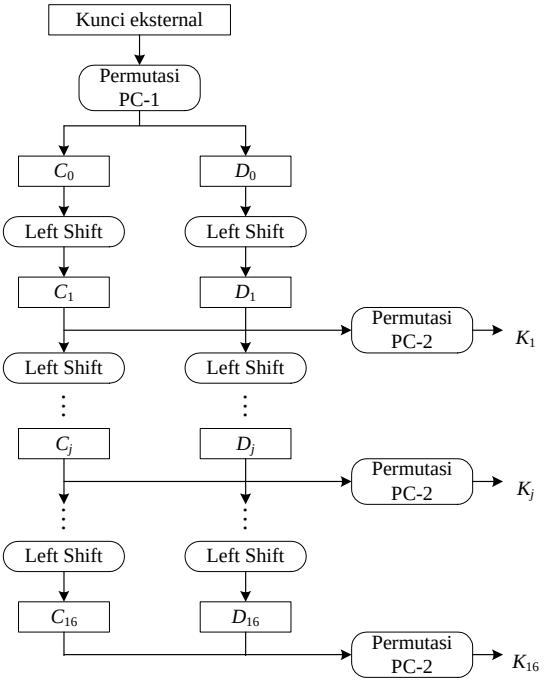
Kunci internal adalah kunci-kunci yang digunakan pada setiap putaran *enciphering* sehingga akan terdapat 16 kunci internal: K_1, K_2, ..., K_3.

Mula-mula, kunci eksternal (64 bit) diacak dengan matriks permutasi kompresi PC-1 (lihat **Apendiks A.2**) menghasilkan 56 bit. Teknik pengacakan sama dengan *initial permutation*. Setelah diacak, selanjutnya kunci dibagi dua yang masing-masing panjangnya 28 bit. Setiap bagian, dilakukan *left shift* dengan aturan pada **Tabel 2.1**.

Setelah itu, setiap bagian di acak kembali dengan matriks permutasi kompresi PC-2 1 (lihat **Apendiks A.3**). Total langkah pembangkitan kunci internal dapat dilihat pada **Gambar 2.1**.

Tabel 2.1 Pergeseran Bit Pada Pembangkitan Kunci Internal

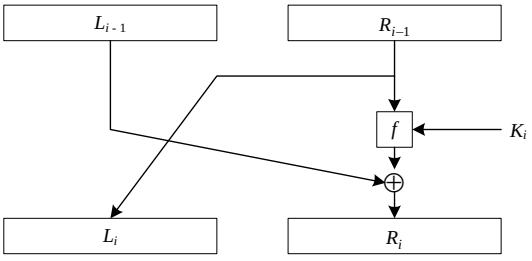
Putaran ke-i	Jumlah Pergeseran Bit	Putaran ke-i	Jumlah Pergeseran Bit
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1



Gambar 2.1 Pembangkitan Kunci Internal

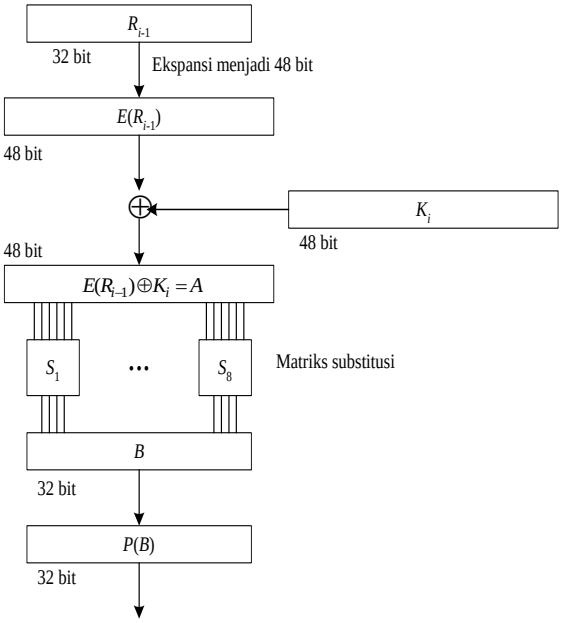
2.3 Eniphering

Setiap blok mengalami 16 putaran *enciphering* dan setiap putaran merupakan jaringan Feistel (Lihat **Gambar 2.2**) yang secara matematis dinyatakan sebagai $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$



Gambar 2.2 Jaringan Feistel

Komputasi funcsi *f* dapat dilihat pada **Gambar 2.3**



Gambar 2.3 Komputasi Fungsi *f*

Fungsi espansi E digunakan untuk memperluas blok R_{i-1} 32 bit menjadi 48 bit dan dilakukan melalui matriks permutasi pada **Apendiks A.4**

Setelah melalui expansi, selanjutnya masuk ke tahapan delapan *S-box*, yaitu proses subtitusi menggunakan delapan *S-box* $S_{1...8}$ yang menerima masukan 6 bit dan menghasilkan keluaran 4 bit.

