Studi Linear Kriptanalis ,Differential Kriptanalis, dan DESCHALL *effort* dalam usaha memecahkan DES

Aditya Nurcholis Hakim – NIM : 13503107

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if13107@students.if.itb.ac.id

Abstrak
Data Encryption Standard telah digunakan pada sejak puluhan tahun lalu. Penggunaannya yang luas dan pendeknya jumlah kunci (bila dibandingkan dengan saat ini) menjadikannya target yang 'empuk' untuk diserang. Tulisan ini akan membahas bagaimana memecahkan pesan yang dienkripsi dengan DES dengan menggunakan brute force; bagaimana kekuatan untuk melakukan hal ini dapat dilakukan oleh individu atau suatu organisasi kecil dengan menggunakan sedikit atau tanpa pendanaan, serta akan membahas teknik dasar kriptanalisis (linear dan diferential kriptanalis) yang nantinya dapat digunakan untuk menyerang pesan yang dienkripsi dengan DES.
Pada akhirnya penulis menyarankan untuk mengganti sistem enkripsi yang berbasis DES dengan sistem yang mengunakan kunci yang lebih panjang.

Kata Kunci : Data Encryption Standard(DES), linear kriptanalis, diferential kriptanalis, DESCHALL effort.

Pendahuluan

Pada 28 January 1997, RSA Laboratories meluncurkan berbagai tantangan kriptografi. Tujuannya adalah menemukan pesan rahasia yang telah dienkripsi dengan panjang kunci tertentu. Salah satu yang paling menantang adalah tantangan yang berbasis DES, algoritma dengan menggunakan panjang kunci 56 bit. Setelah dua tantangan yang lebih mudah dipecahkan, perhatian menuju pada tantangan DES.

Dipimpin oleh Rocke Verser, Matt Curtin, dan Justin Dolske, Deschall Effort berusaha untuk memecahkan tantangan DES dengan skala besar menggunakan proyek komputasi terdistribusi melalui internet. Dengan mencoba setiap kemungkinan dari 2^{56} kunci yang digunakan untuk mengenkripsi pesan rahasia--*brute force attack*. Cara *brute force* seperi ini secara alami cocok dengan usaha komputasi paralel atau dengan komputasi terdistribusi, karena hal ini terdiri dari masalah dengan jumlah besar yang independensi satu sama lainnya--percobaan dari setiap kunci.

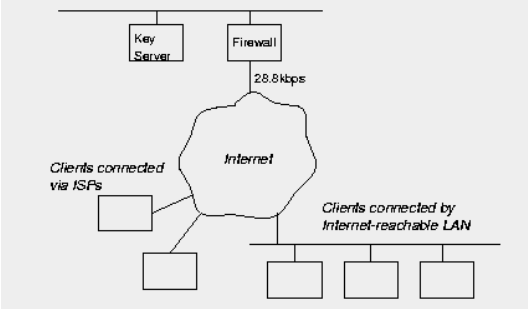
Meskipun bukanlah suatu teknik baru, pencarian kunci dengan *brute force* merupakan suatu parameter ketika suatu kriptosistem diluncurkan. Jika terdapat suatu algoritma yang dipercaya 'aman', tingkat keamanan algortima ini biasa dinilai dengan usaha yang dibutuhkan dengan cara *brute force*.

Meskipun banyak pihak yang percaya bahwa agen intelejen pemerintahan mempunyai teknologi dan sumber daya untuk melakukan *brute force* untuk menyerang DES, tidak ada seorang pun yang telah menyelesaikan usaha ini di publik sebelum proyek ini berhasil. DES tetap digunakan pada berbagai macam hal termasuk masalah finansial. Oleh karena itu DES merupakan sebuah target nyata dan karena ia menggunakan kunci yang relatif pendek (bila dibandingkan dengan saat ini), penyerangan pada DES merupakan suatu hal yang menarik.

1. DESCHALL effort

1.1 Arsitektur

Pendekatannya ialah dengan mengelilingi suatu "key server' yang menyimpan block kunci mana yang telah dicoba. Client akan mengontak server melalui internet untuk meminta pekerjaan dan kemudian melaporkan hasilnya. Arsitektur ini tampak pada gambar di bawah.



Arsitektur Deschall

1.2 Protokol

Semua protokol antara client dan server dilakukan melalui protocol UDP, bagian standar dari stack IP. UDP adalah low overhead , conectionless protocol yang sudah cukup untuk memenuhi kebutuhan. Protokol yang digunakan ialah modifikasi hasil desain oleh Germano Caronni pada tantangan RSA's RC5-32/12/6. Protokol ini terdiri dari beberapa pesan simpel :

``Initial" request
Menyediakan server untuk client dengan type dan versi tertentu dan meminta initial blok yang akan di periksa.
``Not found" request
Melaporkan suatu range keys yang telah diperiksa bahwa tidak ditemukan dan meminta blok kunci yang baru untuk diperiksa.
``Answer" reply
Dikirim oleh server untuk menjawab permintaan client untuk pekerjaan lanjutan (melalui kedua request di atas).
``Message" reply
Dapat dikirim oleh server untuk menampilkan pesan pada klien informasi – informasi yang penting.
``Kill" reply
Dapat dikirim oleh server untuk menghentikan kerja client.

1.3 Server dan Client

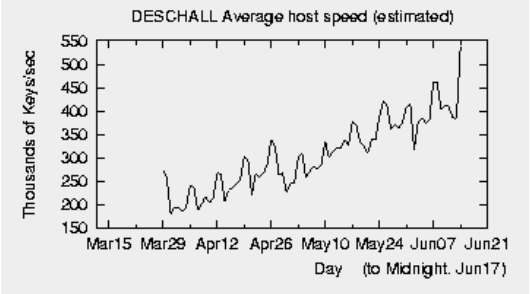
Untuk tantangan ini keyserver yang digunakan ialah IBM PS/2 server (arsitektur 486 yang relatif lambat) dengan 56 MB RAM, terhubung ke internet dengan menggunakan koneksi PPP 28.8 kbps. Server ini dengan mudah mampu menangani load dari hampir 10.000 client meskipun sebuah server backup pentium terkadang digunakan pada periode high load yang tidak umum.

Client yang menggunakan protokol ini didesain untuk menjalankan berbagai macam sistem. Pada akhir kontes didapatkan 40 jenis client yang tersedia untuk berbagai jenis kombinasi perangkat keras dan sistem operasi. Semua client yang berjalan pada Intel (atau yang kompatibel) dan perangkat keras Macintosh PowerPC yang terdiri dari hand-optimized assembly code, sisanya dilakukan dengan menggunakan C. Java sempat dipertimbangkan namun pada akhirnya tidak jadi digunakan karena kecepatan pada Java pada umumnya tidak dapat ditolerir.

