Studi dan Analisis Keamanan *Data Encryption Standard* Dengan Menggunakan Teknik *Differential Cryptanalysis*

Hengky Budiman – NIM : 13505122

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15122@students.if.itb.ac.id

Abstrak

Data Encryption Standard (DES) merupakan sebuah algoritma enkripsi sandi blok dengan kunci simetrik. DES dipilih oleh NBS (*National Bureau of Standard*) sebagai algoritma enkripsi standard yang digunakan di badan pemerintahan Amerika Serikat untuk penyebaran informasi pada tahun 1977. Algoritma ini kemudian dipublikasikan secara luas, dan telah digunakan pada banyak aplikasi di seluruh dunia.

Sekarang ini, DES tidak lagi dianggap aman karena beberapa hal, salah satunya disebabkan karena ukuran kuncinya yang pendek (56 bit), karenanya DES sekarang digantikan oleh algoritma lain yang lebih canggih seperti AES (*Advanced Encryption Standard*). Namun demikian, algoritma DES merupakan algoritma dasar yang menggunakan prinsip – prinsip penting di kriptografi seperti *confussion*, *diffusion* dan jaringan *feistel*. Algoritma DES juga mendorong berbagai badan akademik untuk mempelajari kriptografi, terutama di fokus memecahkan algoritma enkripsi sandi blok. Studi – studi tersebut kemudian menjadi semakin berkembang, sehingga ditemukanlah berbagai teknik untuk memecahkan DES. Salah satu teknik yang ditemukan dan banyak dipakai adalah *Differential Cryptanalysis*.

Makalah ini membahas mengenai bagaimana teknik *Differential Cryptanalysis* dapat digunakan untuk memecahkan algoritma enkripsi DES.

Kata kunci: kriptografi, *Data Encryption Standard*, algoritma enkripsi sandi blok, *Differential Cryptanalysis*, kriptanalisis

1. Pendahuluan

DES merupakan salah satu algoritma kriptografi *cipher block* dengan ukuran blok 64 bit dan ukuran kuncinya 56 bit. Algoritma DES dibuat di IBM, dan merupakan modifikasi daripada algoritma terdahulu yang bernama *Lucifer*. *Lucifer* merupakan algoritma *cipher block* yang beroperasi pada blok masukan 64 bit dan kuncinya berukuran 128 bit. Pengurangan jumlah bit kunci pada DES dilakukan dengan alasan agar mekanisme algoritma ini bisa diimplementasikan dalam satu chip.[4]

DES pertama kali dipublikasikan di *Federal Register* pada 17 Maret 1975. Setelah melalui banyak diskusi, akhirnya algortima DES diadopsi sebagai algoritma standar yang digunakan oleh NBS (*National Bureau of Standard*s) pada 15 Januari 1977 [5]. Sejak saat itu, DES banyak digunakan pada dunia penyebaran informasi untuk melindungi data agar tidak bisa dibaca oleh orang lain.

Namun demikian, DES juga mengundang banyak kontroversi dari para ahli di seluruh dunia[4]. Salah satu kontroversi tersebut adalah *S-Box* yang digunakan pada DES. *S-Box* merupakan bagian vital dari DES karena merupakan bagian yang paling sulit dipecahkan. Hal ini disebabkan karena *S-Box* merupakan satu – satunya bagian dari DES yang komputasinya tidak linear. Sementara itu, rancangan dari *S-Box* sendiri tidak diberitahukan kepada publik. Karena itulah, banyak yang curiga bahwa *S-Box* dirancang sedemikian rupa sehingga memberikan *trapdoor* kepada NSA agar NSA bisa membongkar semua *ciphertext* yang dienkripsi dengan DES kapan saja. Kontroversi yang kedua adalah jumlah bit pada kunci DES yang dianggap terlalu kecil, hanya 56 bit. Akibatnya DES rawan terhadap serangan *brute force*.

Namun demikian, DES tetap digunakan pada banyak aplikasi seperti pada enkripsi PIN (*Personal Identification Numbers*) pada mesin ATM (*Automatic Teller Machine*) dan transaksi perbankan lewat internet. Bahkan, organisasi – organisasi pemerintahan di Amerika seperti *Department of Energy*, *Justice Department*, dan

Federal Reserve System menggunakan DES untuk melindungi penyebaran data mereka.