Subtitusi dilakukan beruntun: 6 bit pertama dengan S_1, 6 bit kedua dengan S_2, dan seterusnya. Adapun kedelapan *S-box* dapat dilihat pada [1] halaman 128.

Keluaran proses subtitusi adalah 48 bit yang selanjutnya menjadi masukan untuk proses permutasi dengan matriks permutasi pada **Apendiks A.5**

2.4 Inverse Initial Permutation

Permutasi terakhir yang dilakukan terhadap gabungan blok kiri dan kanan. Permutasi dilakukan dengan menggunakan matriks permutasi IP^{-1} pada **Apendiks A.6**.

3. Perbandingan Linear Cryptanalysis dengan Differential Cryptanalysis

Linear cryptanalysis memiliki metodologi yang serupa dengan *differential cryptanalysis*. Sebelum melanjutkan ke pembahasan mengenai *linear cryptanalysis*, ada baiknya kita tinjau terlebih dahulu perbandingannya dengan *differential cryptanalysis*.

Pada **Tabel 3.1** terlihat kemiripan tersebut (lihat [6] dan [7])

Tabel 3.1 Perbandingan Differential dan Linear Cryptanalysis

Differential	Linear
Karakteristik diferensial	Approksimasi linear
Aturan karakteristik diferensial: *Match the differences, multiply the probabilities*	Aturan aproksimasi linear: *Match the mask, multiply the imbalance*
Algoritma mencari karakteristik terbaik	Algoritma mencari approksimasi linear terbaik

Disamping memiliki kemiripan metodologi, keduanya memiliki dualitas operasi percabangan XOR Perbedaan mendasar dari keduanya adalah bahwa *differential cryptanalysis* bekerja dengan blok-blok bit sedangkan *linear cryptanalysis* bekerja hanya terhadap bit-bit tunggal.

4. Linear Cryptanalysis

Pada *linear cryptanalysis* dibutuhkan akses terhadap sepasang (P_i, C_i), i = 1, ...N dari *known plaintext* dan *ciphertext* yang berkoresponden.

Berikut ini adalah langkah umum untuk mengeksploitasi korelasi diantara bit-bit pada *P*, *C*, dan kunci:

- Cari persamaan biner dari bit-bit P_i, C_i, dan kunci (aproksimasi linear) yang menunjukan hubungan *non-trivial* di antara mereka (*bias*).
- Kumpulkan sejumlah besar sampel P_i dan C_i
- Mencoba semua nilai dari koleksi P_i dan C_i pada persasman.
- Ambil kunci yang memaksimumkan *bias* sebagai kunci yang benar.

Misalkan A adalah tabel *exhaustive conversion* antara cara standar dan cara Matsui untuk men-denotasikan bit-bit setiap kata. Prinsip *linear cryptanalysis* adalah dengan aproksimasi non-linear

block cipher karena pada dasarnya setiap operasi pada DES bersifat linear kecuali *S-box*. Berikut ini expresi linear yang akan digunakan:

$$P[i_1, i_2, \dots, i_a] \oplus C[j_1, j_2, \dots, j_b] = K[k_1, k_2, \dots, k_c] \quad (3)$$

Dengan *P*, *C*, *K* adalah bit-bit dari *plaintext*, *ciphertext*, dan *key*. Sedangkan *i*, *j*, dan *k* adalah index bit dengan domain:

$$i \in \{1 \dots 64\}, j \in \{1 \dots 64\}, k \in \{1 \dots 56\}$$

Persamaan (3) akan memenuhi probabilitas $p \neq \frac{1}{2}$ (*bias approximation*) untuk menemukan pasangan antara random *plaintext P* dengan *ciphertext C* yang ber-koresponden. Oleh karena itu, efektivitas persamaan (3) adalah

$$\text{Jika } p = \tfrac{1}{2} + \epsilon, \text{ maka } \epsilon = \left| p - \tfrac{1}{2} \right| \quad (4)$$

4.1 Mendapatkan Informasi Satu Bit Kunci

Misalnya *N* adalah random *known plaintext*, *T* adalah banyaknya *plaintext* sedemikian sehingga ruas kiri pada persamaan (3) bernilai nol, maka:

Algoritma 1

```
if (T > N/2) then
  if (p > 1/2) then
    K[k] = 0
  else
    K[k] = 1
  end
else
  if (p > 1/2) then
    K[k] = 1
  else
    K[k] = 0
  end
end
```

Lemma

Misalnya probabilitas expresi linear $p = \frac{1}{2} + \epsilon$, dengan $\epsilon > 0$ dan *K*[*k*] = 0.