Client dengan dipotimasi untuk mendeskripsi pesan dengan berbagai cara untuk mengoptimasi proses DES dan mampu mendeteksi sedini mungkin *non-winning key*. Dengan metode ini, 200MHz Pentium sistem mampu untuk mencoba hampir 1 juta kunci/detik, dan 250MHz PowerPC 604 e based sistem mencapai 1,5 juta kunci/detik. Pada akhir kontes, akan diperkenalkan “bitslice” client yang diinspirasi oleh Biham yang mana menggunakan sistem 64 bit.

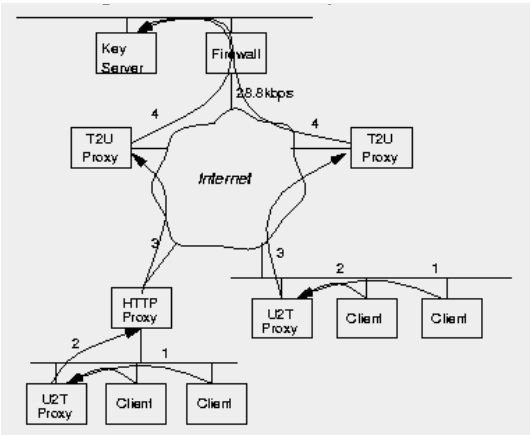
Dengan menggunakan, 500MHz alpha dapat mencoba 5,3 juta kunci/detik, dan 167MHz UltraSPARC dapat mencoba 2,4 juta kunci/detik. Pada akhirnya, Intel-*compatible system* mencari 53,8% dari jumlah seluruh kunci, SPARC based system 21,3%, PowerPC *system* 8,1% dan lainnya mencari sisanya yakni 16,8%.

Seluruh client secara default melakukannya dengan *low priority*, sehingga hanya pada kondisi *idle* saja mereka melakukan komputasi sehingga tidak menggangu aktifitas normal dari komputer client. Efek samping yang menarik dari pendekatan “*only use idle cycles*”, ialah pada saat akhir pekan menampilkan peak yang signifikan, dimana biasanya terjadi status *idle* terjadi. Penigkatan kinerja pada client pun merangakak naik secara teratur.



1.4Gateways and Proxies

Tak lama setelah Deschall mulai terkenal, kemudian ditemukan *firewall* di beberapa situs yang akan mem-*block* UDP messages yang client dan server saling pertukarkan. Untuk mengatasi masalah tersbut, mereka membangun sepasang gateway atau *proxies* yang akan men-*tunnel* UDP messages melalui koneksi TCP, seperti ilistrasi di bawah.



Salah satu dari proxy ini akan berada pada jaringan user, dan pasangannya akan dijaga oleh Deschall organizers. Client yang bekerja dibelakang *firewall* akan menggunakan gateway U2T sebagai keyserver. Gateway U2T akan menerima datagram client dan mengirim data tersebut melalui koneksi TCP menuju gateway T2U. Data tersebut juga di format menjadi suatu HTTP request, untuk melewati *firewall* yang mem-*block* koneksi TCP yang tidak jelas tetapi melewatkan *web acces*. Untuk situs dengan *aplication-layer firewall*, gateway client U2T dapat menggunakan proxy webnya yang dapat memforward request ke gateway T2U. Dengan metode tersbutm gateway T2U kemudian meng-*convert* data yang diterima kembali menjadi UDP datagram dan kemudian mengirim ke keyserver.

Gateway Deschall memperbolehkan banyak orang untuk berpartispasi. Sebagai contoh, kontribusi keseluruhan Sun Microsystems terhubung melalui gateway.

Untuk melakukan komputasi yang sangat besar dapat dilakukan tanpa bantuan expensive dedicated hardware atau superkomputer namun kekuatan tersebut didapat dari *massive internet computing.*

2. Linear Attack

2.1 Gambaran umum

Linear cryptanalysis mencoba menggunakan kelebihan dari probabilitas kemunculan dari ekspresi linear dari bit plainteks yang ada, bit chiperteks, dan subkeybits. *a known plaintext attack*: ialah penyerangan yang berdasar pada informasi yang dimiliki oleh sang kriptanalis terhadap palainteks yang diketahui beserta korespondensinya pada chiperteks. Bagaimanapun juga, kriptanalis tidak tau plainteks (maupun korespondensi chiperteksnya) mana yang tersedia. Pada banyak aplikasi dan skenario logis jika kita berasumsi bahwa kriptanalis mengetahui suatu random set plainteks beserta korespondensinya pada chiperteks.

Ide dasar dari cara ini ialah dengan menggunakan pendekatan operasi dari suatu chiper dengan menggunkana suatu ekspresi linear yang yang linearitasnya mengacu pada modulo 2 operasi bit (contoh : expresi eksklusif or dilambangkan dengan $\oplus$). Contoh dari ekspresi ini ialah :

$X_{i1} \oplus X_{i2} \oplus ... \oplus X_{i3} \oplus Y_{j1} \oplus Y_{j2} \oplus ... \oplus Y_{j3} = 0$ (1)

Dimana Xi adalah bit ke i dari input X = [X1, X2, ..] dan Yj adalah bit kej dari output Y = [Y1, Y2, ..]. Persamaan ini menggambarkan “penjumlahan ” eksklusif-or dari input u input bits dan v output bit.

Pendekatan yang dilakukan pada lienar kriptanalis ialah dengan menentukan bentuk ekspresi di atas yang mempunyai tingkat kemunculan yang rendah atau yang tinggi. Jika suatu chiper menunjukan tendensi bahwa

persamaan satu untuk mempertahankan nilai probabilitas yang tinggi bukan untuk mempertahankan nilai probabilitas yang rendah maka hal ini menandakan bahwa chiper menunjukan kelemahan dari kemampuan pengacakan. Misalkan kita memilih secara random nilai u + v bit dan menempatkannya pada persamaan di atas, probabilitas dari kebenaran ekspresi itu hanya tepat ½. Deviasi atau bias dari probabilitas itulah yang dieksploitasi pada linear kriptanalisis. Selanjutnya, kita akan menggunakan probabilitas berdeviasi dari ½ sebagai deviasi dari linear probabilitas. Kemudian jika ekpresi diatas menggunkan probabilitas pL sebagai plaintkes yang dipilih secara acak maka bias probabilitas nya adalah Pl – ½. Makin tinggi nilai dari probabilitas bias |pL – ½|, makin baik penggunaan linear kriptanalisis dengan lebih sedikit *known plainteks* yang dibutuhkan pada penyerangan.

Ada beberapa cara untuk melakukan penyerangan dengan linear kriptanalisis. Makalah ini akan memfokuskan pada apa yang disebut oleh Matsui sebagai Algorithm 2 [1]. Kita bangun aproksimasi lienar pada bit plainteks sebagai X pada persaman (1) dan input pada putaran terakhir pada chiper sebagai Y. Bit plainteks yang digunakan adalah random maka konsekuensinya adalah input pada putaran terakhir juga random.