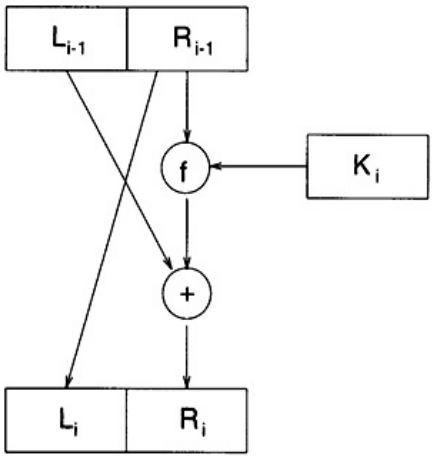
2. Cara Kerja DES

Algoritma dari DES mencakup 3 proses utama, yaitu:[3]

a. Diberikan *plaintext* x, sebuah *bitstring* x_0 dibentuk dengan permutasi bit – bit pada x sesuai dengan sebuah *initial permutation* (IP) yang telah ditentukan sebelumnya dan bersifat tetap. Kita dapat menulis $x_0 = IP(x) = L_0R_0$ di mana L_0 menunjukkan 32 bit pertama dari x_0 dan R_0 menunjukkan 32 bit sisanya.

b. Dilakukan 16 iterasi dari proses berikut:

$$1 \le i \le 16$$



Gambar 1. Struktur Jaringan Feistel pada DES

Tanda ⊕ menunjukkan *exclusive or* dari 2 buah *bitstring*. *F* adalah sebuah fungsi yang akan dijelaskan di bagian berikutnya. $K_1, K_2, \ldots, K_{16}$ adalah *bitstring* dengan panjang masing – masing 48 bit yang dihitung dengan fungsi lain. $K_1, K_2, \ldots, K_{16}$ disebut juga dengan *key schedule*.

c. Lakukan invers dari permutasi sebelumnya, yaitu IP^{-1} pada *bitstring* $R_{16}L_{16}$, maka kita akan mendapatkan *ciphertext* *y*, $y = IP^{-1}(R_{16}L_{16})$. Perhatikan urutan yang dibalik pada L_{16} dan R_{16}.

Pada bagian di atas, telah dikenalkan fungsi *f*. Fungsi *f* menerima 2 buah *input*, yaitu *input* pertama *A*, dengan panjang 32 bit, dan *input* kedua *J* dengan panjang 48 bit. Fungsi tersebut kemudian akan menghasilkan sebuah *output bitstring* dengan panjang 32 bit. Fungsi tersebut kemudian melakukan hal – hal berikut:

a. Argumen pertama yaitu *A* diekspansi menjadi *bitstring* dengan panjang 48 bit sesuai dengan *expansion function* E yang telah ditentukan sebelumnya. E(*A*) terdiri dari 32 bit dari *A*, dipermutasi sedemikian rupa, dengan 16 buah bit muncul dua kali.

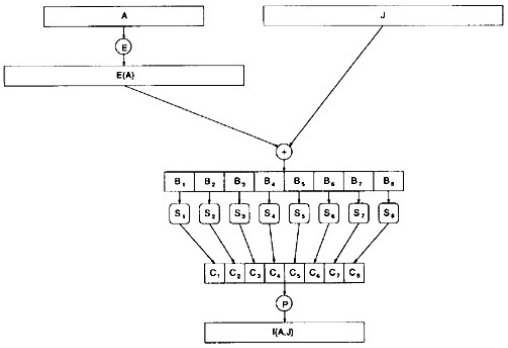
b. Hitung E(*A*) ⊕ *J* dan tulis hasilnya sebagai konkatenasi dari 8 buah *bitstring* dengan masing – masing bitstirng panjangnya 6 bit, $B = B_1B_2B_3B_4B_5B_6B_7B_8$.

c. Langkah selanjutnya menggunakan 8 *S-Boxes* $S_1, \ldots, S_8$. Setiap S_i adalah sebuah array 4×16 tetap yang nilai setiap elemennya berkisar antara 0 - 15. Apabila diberikan masukan berupa *bitstring* dengan panjang 6 bit, misalkan $B_j = b_1b_2b_3b_4b_5b_6$, kita menghitung $S_j(B_j)$ sebagai berikut:
   - 2 Bit b_1b_6 menentukan baris *r* dari S_j ($0 \le r \le 3$)
   - 4 bit $b_2b_3b_4b_5$ menentukan kolom *c* dari S_j ($0 \le c \le 15$).
   - $S_j(B_j)$ adalah nilai elemen $S_j(r, c)$, ditulis dalam representasi *binary* dengan *bitstring* dengan panjang 4 bit.