X	*Px*[*X* = *x*]
0	$1/2 + \epsilon$
1	$1/2 - \epsilon$

Jika random variable T adalah jumlah dari N yang tersebar dan mutual independen terhadap variable X_i, maka nilai ekspektasi E(T) dapat dihitung sebagai berikut:

$$\begin{aligned} E(T) &= N\sum_{x=0}^{1} xP(x) \\ &= N(0(\tfrac{1}{2} + \varepsilon) + 1(\tfrac{1}{2} - \varepsilon)) \\ &= N(\tfrac{1}{2} - \varepsilon) \end{aligned} \quad (4)$$

$$Var(T) = N\left(\tfrac{1}{4} - \varepsilon^2\right) \quad (5)$$

Dengan menerapkan terema *Chebyshev* (lihat [3] hal. 131), probabilitas kesalahan dapat dihitung sebagai $p_f = 1 - p_s$ dengan p_s adalah probabilitas kesuksesan sehingga didapatkan batas sebagai berikut:

$$1 - p_s \leq P_T\left(\left|T - E(T)\right| \geq N_\varepsilon\right) \leq \frac{Var(T)}{N^2\varepsilon^2}$$

$$1 - p_s \leq P_T\left(\left|T - E(T)\right| \geq N_\varepsilon\right) \leq \frac{1}{N}\left(\frac{1}{4\varepsilon^2} - 1\right) \quad (6)$$

Jadi dari ketaksamaan (6) dapat terlihat bahwa probabilitas sukses p_s akan meningkat pada saat *N* dan atau ε meningkat.

4.2 Mendapatkan Informasi Beberapa Bit Kunci

Dalam praktiknya, *known-plaintext attack* DES memberikan lebih dari satu bit kunci. Matsui menggunakan n – 1 putaran aproksimasi linear yang lebih efektif. Dengan kata lain, putaran akhir menggunakan kandidat upa kunci $K_{16}(i)$, yang membangun aproksimasi linear yang menerima sabuah fungsi *f* sebagai intinya.

Berikut ini adalah ekspresi yang sedikit berbeda dari (3) yang menangani putara n – 1 DES

$$P[i_1, i_2, \dots, i_{64}] \oplus C[j_1, j_2, \dots, j_{64}] \oplus F_n(C_L, K_n)[l_1, l_2, \dots, l_{16}] = K[k_1, k_2, \dots, k_{56}] \quad (7)$$

Jika sebuah subtitusi kandidat upa kunci K_{16} salah, efektivitas dari ekspresi linear sebelumnya akan menurun.

Algoritma di bawah ini menerima sejumlah probabilitas kesuksesan *i* bit yang berasal dari kandidat upa kunci yang benar dan satu bit

informasi mengenai jumlah beberapa bit-bit kunci yang membangun ruas kanan pada ekspresi linear.

Algoritma 2

```
foreach subkey candidate K[i] of K do
  T[i] := banyaknya plaintext
  sedemikian sehingga ruas kiri pada
  aproksimasi linear bernilai 0
end

Tmax := max{T}
Tmin := min{T}

if(|Tmax – N/2| > |Tmin – N/2|) then
  ambil subkey candidate yang
  berkorespondensi dengan Tmax dan
  memenuhi K[k] = 0 pada saat p > 1/2
  atau 1 jika sebaliknya
end

if(|Tmax – N/2| < |Tmin – N/2|) then
  ambil subkey candidate yang
  berkorespondensi dengan Tmax dan
  memenuhi K[k] = 1 pada saat p > 1/2
  atau 0 jika sebaliknya
end
```

Dengan 2^{47} pasang *known plaintext* dan *ciphertext*, Matsui mengestimasi probabilitas kesuksesan serangan sebesar 97.7% dengan kompleksitas 2^{42} evaluasi DES untuk bagian *exhaustive key search*.

5. Serangan Terhadap 16-Putaran DES

Untuk dapat memecahkan 16-putaran DES, Matsui (lihat [5]) menunjukan bahwa sangatlah mungkin untuk meningkatkan serangan yang dijelaskan sebelumnya. Pertama-tama, ia menggunakan ekspresi linear pada 14-putaran DES. Setiap persamaan memiliki dua *S-box* (*S-box* 1 dan *S-box* 5) yang aktif dan dapat menutupi 13 bit kunci atau total 26.