Persamaan (1) dapat disetarakan sehingga sisi kanan adalah jumlah dari jumlah bit *subkey*. Pada persamaan (1) tertulis “0” pada sisi kanannya, persamaan ini secara implisit menyatakan mempunyai subkey. Bit – bit ini ialah fix namun tidak diketahui (yang akan ditentukan pada saat penyerangan). Jika jumlah dari *subkey* yang terlibat adalah “0” maka bias dari persamaan (1) mempunyai tanda yang sama dengan bias dari ekspresi yang digunakan pada penjumlahan *subkey*, sebaliknya jika jumlah dari *subkey* yang terlibat adalah “1” maka bias dari persaaman (1) akan memiliki tanda yan berlainan.

Perhatikan bahwa jika pL =1 maka ekspresi linear merupakan representasi yang tepat sama dari proses chiper.
Pertanyaan yang muncul kemudian bagaimana kita membuat persamaan yang memiliki tingkat linear yang tinggi dan selanjutnya dapat kita eksploitasi? Hal ini dilakukan dengan cara mempertimbangkan komponen nonlinear : S-Box. Ketika properti nonlinear dari S-Box dienumerasi maka memnugkinkan kita untuk membangun aproksimasi linear antara bit-bit input denan bit-bit output pada S-Box.

2.2 Prinsip *pilling up*

Sebelum kita membangun sebuah ekspresi linear sebagai contoh dari makalah ini, kita membutuhkan beberapa kakas bantu dasar. Ambil dua random variabel binary, X_1 dan X_2. Kita mulai dengan mengambil persamaan : $X_1 \oplus X_2 = 0$ dan ekivalen dengan $X_1 = X_2$; $X_1 \oplus X_2 = 1$ adalah ekspresi affinenya dan ekivalen dengan $X_1 \neq X_2$.

Lalu, kita asumsikan probabiltas distribusinya sebagai berikut :

$$\Pr(X_1=i) = p_1, \quad \text{untuk } i=0;$$
$$= p_1 - 1 \quad \text{untuk } i=1;$$

dan

$$\Pr(X_2=i) = p_2, \quad \text{untuk } i=0;$$
$$= p_2 - 1 \quad \text{untuk } i=1;$$

jika kedua random variable saling independent, maka

$$\Pr(X_1=i, X_2=j) = p_1p_2 \quad \text{untuk } i=0, j=0;$$
$$= p_1(1-p_2) \quad \text{untuk } i=0, j=1;$$
$$= (1-p_1)p_2 \quad \text{untuk } i=1, j=0;$$
$$= (1-p_1)(1-p_2) \quad \text{untuk } i=1\ j=1;$$

dan dapat ditunjukkan bahwa

$$\Pr(X_1 \oplus X_2 = 0)$$
$$= \Pr(X_1 = X_2)$$
$$= \Pr(X_1 = 0, X_2 = 0) + \Pr(X_1 = 1, X_2 = 1)$$
$$= p_1p_2 + (1-p_1)(1-p_2).$$

Kemudian jika $p1 = 1/2+\varepsilon_1$ dan $p2 = 1/2+\varepsilon_2$, dimana ε_1 dan ε_1 adalah probabilitas bias dan $-1/2 \leq \varepsilon_1, \varepsilon_2 \leq +1/2$, maka :

$$\Pr(X_1 \oplus X_2 = 0) = 1/2 + 2\varepsilon_1 \varepsilon_2$$

Dan bias $\varepsilon_1, \varepsilon_2$ dari $X_1 \oplus X_2 = 0$ adalah

$$\varepsilon_{1,2} = 2\ \varepsilon_1,\varepsilon_2$$

Ini dapat dikembangkan menjadi lebih dari dua buah random variabel binary, X1 hingga Xn, dengan probabilitas p1=1/2 + ε_1 hingga pn=1/2 + ε_n . Probabilitas $X_1 \oplus .. \oplus X_n = 0$ dapat ditentukan oleh *Pilling Up Lemma* dengan asumsi bahwa semua n random variabel binary adalah independent.

Pilling-up Lemma (Matsui[1])
Untuk n independent, random variabel binary, $X_1, X_{2, .. ,} X_n$

$$\Pr(X_1 \oplus .. \oplus X_2 = 0) = 1/2 + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i$$

atau ekuivalen dengan

$$\varepsilon_{1,2,..,n} = 2^{n-1} \prod_{i=11}^{n} \varepsilon_i$$

dimana $\varepsilon_{1, 2, ..,n}$ merepresentasikan bias dari $X_1 \oplus .. \oplus X_n = 0$.
Perhatikan bahwa jika $p_i = 0$ atau 1 untuk seluruh i, maka $\Pr(X_1 \oplus .. \oplus X_n = 0) = 0$ atau 1. Jika ada satu $p_i = ½$ maka $\Pr(X_1 \oplus .. \oplus X_n = 0) = ½$.

Pada pembuatan aproksimasi linear sebuah chiper, nilai X_i akan merepresentasikan aproksimasi linear dari S-Boxes. Sebagai contoh, terdapat 4 buah random binary variabel, $X_{1,} X_{2,} X_{3,} X_4$. Misalkan $\Pr(X_1 \oplus X_2 = 0) = 1/2 + 2\varepsilon_{1,2}$ dan $\Pr(X_2 \oplus X_3 = 0) = 1/2 + 2\varepsilon_{2,3}$. Maka

$\Pr(X_1 \oplus X_3=0) = \Pr([X_1 \oplus X_2] \oplus [X_2 \oplus X_3]=0)$

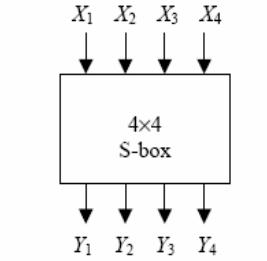
Jadi kita mengkombinasikan ekspresi linear menjadi bentuk linear ekspresion yang baru. Karena kita menggunakan variabel random $X_1 \oplus X_2$ dan $X_2 \oplus X_3$ yang saling independen maka kita dapat menggunakan *Pilling-up Lemma* untuk menentukan

$\Pr(X_1 \oplus X_3=0) = 1/2 + 2\varepsilon_{1,2}\, \varepsilon_{2,3}$
Maka,
$\varepsilon_{1,3} = 2\varepsilon_{1,2}\, \varepsilon_{2,3}$.

Dapat kita lihat, ekspresi $X_1 \oplus X_2 = 0$ dan $X_2 \oplus X_3 = 0$ adalah analogi dari aproksimasi linear dari S-Box dan $X_1 \oplus X_3 = 0$ adalah analogi dari aproksimasi chiper dimana perantara bit X_2 dihilangkan. Tentu saja sesungguhnya analisis akan lebih kompleks dengan melibatkan banyak aproksimasi S-Box.