   Kita menghitung nilai setiap $C_j = S_j(B_j)$, $1 \le j \le 8$.
   S-Box ini memiliki sifat – sifat sebagai berikut:[2]
   1. Setiap baris pada setiap *S-Box* adalah permutasi dari bilangan bulat 0, . . . , 15.
   2. Tidak ada *S-Box* yang merupakan fungsi linear atau fungsi *affine* dari *input*nya.
   3. Dengan merubah sebuah bit pada *input S-Box*, kita dapat mengubah setidaknya 2 buah bit pada *output*.
   4. Untuk setiap *S-Box* dan setiap *input* x, *S*(*x*) dan *S*(*x* ⊕ 001100) berbeda pada setidaknya 2 bit (x adalah sebuah *bitstring* dengan panjang 6 bit).

d. *Bitstring* $C = C_1C_2C_3C_4C_5C_6C_7C_8$ dengan panjang 32 bit kemudian dipermutasi dengan fungsi permutasi P yang telah ditentukan sebelumnya. Hasilnya adalah *bitstring* P(*C*) yaitu *f*(*A, J*).

Berikut ini adalah ilustrasi dari keempat langkah di atas:



Gambar 2. Ilustrasi Fungsi f pada DES

Secara singkat, fungsi *f* mengandung substitusi (dengan menggunakan *S-Box*) diikuti oleh permutasi P.

Terdapat satu lagi variabel yang belum diketahui asalnya, yaitu *key schedule* yang dihasilkan dari komputasi dengan kunci K. Sebenarnya, K adalah *bitstring* dengan panjang 64, di mana 56 bit berupa kunci yang digunakan dan 8 bit berupa bit *parity-check*, 8 bit ini akan digunakan untuk mendeteksi apakah terdapat kesalahan pada kunci yang digunakan karena bisa saja kunci tersebut mengalami perubahan selama proses pengiriman data. Ke 8 bit ini diletakkan pada posisi 8, 16, . . . , 64. Bit *parity check* ini akan membuat agar setiap 8 bit (1 byte) memiliki jumlah 1 yang ganjil. Bit *parity check* diabaikan pada komputasi *key schedule*.

Berikut ini adalah cara mendapatkan *key schedule*:

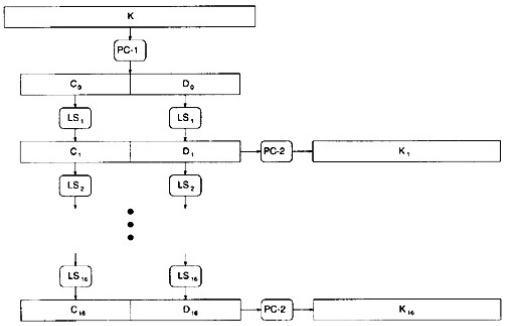
a. Dengan kunci K sepanjang 64 bit, buang bit *parity check* sehingga hanya tersisa 56 bit. Permutasikan bit tersebut dengan fungsi permutasi yang telah ditentukan sebelumnya yaitu PC^{-1}. Kita dapat menuliskan PC^{-1} $(K) = C_0D_0$, di mana C_0 menunjukkan 28 bit pertama dari $PC^{-1}(K)$ dan D_0 menunjukkan 28 bit sisanya.
b. Iterasikan langkah berikut dari i = 1 sampai dengan i = 16

$$C_i = LS_i(C_{i-1})$$

$$D_i = LS_i(D_{i-1}),$$

LS_i merepresentasikan *cyclic shift* (penggeseran bit ke kiri) sebanyak 1 atau 2 bit, tergantung pada nilai dari i. Geser 1 posisi bila nilai i = 1, 2, 9 or 16, dan geser 2 posisi untuk nilai i selain di atas.

Setelah itu kita mendapatkan K_i = PC-2(C_iD_i), dengan PC-2 adalah sebuah fungsi permutasi lain yang telah ditentukan juga sebelumnya. Ilustrasinya adalah sebagai berikut:



Gambar 3. Ilustrasi Cara Memperoleh *Key Schedule*

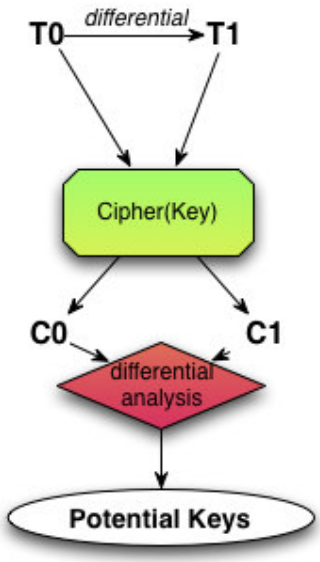
3. Teknik *Differential Cryptanalysis*

Salah satu serangan yang paling terkenal pada DES adalah metode "*Differential Cryptanalysis*" yang dikenalkan oleh Biham dan Shamir[5]. Serangan ini adalah serangan *chosen plaintext*, yaitu penyerang memiliki kemampuan untuk memilih *plaintext* tertentu dan mendapatkan *ciphertext* yang berkesusaian. Serangan ini mungkin tidak efektif untuk memecahkan DES 16 ronde seperti pada umumnya, tetapi serangan ini dapat memecahkan DES dengan iterasi lebih rendah. Sebagai contoh, DES 8 ronde dapat dipecahkan hanya dalam beberapa menit dengan menggunakan sebuah PC sederhana.