5.1 14-Putaran Aproksimasi Linear

Matsui telah menemukan dua 14-putaran ekspresi linear yang merupakan pusat dari serangan.

$$P_L[7,18,24] \oplus C_H[7,18,24,29] \oplus C_L[15] = K_2[22] \oplus K_3[44] \oplus K_4[22] \oplus K_6[22] \oplus K_7[44] \oplus K_8[22] \oplus K_{10}[22] \oplus K_{11}[44] \oplus K_{12}[22] \oplus K_{14}[22] \quad (8)$$

dan

$$C_L[7,18,24] \oplus P_H[7,18,24,29] \oplus P_L[15] = K_{13}[22] \oplus K_{12}[44] \oplus K_{11}[22] \oplus K_9[22] \oplus K_8[44] \oplus K_7[22] \oplus K_5[22] \oplus K_4[44] \oplus K_3[22] \oplus K_1[22] \quad (9)$$

Persamaan (8) dan (9) memiliki probabilitas aproksimasi sebesar $\frac{1}{2} - (1.19)2^{-21}$.

Apabila persamaan (8) dan (9) diimplementasikan pada 14 fungsi *F* berurutan dari putaran ke-2 hingga ke-15 pada DES, maka akan didapat dua linear aproksimasi ahir sebagai berikut:

$$P_H[7,18,24] \oplus C_H[15] \oplus C_L[7,18,24,29] \oplus F(C_L, K_{16})[15] \oplus F(P_L, K_1)[7,8,24] = K_3[44] \oplus K_4[44] \oplus K_5[22] \oplus K_7[22] \oplus K_8[44] \oplus K_8[22] \oplus K_{11}[22] \oplus K_{12}[44] \oplus K_{13}[22] \oplus K_{15}[22] \quad (10)$$

dan

$$C_H[7,18,24] \oplus P_H[15] \oplus P_L[7,18,24,29] \oplus F(P_L, K_1)[15] \oplus F(C_L, K_{16})[7,8,24] = K_{14}[22] \oplus K_{13}[44] \oplus K_{12}[22] \oplus K_{10}[22] \oplus K_9[44] \oplus K_8[22] \oplus K_6[22] \oplus K_5[44] \oplus K_4[22] \oplus K_2[22] \quad (11)$$

5.2 Bit-Bit Teks dan Kunci yang Efektif

Konsep dari bit-bit teks dan kunci yang efektif yaitu bit-bit yang mempengaruhi ruas kiri dari aproksimasi linear. Berikut ini perhitungan nilai *Xor* dari sejumlah bit-bit teks atau kunci yang efektif:

P_L[11], P_L[12], P_L[13], P_L[14], P_L[15], P_L[16], C_L[0], C_L[27], C_L[28], C_L[29], C_L[30], C_L[31], P_H[7,18,24], C_L[7,18,24,29] ⊕ C_H[15], K_1[18], K_1[19], K_1[20], K_1[21], K_1[22], K_1[23], K_{16}[42], K_{16}[43], K_{16}[44], K_{16}[45], K_{16}[46], K_{16}[47]

(12)

dan

C_L[11], C_L[12], C_L[13], C_L[14], C_L[15], C_L[16], P_L[0], P_L[27], P_L[28], P_L[29], P_L[30], P_L[31], C_H[7,18,24], P_L[7,18,24,29] ⊕ P_H[15], K_{16}[18], K_{16}[19], K_{16}[20], K_{16}[21], K_{16}[22], K_{16}[23], K_{16}42], K_1[43], K_1[44], K_1[45], K_1[46], K_1[47]

(13)

Perlu diperhatikan di sini bahwa 13 bit teks dapat digunakan untuk menurunkan 12 bit kunci dari ruas kiri pada setiap persamaan. Oleh karena itu akan didapatkan 26 bit kunci rahasia dari kedua operasi menggunakan 26 bit teks.

5.3 Improvisasi Algoritma

Berikut ini akan dijelaskan langkah-langkah selanjutnya untuk memecahkan DES berdasarkan ekspresi linear yang didapat dari 3.1.