2.3 Analisis komponen chiper

Sebelum masuk pada hal-hal yang lebih detil, kita akan melihat bagaimana vulnerebilitas linear dari suatu S-Box. Perhatikan pada gambar S-Box dibawah; dengan input X = [X1, X2, X3, X4] dan korespondensi ouptut Y = [Y1, Y2, Y3, Y4]. Semua aproksimasi linear dapat diperiksa dengan menentukan kegunaan mereka dengan menghitung setiap probabilitas bias. Maka kita dapat memeriksa seluruh ekspresi pada persamaan (1) dimana X dan Y adalah S-Box input dan output, secara urut.



Sebagai contoh, untuk S-box yang digunakan pada chiper, misal ekspresi linear
$X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0$ atau ekivalen dengan

$X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4$

Menerapkan seluruh 16 kemungkinan nilai input untuk X dan memeriksa korespondensi output nilai Y, dapat terlihat bahwa terdapat 12 dari 16 kemungkinan. Maka probabilitas biasnya adalah 12/16 – 1/2 = 1/4. Hal ini terdapat pada tabel dibawah.

Begitu juga pada persamaan

$X_1 \oplus X_4 = Y_2$

Dapat terlihat bahwa probabilitas biasnya adalah 0 dan untuk persamaan

$X_3 \oplus X_4 = Y_1 \oplus Y_4$

Probabilitas biasnya adalah 2/16 – 1/2 = -3/8. pada kasus yang terakhir, aproksimasi terbaiknya merupakan aproksimasi affinenya sebagaimana terlihat adanya tanda minus. Bagaimanapun juga, keberhasilan dari penyerangan berdasarkan pada nilai dari bias dan akan kita lihat, aproksimasi affine dapat digunakan untuk aproksimasi linear.

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

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

Contoh aproksimasi linear suatu S-Box

		Output Sum															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input Sum	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	−6	−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	+4	−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	+2	−2	−2	−6
	9	0	0	−2	−2	0	0	−2	−2	−4	0	−2	+2	0	+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 Aproksimasi Linear

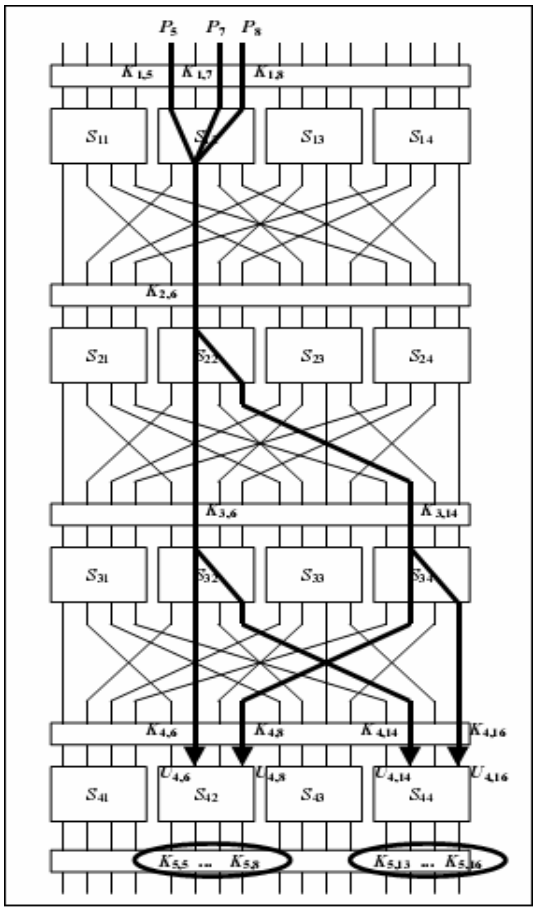
Hasil seluruh enumerasi dari seluruh aproksimasi linear dari S-Box tersaji pada tabel diatas. Setiap elemen pada tabel merepresentasikan jumlah kecocokan antara persamaan linear yang tersaji dengan hexadecimal sebagai “Input Sum” dan jumlah bit yang tersaji dengan hexadecimal sebagai “Output Sum” dikurang delapan. Maka dengan membagi elemen dengan 16 akan menghasilkan probabilitas bias antara kombinasi linear input dan output bit. Nilai hexadecimal merepresentasikan nilai penjumlahan, ketika dilihat sebagai binary maka nilai tersebut menyatakan jumlah variabel yang terlibat pada penjumlahan. Untuk kombinasi linear dari variable input diwakilkan oleh $a_1 \cdot X_1 \oplus a_2 \cdot X_2 \oplus a_3 \cdot X_3 \oplus a_4 \cdot X_4$ dimana $a_i \in \{0,1\}$ dan “.” Merepresentasikan binary AND, nilai hexadecimal merepresentasikan nilai biner dari $a_1a_2a_3a_4$, dimana a_1 adalah MSB(*Most significant bit*). Seperti halnya input, kombinasi linear dari bit-bit output $b_1 \cdot Y_1$? $b_2 \cdot Y_2$? $b_3 \cdot Y_3$? $b_4 \cdot Y_4$ dimana $b_i \in \{0,1\}$, nilai hexadecimal merepresentasikan vektor biner $b_1b_2b_3b_4$. Maka bias dari persamaan linear $X_3 \oplus X_4 = X_1 \oplus Y_4$(hex input 3 dan hex output 9) adalah -6/16 = -3/8 dan probabilitas kebenarnnya adalah 1/2 - 3/8 =1/8.

Beberapa properti dasar dari tabel aproksimasi linear dapat terlihat. Sebagai contoh probabilitas dari penjumlahan output bit dari subset selain subset kosong adalah 1/2 karena kombinasi linear dari ouptut bit harus sama dengan jumlah 0 dan 1 pada bijective S-Box. Juga kombinasi linear yang no-bit output akan selalu sama dengam kombinasi linear dari no-bit input menghasilkan bias 1/2 dan nilai + 8 pada ujung kiri atas. Sehingga seluruh nilai pada baris paling atas akan bernilai 0 kecuali nilai pada tabel paling kiri. Begitu juga dengan kolom petama dimana seluruh nilainya bernilai 0 kecuali pada baris pertama. Dapat terlihat juga bahwa jumlah seluruh nilai pada suatu baris atau suatu kolom akan bernilai +8 atau -8.

2.4 Pembangunan Aproksimasi Linear untuk Chiper Keseluruhan Chiper

Ketika aproksimasi linear telah dicompiled untuk SPN(*Subtitution-Permutation Network*) pada S-Box, kita mempunyai data untuk menentukan aproksimasi linear untuk keseluruhan chiper dar i bentuk persamaan 1. Hal ini dilakukan dengan cara mengkonkatkan aproksimasi -aproksimasi linear S-Boxes. Dengan membangun suatu aproksimasi linear dengan mengikutsertakan bit-bit plaintext dan bit-bit output dari putaran terakhir yang kedua dari S-Boxes, kita dapat menyerang chiper dengan cara merecover subset dari suatu bit-bit

subkey yang akan mengikuti putaran terkahir. Kita ilustrasikan dengan suatu contoh.
Milsalkan terdapat aproksimasi dengan S_{12}, S_{22}, S_{32}, S_{34} seperti yang tampak pada gambar dibawah.