Pada DES, umumnya kerahasiaannya terletak pada kunci yang digunakan, sementara tabel permutasi dan tabel substitusi yang digunakan tidak berubah. Karena itulah kita mengasumsikan kriptanalis sudah mengetahui tabel permutasi dan tabel substitusi yang digunakan.

Differential Cryptanalysis adalah suatu teknik di mana kita membuat perubahan tertentu pada *plaintext* sehingga dari *ciphertext* yang dihasilkan, kita bisa mencari kunci yang digunakan.[1]

Konsep dari *Differential Cryptanalysis* adalah membandingkan *exclusive-or* dari 2 buah *plaintext*s dengan *exclusive-or* dari 2 *ciphertext*s yang bersangkutan. Kedua *plaintext* tersebut kita sebut L_0R_0 dan $L_0^*R_0^*$ dengan hasil *x-or* nya adalah $L_0`R_0` = L_0R_0 \oplus L_0^*R_0^*$. Tanda petik (') artinya adalah hasil dari operator *x-or.* membuat untuk membandingkan ciphertext yang didapat



Gambar 4. Ilustrasi Cara Kerja Differential Cryptanalysis

Bila S_j adalah suatu *S-Box* ($1 \le j \le 8$), dan (B_j, B_j^*) adalah sepasang bitstring dengan panjang 6 bit, maka kita mengatakan *input x-or* dari S_j adalah $B_j \oplus B_j^*$ dan *output x-or* dari S_j adalah $S_j(B_j^*) \oplus S_j(B_j^*)$. *Input* dari S_j adalah bitstring dengan panjang 6 bit, sedangkan *output*nya adalah bitstring sepanjang 4 bit.

Untuk setiap $B_j` \in (Z_2)^6$, kita definisikan himpunan $\Delta(B_j`)$ yang terdiri dari pasangan (B_j, B_j^*). Di mana $\Delta(B_j`) = \{(B_j, B_j \oplus B_j`): B_j` \in (Z_2)^6\}$. Maka himpunan $\Delta(B_j`)$ akan terdiri dari $2^6 = 64$ elemen.

Untuk setiap pasangan pada $\Delta(B_j`)$, kita dapat menghitung *output* dari S_j. Terdapat 64 *output* yang ada, yang didistribusikan ke $2^4 = 16$ nilai yang berbeda. Distribusi yang tidak merata inilah yang akan menjadi dasar dari serangan ini.
Contohnya adalah pada *input x-or* 110100. Maka
$\Delta(110100) = \{(000000,110100), (000001,110101), ..., (111111,001011)\}$

Untuk setiap pasangan pada himpunan $\Delta(110100)$, kita menghitung *output x-or* dari S_1. Sebagai contoh $S_1(000000) = E_{16} = 1110$ dan $S_1(110100) = 9_{16} = 1001$, sehingga *output x-or* dari pasangan (000000, 110100) adalah 0111.
Bila hal ini dilakukan untuk 64 pasangan tersebut, maka kita akan mendapatkan distribusi berikut

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12

1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

Bisa kita lihat bahwa hanya 8 nilai yang muncul dari ke 16 nilai yang ada. Pada umumnya, hanya terdapat 75 - 80% nilai yang muncul.[5]

Untuk $1 \le j \le 8$, dan untuk bitstring $B_j`$dengan panjang 6 bit dan bitstring $C_j`$ dengan panjang 4 bit, kita definisikan

$IN_j (B_j`, C_j`) =$
$\{B_j \in (Z_2)^6 : S_j(B_j) \oplus S_j(B_j \oplus B_j`) = C_j`\}$

Dan
$N_j (B_j`, C_j`) = | IN_j (B_j`, C_j`) |$

Nj (Bj`, Cj`) menghitung jumlah pasangan dengan *input x-or* Bj` dengan *output x-or* $C_j`$ untuk *S-Box* S_j. Sedangkan nilai – nilai pasangan yang membentuk Bj` dapat dilihat pada himpunan $IN_j(B_j`, C_j`)$.
Contoh tabel distribusi di atas adalah tabel yang berisi nilai $N_j (110100, C_j`)$, $C_j` \in (Z_2)^4$.