- Untuk setiap kandidat upa kunci dan untuk kedua aproksimasi linear, dihitung banyaknya ruas kiri dari ekspresi linear yang bernilai 0.
- Setelah mendapat daftar kandidat upa kunci setiap ekspresi linear selanjutnya diurutkan secara *descending* berdasarkan nilai *bias* dalam aproksimasi linear.
- Selanjutnya adalah menggabungkan kedua daftar upa kunci untuk mendapatkan 24 bit kandidat upa kunci terurut. Dengan cara ini akan mereduksi jumlah pasangan *plaintext* dan *ciphertext* dari 2^{47} menjadi 2^{43}.

Algoritma langkah-langkah di atas adalah sebagai berikut:

Algoritma 3

```
Sediakan 2^13 counter untuk setiap
ekspresi linear C1[i] dan C2[i] dengan
0 ≤ i ≤ 2^13 dan inisialisasi dengan 0.

{Setiap indeks i berkoresponden dengan
state 13 efektif bit text}
foreach 2^43 pasang plaintext-ciphertext
do
  Hitung nilai i1 dan i2 penggunakan P
  dan C untuk ekspresi linear  l1 dan
  l2.
  C1[i1] = C1[i1] + 1
  C2[i2] = C2[i2] + 1
end

Sediakan 2^12 counter untuk setiap
ekspresi linear K1[k] dan K2[k] dengan
0 ≤ k ≤ 2^12 dan inisialisasi dengan 0.

{Setiap indeks k berkoresponden dengan
state 12 efektif bit text}
foreach k1, k2 do
  K1[k1] := jumlah C1[x] dimana
            l1(px, cx, k1) = 0
  K2[k2] := jumlah C2[x] dimana
            l2(px, cx, k1) = 0

end
```

```
Urutkan K1 dan K2 secara descending
berdasarkan nilai |Kx[y] - 2^42| dengan
x ∈ {1,2} dan 0 ≤ y ≤ 2^12.

Akan didapat dua list dengan 2^12
counter yang dinotasikan dengna S1[r]
dan S2[r] dengan 0 ≤ r ≤ 2^12.

{Ambil informasi bit teraskhir dari
setiap kandidat upa kunci}
foreach Sx[r] do
  if (|Sx[r] – 2^42| ≤ 0) then
    tebak ruas kanan ekspresi linear
    dengan x adalah 0
  end
  if (|Sx[r] – 2^42| > 0) then
    tebak ruas kanan ekspresi linear
    dengan x adalah 1
  end
end


{Menggabungkan dua list}
F(r1, r2) := (r1 + 1) * (r2 + 1)
{rx adalah index yang berkoresponden
dengan list terurut x}

L adalah gabungan kedua list S1 dan S2
sesuai dengan kenaikan nikai F()

foreach kandidat upakunci pada L do
  cari 30 bit sisanya
  if (kunci benar ditemukan) then
    exit
  end
end
```

6. Kesimpulan

DES adalah salah satu standar enkripsi yang *cukup* baik karena memiliki beberapa parameter keamanan yang sulit ditembus diantaranya permutasi awal dan akhir, 16 putaran enciphering, dan 8 *S-box*.

Kelemahan DES adalah terletak pada sifat kelinearannya sehingga dengan perhitungan probabilitas terhadap nilai *bias* dan pasangan *plaintext-ciphertext* yang diketahui, dapat dilakukan kriptanalisis

Linear cryptanalysis merupakan algoritma yang jauh lebih mangkus dibanding *brute force*. Namun, untuk dapat melakukan kriptanalisis dengan lebih cepat, perlu didukung oleh hardware yang menunjang.

DAFTAR PUSTAKA

[1] Rinaldi Munir. *Diktat Kuliah Kriptografi Program Studi Teknik Informatika Institut Teknologi Bandung*. 2006

[2] Pascal Junod. *Linear Cryptanalysis of DES*

[3] Walpole, Myers, Ye. *Probability and Statistics,*. 8th Ed. Pearson Prantice Hall.

[4] E. Biham and A. Shamir. Differential Cryptanalysis of the Data Encryption

[5] M. Matsui. *Linear cryptanalysis method for DES cipher*, in Advances in Cryptol-ogy – EUROCRYPT'93 T. Helleseth, ed. 1993.

[6] Alex Biryukov, Christophe De Canni`ere. *Linear Cryptanalysis.*

[7] Ali Aydin Selcuk, *Differential & Linear Cryptanalysis*.

APENDIKS

A. Data Encryption Standard

A.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

A.2 Matriks Permutasi Kompresi PC-1

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

A.3 Matriks Permutasi Kompresi PC-2

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

A.4 Matriks Permutasi Expansi

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	31	32	1

A.5 Matriks Permutasi Pasca S-Box

16	7	20	21	29	12	28	17	1	15	23	26	5	8	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

A.6 Inverse Permutasi (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