Contoh Aproksimasi Linear

Perhatikan bahwa ini pembuatan ekspresi untuk 3 putaran pertama dari chiper, bukan full 4 putaran. Kita dapat lihat bahwa bagaimana ini sangat berguna untuk menurunkan bit subkey setelah putaran terakhir pada bagian selanjutnya.

Kita menggunakan aproksimasi S-Box dibawah ini:

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

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

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

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

Misalkan U_i (V_i) merepresentasikan blok 16 bit pada input(output) dari putaran i S-Box dan $U_{i,j}$ ($V_{i,j}$) merepresentasikan bit ke j pada blok U_i (V_i) (dimana bit – bit dinomori 1 sampai 16 dari kiri ke kanan). Begitu juga, misalkan K_i merepresent subkey dari blok of bit yang diXOR kan pada input di putaran ke i, dengan pengecualian bahwa K_5 adalah kunci yang diXOR kan pada output di putaran ke 4.

Maka $U_1 = P \oplus K_1$ dimana P adalah blok dari 16 bit plainteks dan "⊕" adalah operasi bit XOR. Dengan menggunakan aproksimasi linear dari putaran pertam, maka kita akan mempunyai :

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

dengan probabilitas 3/4. Untuk aproksimasi pada putaran ke 2, kita mempuyai

$V_{2,6} \oplus V_{2,8} = U_{2,6}$

dengan probabilitas ¼.
Karena $U_{2,6} = V_{1,6} \oplus K_{2,6}$, kita dapat menentukan aproksimasi :

$V_{2,6} \oplus V_{2,8} = V_{1,6} \oplus K_{2,6}$

dengan probabilitas 1/4 dan mengkombinasikan ini dengan (2) yang mempunyai probabilitas 3/4 akan menghasilkan

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

dengan nilai probabilitas 1/2 + 2(3/4-1/2)(1/4-1/2) = 3/8 (dengan bias -1/8) dengan menngunakan Pilling-up Lemma. Perhatikan bahwa kita mengunakan asumsi bahwa aproksimasi dari S-Box adalah independen, walaupun tidak sepenuhnya benar, dapat bekerja dengan baik untuk chiper pada umumnya.

Pada putaran ke tiga, kita lihat bahwa

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

dengan probabilitas 1/4 dan

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

dengan probabilitas 1/4. Maka, dengan $U_{3,6} = V_{2,6} \oplus K_{3,6}$ and $U_{3,14} = V_{2,8} \oplus K_{3,14}$,

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

Dengan probabilitas 1/2 + 2(1/4-1/2) = 5/8 (dengan bias of +1/8). Sekali lagi kita telah menerapkan Pilling-Up Lemma.

Kemudian dengan mengkombinasikan (3) dan (4), untuk menggabungkan keempat aproksimasi S-Box, kita mendapat

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

Karena $U_{4,6} = V_{3,6} \oplus K_{4,6}$, $U_{4,8} = V_{3,14} \oplus K_{4,8}$, $U_{4,14} = V_{3,8} \oplus K_{4,14}$, and $U_{4,16} = V_{3,16} \oplus K_{4,16}$, maka kita dapat menulis

$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus \sum K = 0.$

dimana

$\sum K = K1,5 \oplus K1,7 \oplus K1,8 \oplus K2,6 \oplus K3,6 \oplus K3,14 \oplus K4,6 \oplus K4,8 \oplus K4,14 \oplus K4,16$

dan $\sum K$ adalah pasti apakah 0 atau 1 tergantung dari kunci chipernya. Dengan menggunakan Pilling-Up Lemma, persamaan diatas memiliki probabilitas $1/2+2^3$ $(3/4-1/2)(1/4-1/2)^3$ = 15/32 (dengan bias -1/32).

Sekarang karena $\sum K$ pasti, maka,

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

Mempunyai probabilitas 15/32 atau (1-15/32)= 17/32, tergantung apakah $\sum K = 0$ atau 1. Dengan kata lain, kita sekarang telah memiliki aproksimasi lineaar dari tiga putaran pertama chiper dengan nilai bias 1/32. Selanjutnya kita akan menentukan kunci dari nilai bias tersebut.

2.5 Mengekstrak bit kunci

Ketika aproksimasi linear dari putaran R-1 dtemukan, penyerangan dapat dilakukan dengan cara merecover bit-bit dari subkey terakhir. Pada contoh ini, memungkinkan kita untuk mengekstrak bit dari subkey K5 dengan tiga putaran aproksimasi linear. Kita menyebut bit yang direcover dari subkey terakhir sebagai target partial subkey. Secara spesifik, target spatial subkey adalah bit-bit dari subkey terakhir yang berasosiasi dengan S-Boxes pada putaran terakhir yang dipengaruhi oleh bit-bit data yang terlibat pada aproksimasi linear.

Proses selanjutnya melibatkan sebagian dekrispi pada putaran terakhir dari chiper. Secara spesifik, untuk semua kemungkinan nilai target partial subkey, koresponden dari chiper tersebur diXORkan dengan bit-bit target partial subkey dan hasilnya dimasukkan lagi ke dalam korespondensi S-Boxnya. Hal ini dilakukan untuk semua sampel *known* plainteks/chiperteks dan jumlahnya disimpan untuk setiap nilai dari target partial subkey. Jumlahnya dari suatu target partial subkey akan di*increment* ketika ekspresi linear bernilai *true* untuk bit-bit yang dimasukkan pada putaran terakhir S-Box (ditentukan oleh partial dekripsi) dan bit-bit pada *known* plainteks. Nilai target partial subkey yan gmemiliki perbedaan terbesar dari setengah jumlah sampel plainteks/chiperteks diasumsikan sebagai sebagai niali target partial subkey yang benar. Hal ini berlaku karena diasumsikan bahwa partial subkey yang benar akan menghasilkan aproksimasi linear dengan nilai probabilitas berbeda signifikan dari 1/2 (Apakah itu itu diatas atau dibawah 1/2 tergantung apakah ekspesi linear atau affine linearnya adalah aproksimasi yang terbaik dan ini juga tergantung dari *unknown value* dari bit-bit subkey yang secara implist terlibat pada ekspresi linear). Subkey yang salah diasumsikan menghasilkan tebakan random pada bit-bit yang memasuki S-Box pada putaran terakhir dan sebagai hasilnya, ekspresi linear yang memiliki probabilitas mendekati 1/2.