Sedangkan contoh himpunan IN_j (110100, $C_j`$) adalah sebagai berikut:

Keluaran *X-or*	Masukan yang mungkin
0000	
0001	000011, 001111, 011110, 011111, 101010, 101011, 110111, 111011
0010	000100, 000101, 001110, 010001, 010010, 010100, 011010, 011011, 100000, 100101, 010110, 101110, 101111, 110000, 110001, 111010
0011	000001, 000010, 010101, 100001, 110101, 110110
0100	010011, 100111
0101	
0110	
0111	000000, 001000, 001101, 010111, 011000, 011101, 100011, 101001, 101100, 110100, 111001, 111100
1000	001001, 001100, 011001, 101101, 111000, 111101
1001	
1010	
1011	
1100	
1101	000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010
1110	
1111	000111, 001010, 001011, 110011, 111110, 111111

Kita juga dapat menghitung *input x-or* dari semua *S-Box* dengan cara berikut:

$B \oplus B^* = (E \oplus J) \oplus (E^* \oplus J)$
$= E \oplus E^*$

Hal ini berarti *input x-or* untuk *S-Box* tidak tergantung pada nilai *J*. Tetapi *output x-or* nya tetap tergantung dari nilai *J*.

Kita dapat juga menulis *B*, *E* dan *J* sebagai konkatenasi dari 8 *bitstring* dengan panjang 6-bit seperti berikut :

$B = B_1\ B_2\ B_3\ B_4\ B_5\ B_6\ B_7\ B_8$
$E = E_1\ E_2\ E_3\ E_4\ E_5\ E_6\ E_7\ E_8$
$J = J_1\ J_2\ J_3\ J_4\ J_5\ J_6\ J_7\ J_8$

Apabila kita mengetahui nilai E_j dan E_j^* untuk beberapa *j*, $1 \le j \le 8$, dan nilai dari *output x-or* untuk S_j, $C_j` = S_j(B_j) \oplus S_j(B_j^*)$. Maka kita dapat mengetahui bahwa $E_j \oplus J_j \in IN_j(E_j`, C_j`)$. Di mana $E_j` = E_j \oplus E_j^*$.

Berikutnya, kita definisikan himpunan
$Test_j (E_j, E_j^*, C_j`) = \{B_j \oplus E_j : B_j \in IN_j (E_j`, C_j`) \}$
Di mana $E_j` = E_j \oplus E_j^*$

Definisi di atas menunjukkan bahwa himpunan *Test* mengambil *x-or* dari E_j dengan setiap elemen dari himpunan ($E_j`$, $C_j`$). Himpunan *test* ini penting karena *bitstring* dari kunci yang kita cari, yaitu J_j terdapat di dalamnya.
Berikut ini adalah sebuah contoh dari teori di atas. Misalnya diketahui $E_1 = 000001$, $E_1^* = 110101$ dan $C_1` = 1101$. Karena N_1 (110100, 1101) = 8, maka ada tepat 8 *bitstring*s pada himpunan $test_1$ (000001, 110101, 1101). Dari tabel di atas, dapat kita lihat bahwa IN_1 (110100, 1101) = {000110, 010000, 010110, 011100, 100010, 100100, 101000, 110010}

Dan apabila kita meng-*x-or* kannya dengan E1, kita dapatkan
$Test_1$ (000001, 110101, 1101) = {000111, 010001, 010111, 011101, 100011, 100101, 101001, 110011}
Apabila kita mempunyai data E_1, E_1^*, dan $C_1`$ yang lain, maka kita bisa membuat himpunan *test* yang lain, kemudian kita bisa mengambil irisannya. Kita lakukan ini berulang – ulang, hingga akhirnya elemen pada himpunan ini hanya tersisa 1, dan itulah nilai dari J_1.

4. Penerapan Teknik *Differential Cryptanalysis* Pada DES 3 ronde

Misalkan penyerang memilih 2 buah *plaintext* yang diinginkan dan dinotasikan dengan L_0R_0, $L_0^*R_0^*$, algoritma DES kemudian akan menghasilkan *ciphertext* L_3R_3, $L_3^*R_3^*$. Kita dapat menyatakan R_3 menjadi:

$R_3 = L_2 \oplus f(R_2,K_3)$
$= R_1 \oplus f(R_2,K_3)$
$= L_0 \oplus f(R_0, K_1) \oplus f(R_2,K_3)$