Sekarang kita lakukan pada contoh kasus kita. Ekspresi linear (5) mempengaruhi input ke S-Box S_{42} dan S_{44} pada putaran terakhir. Untuk setiap sampel plainteks/chiperteks, kita mencoba keseluruh 256 nilai pada target partial subkey[$K_{5,5}$... $K_{5,8,}$ $K_{5,13}$..$K_{5,16}$]. Untuk setiap nilai partial subkey kita menaikkan *count* ketika persamaan (5) bernilai *true*, dimana kita menentukan nilai dengan [$U_{4,5}$...$U_{4,8}$, $U_{4,13}$...$U_{4,16}$] dengan melakukan proses *backward* dengan target partial subkey dan S-Box S_{24} dan S_{44}.
Count yang memiliki deviasi terbesar dari jumlah sampel plainteks/chiperteks diasumsikan sebagai nilai yang benar. Apakah deviasi itu positif atau negatif akan tergatung dari nilai dari bit-bit subkey yang terliat pada $\sum K$. Ketika $\sum K=0$, aproksimasi linear dari (5) akan sesuai dengan estimasi (dengan probabilitas <1/2) dan ketika $\sum K = 1$, maka probabilitas >1/2.

Kita mensimulasikan penyerangan dengan menggunakan 10000 nilai *known* plainteks/chiperteks dan mengikuti proses kriptanalitis untutk nilai subkey $[K_{5,5..} K_{5,8}] = [0010]$ (hex 2) dan $[K_{5,13} .. K_{5,16}] = [0100]$ (hex4). Seperti yang diharapkan, *count* yang paling berbeda dari 5000 berkorespondensi dengan target partial subkey $[2,4]_{hex}$, menyatakan bahwa penyerangan telah berhasi menurunkan bit-bit subkey. Tabel di bawah meng*highlight* partial data summary yang diturunkan dari *count* subkey(keseluruhan data melibatkan 256 *entries*, satu untuk setiap nilai partial subkey). Nilai di tabel mengindikasi kan nilai bias diturunkan dari

$$| bias | = | count - 5000 | / 10000$$

dimana *count* adalah jumlah yang berkorespondensi pada suatu partial subkey

partial subkey $[K_{5,5}...K_{5,8}, K_{5,13}...K_{5,16}]$	| bias |	*partial subkey* $[K_{5,5}...K_{5,8}, K_{5,13}...K_{5,16}]$	| bias |
1 C	0.0031	2 A	0.0044
1 D	0.0078	2 B	0.0186
1 E	0.0071	2 C	0.0094
1 F	0.0170	2 D	0.0053
2 0	0.0025	2 E	0.0062
2 1	0.0220	2 F	0.0133
2 2	0.0211	3 0	0.0027
2 3	0.0064	3 1	0.0050
2 4	**0.0336**	3 2	0.0075
2 5	0.0106	3 3	0.0162
2 6	0.0096	3 4	0.0218
2 7	0.0074	3 5	0.0052
2 8	0.0224	3 6	0.0056
2 9	0.0054	3 7	0.0048

Seperti terlihat pada tabel, bias terbesar terjadi pada nilai partial subkey $[K_{5,5}.. K_{5,8}, K_{5,13}..K_{5,16}]=[2,4]$ dan pada pengamatan ini benar ditemukan untuk penyelesaian set dari nilai partial subkey.

Percobaan ini menentukan nilai bias dari 0.0336 adalah sangat dekat dengan nilai yang diharapkan yakni 1/32 = 0.03125. Perhatikan bahwa meskipun target partial subkey yan benar jelas memiliki bias yang paling tinggi, nilai bias besar yang lain terjadi, ini menandakan bahwa pengujian target partial subkey yang salah tidak tepat ekivalen dengan data random pada ekspresi linear (dimana bias dapat diharapkan bernilai mendekati 0). Inkosistensi pada percobaan bias terjadi karena beberapa alasan termasuk properti S-Box yang mempengaruhi deskripsi partial untuk nilai partial subkey yang berbeda, ketidakakuratan dari asumsi independen yang dibutuhkan yang digunakan pada Pilling-Up Lemma, dan pengaruh dari *linear hulls*(skenario aproksimasi linear yang melibatkan plainteks yang sama dan putaran terakhir bit-bit input teteapi set yang berebda dari aktif S-Box dapat mengkombinasikan untuk mendapatkan probabilitas linear ang lebih tinggi dari yang diprediksikan leh satu set aktif S-Box[2].).

2.6 Kompleksitas penyerangan

S-Box yang terlibat pada aproksi linear kita sebut dengan aktif S-Box. Pada gambar diatas tadi, empat S-Box pada putaran 1 sampai 3 yang dihighlight sedang aktif. Probabilitas yang menyatakan ekspresi linear bernilai benar dihubungkan dengan aktif S-Box dan jumlah S-Box yang aktif. Secara umum, makin besar nilai bias pada S-Box maka makin besar nilai bias pada ekspresi linear secara keseluruhan. Dan juga, makin sedikit aktif S-Box maka makin besar pula nilai bias ekspresi linear secara keseluruhan.

Misalkan ε merepresentasikan bias dari probabilitas 1/2 ekspresi linear complete chiper. Pada papernya, Matsui memperlihatkan bahwa jumlah *known* plainteks yang dibutuhkan dalam penyerangan ialah ε^{-2}, dan NL merepresentasikan jumlah *known* plainteks yang digunakan.

$$N/L \approx 1/\varepsilon^2$$

3. Differential Kriptanalisis

3.1 Gambaran Umum

Differential kriptanalis mengeksploitasi tingginya probabilitas dari kepastian terjadinya dari perbedaan plainteks dan perbedaan pada putaran terakhir dari chiper. Sebagai contoh, misalkan sebuah sistem dengan input $X = [X_1 X_2 X_n]$ dan output $Y = [Y_1 Y_2 Y_3]$. Misalkan input pada sistem ialah X' dan X'' dengan korespondensi output Y' dan y". Perbedaan input diberikan dalam $\Delta X = X' \oplus X"$ dimana $\oplus$ merepresentasikan operasi bit XOR dari n-bit vektor, maka

$$\Delta X = [\Delta X_1 \Delta X_2 .. \Delta X_n]$$

dimana $\Delta X_i = \Delta X_i' \oplus \Delta X_i''$ dengan X_i' dan X_i'' merepresentasikan bit ke i dari X' dan X".
Begitu juga, $\Delta Y_i = \Delta Y_i' \oplus \Delta Y_i''$ sebagai perbedaan output dan

$\Delta Y = [\Delta Y_1 \Delta Y_2 .. \Delta Y_n]$

dimana $\Delta Y_i = \Delta Y_i' \oplus \Delta Y_i''$

Pada pengacakan chiper yang ideal, probabilitas bahawa suatu output ΔY terjadi dengan input ΔX ialaah $1/2^n$ dimana n adalah jumlah bit X. Diferential kriptanalis mengexploitasi suatu skenario dimana terdapat sutu input ΔY dengan input ΔX dengan probabilitas tinggi p_D (sebagai contoh : lebih dari $1/2^n$). Pasangan ($\Delta X \Delta Y$) disebut sebagai diferential.

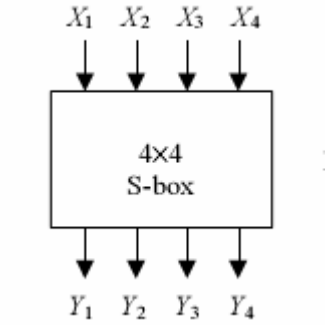
Differential kriptanalisis adalah penyerangan chosen plainteks, yan gberati bahwa penyerang mampu mengetahui input dan menguji output dalam usaha untuk menurunkan kunci. Untuk diferential kriptanlis, penyerang akah memilih pasangan input, X' dan X", untuk membuat suatu ΔX, mengetahui nilai ΔX, suatu ΔY terjadi dengan probabilitas yang tinggi.

Pada makalah ini, kita akan membangun diferential ($\Delta X \Delta Y$) dengan bit-bit plainteks diwakilkan dalam X dan input pada putaran terakhir chiper diwakilkan dalam Y. Kita akan melakukan ini dengan cara memeriksa suatu deiferential karateristik dimana diferential karakteristik tersebut adalah urutan perbedaan input dan output pada putaran-putaran sehingga perbedaan output pada suatu putaran berkorespondensi menjadi perbedaan input pada putaran selanjutna. Dengan menggunakan hampiran, differntial karakteristik memberi kita kemungkinan untuk mengekploitasi informasi yang datang pada putaran terakhir untul mendapatkan bit-bit dari lapisan terakhit subkey.

Seperti linear kriptanalisis, kita juga akan memeriksa properti satu-persatu S-Box dan menggunakan properti ini untuk menentukan diferential karakteristik secar keseluruhan. Secara spesifik, kita perhatikan perbedaan input dan output dari S-Box untuk menentukan suatu pasangan perbedaan probabilitas tinggi (*high probability difference pair*). Mengkombinasikan perbedaan pasangan S-Box putaran demi putaran, sehigga bit-bit perbedaan output bit-bit nonzero, memungkinkan kita untuk menentukan diferential probabilitas tinggi yang terdiri dari perbedaan plainteks dan perbedan input pada putaran terkhir. Bit-bit subkey chiper akan dihilangkan dari persamaan diferential karena mereka ada pada kedua data set.

3.2 Analisis komponen Chiper

Kita akan memeriksa perbedaan pasangan pada S-Box. Misalkan 4x4 S-Box dengan input $X = [X_1X_2X_3X_4]$ dan output $Y = [Y_1Y_2Y_3Y_4]$.



Semua perbedaaan pasangan dari S-Box, (ΔX, ΔY), dapat diperiksa dan probailitas ΔY dengan masukan ΔX dapat diturunkan dengan cara memeriksa pasangan input (X', X") dimana $\Delta X = X' \oplus X"$. Karena keterurutan pasangan tidaklah relevan, maka untuk 4x4 S-Box kita hanya memperhatikan ke 16 nilai dari X' dan kemudian nilai ΔX ditentukan dengan batasan $X" = X' \oplus \Delta X$.

Kita dapat menurunkan nilai ΔY dari setiap pasngan input ($X', X" = X' \oplus \Delta X$). Contoh nilai biner dari X, Y, dan nilai korespondensi ΔY dari pasangan input ($X, X \oplus \Delta X$) disajikan pada tabel di bawah ini dengan nilai ΔX = 1011 (hex b), 1000 (hex 8), dan 0100 (hex 4).

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Contoh *difference pair* suatu S-Box

Tiga kolom terakhir pada tabel memeperlihatkan nilai ΔY untuk suatu nilai X dan suatu nilai ΔX dari tiap kolom. Dari tabel terlihat bahwa jumlah terjadinya ΔY = 0100 untuk ΔX = 1011 adalah 8 dari 16; jumlah terjadinya ΔY = 1011 untuk ΔX = 1000 adalah 4 dari 16; jumlah terjadinya ΔY = 1010 untuk ΔX = 0100 adalah 0 dari 16. Jika S-Box tesebut "ideal", semua pasngan nilai harus 1 atau mempunyai probabilitas 1/16 dari suatu ΔY untuk suatu nilai ΔX(terjadinya S-Box yang ideal adalh suatu hal yang mustahil).

Kita dapat mentabulasikan semua data pada S-Box pada tabel distribusi yang berbeda dimana barisnya mewakilkan nilai ΔX (pada hexadesimal) dan kolomnya mewakilkan nilai ΔY (pada hexadecimal). Tabel distribusi S-Box dibawah akan ditampilkan pada tabel selanjutnya.

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

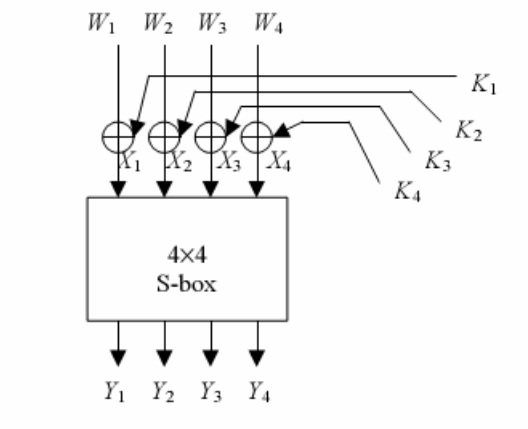
Setiap elemen dari tabel menampilkan jumlah terjadinya korespondensi perbedaan output ΔY untuk suatu perbedaan input ΔX. Perhatikan, selain kasus special (ΔX=0, ΔY=0), nilai terbesar pada tabel adalah 8, yakni untuk ΔX=B dan ΔY=2. Maka, probabilitas ΔY=2 untuk suatu pasangan randomnya dari suatu nilai input yang memenhui ΔX=B adalah 8/16. Nilai terkecil pada tabel adalah 0; terjadi pada banyak pasangan. Pada kasus ini, probabilitas nilai ΔY terjadi untuk suatu ΔX adalah 0.

		Output Difference															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Input Difference	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
	2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
	3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
	4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
	5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
	6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
	7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
	8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
	9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
	A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
	B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
	C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
	D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
	E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
	F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

Tabel ditribusi difference

Ada beberapa properti umum dari tabel perbedaan distribusi yang harus disebutkan. Pertama, harus diperhatikan bahwa jumlah seluruh elemen pada suatu baris adalah $2^n = 16$; begitu juga jumlah pada suatu baris yakni $2^n = 16$. Juga, seluruh nilai elemen adalah genap; hal ini terjadi karena pasangan input nilai (atau output) dilambangkan dengan (X', X") mempunyai nilai yang sama dengan nilai ΔX sebagai pasangan (X', X") karena ΔX = X" ⊕ X' = X' ⊕ X". Begitu juga, perbedaan input dari ΔX =0 menghasilkan perbedaan output ΔY =0 untuk koresepondensi satu-satu dari S-Box. Nilai pojok kanan atas adalah 2^n=16 dan semua nilai lainn pada kolom dan baris pertama adalah 0. Akhirnya jika kita dapat membuat S-Box yang ideal, yang tidak memberikan informasi diferential suatu nilai output untuk suatau nilai input, S-Box akan mempunyai nilai elemen 1 untuk semua nilai dan probabilitas terjadinya suatu nilai ΔY untuk suatu ΔX adalah $1/2^n$ atau 1/16. Namun seperti yag telah diebutkan di atas hal ini tidak mungkin.