Begitu juga dengan R_3^*, sehinga kita bisa menyatakan $R_3`$ dalam bentuk
$R_3` = R_3 \oplus R_3^*$
$= L_0 \oplus f(R_0, K_1) \oplus f(R_2,K_3) \oplus L_0^* \oplus f(R_0^*, K_1) \oplus f(R_2^*,K_3)$
$= L_0` \oplus f(R_0, K_1) \oplus f(R_2,K_3) \oplus f(R_0^*, K_1) \oplus f(R_2^*,K_3)$ .....(1)

Andaikan penyerang memilih *plaintext*nya sedemikin rupa sehingga $R_0 = R_0^*$ dan karenanya $R_0` = 000....0$, maka $f(R_0, K_1) = f(R_0^*, K_1)$. Hal ini akan membuat persamaan (1) di atas menjadi
$R_3` = L_0` \oplus f(R_0, K_1) \oplus f(R_2,K_3) \oplus f(R_0^*, K_1) \oplus f(R_2^*,K_3)$
$= L_0` \oplus f(R_2,K_3) \oplus f(R_2^*,K_3)$ .....(2)
Hal ini disebabkan karena $f(R_0, K_1) \oplus f(R_0^*, K_1) = 0000..0$

Sementara itu, kita mengetahui nilai dari $R_3`$ dan $L_0`$ karena serangan ini adalah chosen *plaintext* dan juga karena kita mengetahui fungsi permutasi awal dan invers permutasinya, hanya kuncinya saja yang belum diketahui. Hal ini berarti kita bisa menulis bentuk di atas menjadi:

$f(R_2,K_3) \oplus f(R_2^*,K_3) = R_3` \oplus L_0`$ .....(3)

Apabila kita melambangkan keluaran dari *S-Box* pada ronde ketiga sebagai C, maka persamaan di atas menjadi:

$P(C) \oplus P(C^*) = R_3` \oplus L_0`$ .....(4)
Dan karenanya
$C` = C \oplus C^* = P^{-1}(R_3` \oplus L_0`)$ .....(5)

Selain itu, karena kita juga mengetahui nilai dari L_3 dan L_3^*, dan karena $R_2 = R_2^*$, maka kita bisa mengetahui *input* dari *S-Box* pada ronde ketiga. Kita nyatakan *input* tersebut sebagai *E,* di mana:

$E = E(L_3)$ .....(6)
Dan
$E^* = E(L_3^*)$ .....(7)

Sekarang kita dapat membuat himpunan *test* seperti yang dijelaskan di bagian sebelumnya mengenai teknik *Differential Cryptanalysis* sebagai *test*(E,E^*,C`). Karena terdapat 8 buah *S-Box*, maka kita harus mengulangi langkah ini sebanyak 8 kali.

Algoritma lengkap dari teknik ini adalah:
Input : L_0R_0, $L_0^*R_0^*$, L_3R_3, dan $L_3^*R_3^*$ di mana $R_0 = R_0^*$

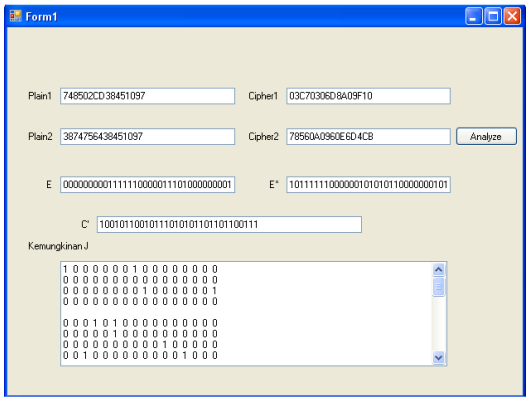
- Hitung $C' = P^{-1}(R_3` \oplus L_0`)$
- Hitung $E = E(L_3)$ dan $E^* = E(L_3^*)$
- For i=1 to 8 do
  Hitung $test_j (E_j,E_j^*,C_j`)$

5. **Contoh Kasus Penerapan Teknik *Differential Cryptanalysis* Pada DES 3 Ronde**

Pada bagian ini, kita akan mencoba menggunakan teknik yang dijelaskan pada bagian di atas untuk memecahkan suatu DES 3 ronde. Misalkan kita memiliki data sebagai berikut

Plaintext	*Ciphertext*
748502CD38451097	03C70306D8A09F10
3874756438451097	78560A0960E6D4CB
486911026ACDFF31	45FA285BE5ADC730
375BD31F6ACDFF31	134F7915AC253457
357418DA013FEC86	D8A31B2F28BBC5CF
12549847013FEC86	0F317AC2B23CB944

Data di atas adalah 3 buah pasangan *plaintext-ciphertext* yang dihasilkan. Kita akan mencoba mendapatkan kunci yang diinginkan dari 3 buah pasangan data tersebut. Untuk tujuan ini, penulis telah membuat sebuah program sederhana yang dapat membantu proses analisis dengan prinsip yang telah dijelaskan di bagian 4. Tabel untuk *Initial Permutation* dan *inverse*nya, serta tabel *S-Box* diambil dari [5]. Program ini juga dapat diunduh di "students.if.itb.ac.id/~if15122/kripto"



Gambar 5. Program Sederhana Untuk Mencari Kunci DES 3 ronde

Pada gambar di atas, pada *textbox* paling bawah, terdapat *counter* yang menunjukkan kemunculan nilai pada himpunan *test*$_j$ $(E_j,E_j^*,C_j`)$ yang menyatakan kemungkinan nilai J_i. Apabila dijalankan untuk semua kemungkinan pasangan nilai, dan kemudian *counter* tersebut semuanya ditambahkan, maka kita akan mendapatkan hasil berikut:

Untuk J_1

1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Untuk J_2

0	0	0	1	0	3	0	0	1	0	0	1	0	0	0	0
0	1	0	0	0	2	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1	0	1	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0	1	0	2	0	0	0

Untuk J_3

0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	1
0	2	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

Untuk J_4

3	1	0	0	0	0	0	0	0	0	2	2	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	1	0	1	1
1	1	1	0	1	0	0	0	0	1	1	1	0	0	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	2	1

Untuk J_5

0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	1	0	0	0	0	2	0

Untuk J_6

1	0	0	1	1	0	0	3	0	0	0	0	1	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0

Untuk J_7

0	0	2	1	0	1	0	3	0	0	0	1	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	2	0	0	0	2	0	0	0	0	1	2	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1

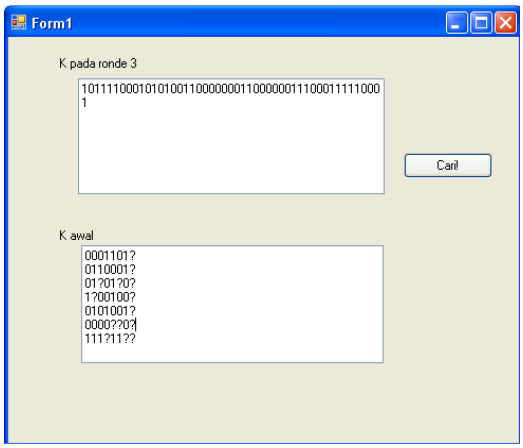
Untuk J_8

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1
0	3	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bila kita ambil counter yang tertinggi, yaitu 3 (berarti di setiap himpunan test dari setiap pasangan data terdapat nilai tersebut) dari setiap J_1, J_2, .. J_8 maka kita akan mendapatkan

J1	101111
J2	000101
J3	010011
J4	000000
J5	011000
J6	000111
J7	000111
J8	110001

Kunci tersebut adalah kemungkinan kunci pada ronde ketiga. Untuk mendapatkan kunci pada ronde pertama, kita harus melakukan *inverse* dari fungsi *Cyclic shift* dan permutasi key, kemudian menambahkan bit *parity check*. Untuk menghitung kunci awal ini, penulis telah membuat sebuah program sederhana juga. Program ini juga dapat diunduh di "students.if.itb.ac.id/~if15122/kripto" Hasilnya terlihat seperti gambar berikut:



Gambar 6. Program Sederhana Untuk Mencari Kunci awal dari Kunci Ke-n

Kunci yang didapat adalah:
0001101?
0110001?
01?01?0?
1?00100?
0101001?
0000??0?
111?11??

Terlihat ada tanda tanya "?" pada Kolom Kunci yang didapat, hal ini disebabkan karena pada tabel permutasi kunci, terdapat beberapa bit yang dihilangkan, sehingga ukuran kunci dari 56 bit menjadi 48 bit. Hal ini menyebabkan ada 8 bit yang hilang. Selain itu, terdapat 8 bit lagi yang belum diketahui karena merupakan bit *parity check*, tetapi bit ini dapat diketahui bila bit – bit lainnya telah lengkap. Setelah melalui tahap ini, maka kita hanya perlu melakukan *exhaustive search* sebanyak 2^8 = 256 kali

6. Serangan Pada DES dengan Jumlah Ronde yang Lebih Banyak

Pada dasarnya, teknik yang digunakan adalah sama. Misalnya pada DES 6 ronde, dasarnya adalah sebagai berikut:

$R_6 = L_5 \oplus f(R_5, K_6)$
$= R_4 \oplus f(R_5, K_6)$
$= L_3 \oplus f(R_3, K_4) \oplus f(R_5, K_6)$

Bisa kita lihat bahwa hal ini mirip sekali dengan penerapan pada DES 3 ronde, hanya saja pada kasus ini, kita tidak mengetahui nilai dari L_3 dan R_3. Untuk itu kita harus mengikutsertakan kemungkinan dari suatu *plaintext* L_0R_0 dapat menghasilkan L_3R_3 yang diharapkan. Ilustrasinya adalah sebagai berikut:

Misalkan L_0 = 00000....0 (heksadesimal)
R_0 = 60000000
Maka L_1 = 60000000
R_1 = 00808200

Dengan kemungkinan p =14/64, kemungkinan ini didapat dari langkah berikut:

- Expand R_0 menjadi 001100.....0
- *Input* pada *S-Box* pertama adalah 001100
- *Input* pada 7 *S-Box* lainnya adalah 000000
- Akibatnya *output* pada 7 *S-Box* lainnya pastilah 0000
- Sedangkan *output* dari *S-Box* yang pertama bisa bermacam – macam, dan kemungkinan *output*nya 1110 adalah 14/64, N(001100, 1110) = 14 dan ada 64 kemungkinan

Kemungkinan - kemungkinan dari setiap ronde tersebut kemudian kita kalikan, misalnya p pada ronder 1= ¼, ronde 2 = 1, ronde 3 = ¼, maka kemungkinan L_3R_3 sesuai dengan dugaan kita adalah 1/16. Karena itu pada 15/16 dari percobaan kita akan menemukan data percobaan yang salah. Namun demikian, kita bisa menaikkan kemungkinan keberhasilan kita dengan menggunakan prinsip berikut:

Apabila jumlah dari suatu himpunan *test*, yaitu $|test_j|$ adalah 0, maka data percobaan yang dipakai sudah pasti salah.

Berdasarkan penelitian, jumlah chosen *plaintext* yang dibutuhkan untuk memecahkan DES pada berbagai ronde adalah:[5]

Jumlah Ronde	Jumlah chosen *plaintext*
8	2^{14}

10	2^{24}
12	2^{31}
14	2^{39}
16	2^{47}

Sedangkan kompleksitas algoritmanya adalah sebagai berikut:[2]

Jumlah Ronde	Kompleksitas
4	2^{4}
6	2^{8}
8	2^{16}
9	2^{26}
10	2^{35}
11	2^{36}
12	2^{43}
13	2^{44}
14	2^{51}
15	2^{52}
16	2^{58}

7. Kesimpulan

Meskipun DES merupakan algoritma yang sudah banyak digunakan, ternyata algoritma ini dianggap belum memiliki tingkat keamanan yang cukup. Karena itulah, untuk meningkatkan keamanannya dilakukan beberapa cara. Cara yang pertama adalah mengubah susunan *S-Box* nya sedemikian rupa, sehingga distribusinya lebih merata. Sedangkan cara yang kedua adalah dengan menggunakan teknik DES yang diulang seperti *Double* DES dan *Triple* DES.

Teknik *Differential Cryptanalysis* merupakan sebuah teknik yang sangat banyak digunakan untuk memecahkan berbagai algoritma enkripsi blok berbasiskan permutasi dan substitusi. Beberapa algoritma enkripsi blok lain yang juga lemah terhadap serangan ini misalnya adalah algoritma FEAL, REDOC-II, and LOKI.

Meskipun demikian, apabila jumlah ronde pada DES dinaikkan, maka teknik ini juga akan membutuhkan waktu yang lama untuk memecahkannya. Untuk jumlah ronde di atas 10, maka teknik ini sudah tidak terlalu ampuh lagi, meskipun masih lebih baik daripada teknik *brute force*.

8. Daftar Pustaka

[1] Chu-Carroll, Mark C.(2008). Differential Cryptanalysis.http://scienceblogs.com/goodmath/2008/10/differential_cryptanalysis.php. Tanggal akses : 30 Maret 2009 pukul 15.25 WIB.

[2] Heys, Howard M. A Tutorial on Linear and Differential Cryptanalysis. Electrical and Computer Engineering, Memorial University of Newfoundland, Canada.

[3] Munir, Rinaldi. (2006). Diktat Kuliah IF5054 Kriptografi. STEI, Bandung.

[4] Savitri, Dian Intania. (2006). Analisis Keamanan Algoritma Kriptografi DES, Double DES, dan Triple DES. Informatika ITB, Bandung.

[5] Stinson, Douglas. (1991). Cryptography: Theory and Practice. CRC Press LLC.