Sebelum kita lanjut untuk mengkombinasikan pasangan perbedaan S-Box untuk menurunkan diferential karakteristik dan mengestimasi diferential yang baik untuk digunakan pada penyerangan, perhatikan pengaruh kunci pada diferential S-Box. Perhatikan gambar di bawah berikut.



Input pada "unkeyed" S-Box adalah X dan output Y. Pada struktur chiper perhatikan kunci-kunci yang digunakan pada input dari setiap S-Box. Pada kasus ini, misalkan input untuk mengunci S-Box W = $[W_1W_2W_3W_4]$, kita misalkan perbedaan input S-Box yang memiliki kunci adalah

$\Delta W = [W_1' \oplus W_2''\ W_1' \oplus W_2''\ W_n'' \oplus W_n'']$

dimana W' = [W_1' W_2' .. W_n'] dan W" = [W_1" W_2" .. W_n"] sebagai nilai input.

Karena bit-bit kunci tetap sama untuk W' dan W",

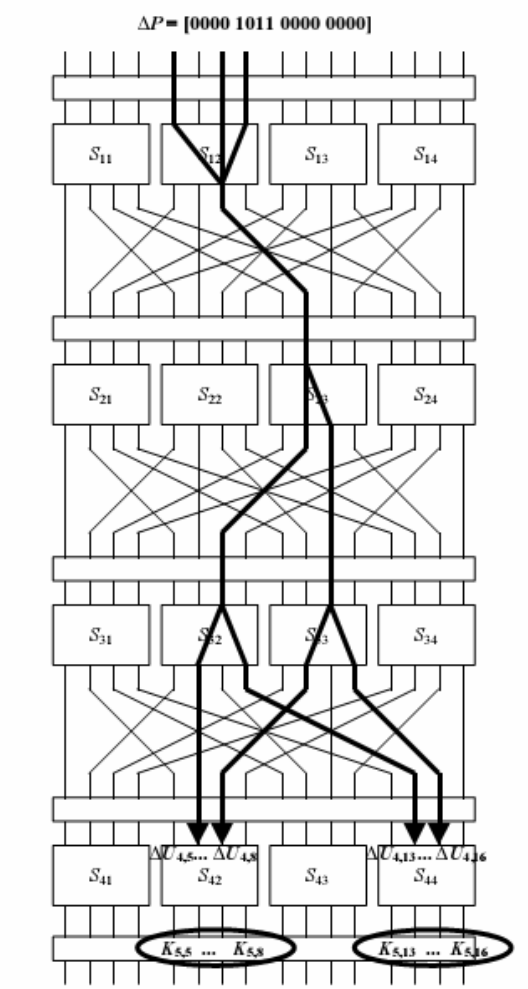
$$\Delta W_i = W_i' \oplus W_i'' = (X_i' \oplus K_i) \oplus (X_i'' \oplus K_i)$$
$$= X_i' \oplus X_i'' = \Delta X_i$$

karena $K_i \oplus K_i = 0$ maka bit-bit kunci tidak mempunyai pengaruh pada perbedaan nilai input dan dapat diabaikan. Dengan kata lain, S-Box yang memiliki kunci mempunyai tabel perbedaan distribusi yang sama dengan "unkeyed" S-Box.

3.3 Membangun karakteristik diferential

Ketika informasi diferensial telah di-compile untuk S-Box pada SPN(*Subtitution Permutation Network*), kita mempunyai data untuk menentukan karakteristik diferential untuk keseluruhan chiper. Hal ini dapat dilakukan dengan cara mengkonkatenasi pasangan perbedaan S-Box yang cocok. Dengan membangun karakteristik diferential pada suatu pasangan perbedaan S-Box pada setiap putaran, seperti suatu diferential yang digunakan pada bit-bit plainteks dan bit-bit data pada input dari putaran terakhir S-Box memungkinkan kita untuk menyerang chiper dengan cara merecover bit-bit subset subkey hingga putaran terakhir. Kita ilustrasikan karakteristik diferential dengan suatu contoh.

Misalkan karakteritik diferential yang terdiri dari S_{12}, S_{23}, S_{32}, dan S_{33}. Seperti pada linear kriptanalisis, berikut visualisasinya.



Contoh Karakteristik diferential

Diagaram tersbut menggambarkan pengaruh perbedaan bit-bit non-zeo sebagaimana mereka melewati network, highlight S-Box merupakan S-box aktif. Perhatikan bahwa hal ini membangun karakteristik diferential untuk 3 putaran dari chiper bukan untuk keempat putaran. Kita lihat bagaimana hal ini berguna untuk menetukan bit-bit dari subkey terakhir pada bagian selanjutnya.

Kita menggunakan pasangan perbedaan S-Box berikut ini :

S_{12}: $\Delta X = B \rightarrow \Delta Y = 2$ dengan probablilitas 8/16
S_{23}: $\Delta X = 4 \rightarrow \Delta Y = 6$ dengan probablilitas 6/16
S_{32}: $\Delta X = 2 \rightarrow \Delta Y = 5$ dengan probablilitas 6/16
S_{33}: $\Delta X = 2 \rightarrow \Delta Y = 5$ dengan probablilitas 6/16

Semua S-Box lainnya akan mempunyai zero input difference dan konsekuensinya zero output diffenrece.

Perbedaan input pada chiper adalah ekivalen dengan perbedaan input pada putaran pertama dan dengan

$\Delta P = \Delta U_1$ = [0000 0000 1011 0000]

sekali lagi kita menggunakan U_i sebagai input pada putaran ke-i S-Box dan V_i sebagai output dari putaran ke-i S-Box. ΔU_i dan ΔV_i merupakan perbedaan korespondensi . Hasilnya

ΔV_i = [0000 0010 0000 0000]

dengan memperhatikan perbedaan pasangan S_{12} diatas maka

ΔU_2 = [0000 0000 0100 0000]

dengan probabilitas 8/16 = 1/2 untuk suatu ΔP.

Sekarang putaran kedua diferential dengan menggunakan pasangan diferential S_{23} menghasilkan

ΔV_2 = [0000 0000 0110 0000]

dan permutasi putaran ke 2 menghasilkan

ΔU_3 = [0000 0010 0010 0000]

dengan probabilitas 6/16 untuk suatu ΔU_2 dan probabilitas 8/16 x 6/16 = 3/16 untuk suatu ΔP. Pada menentukan probabilitas untuk suatu plainteks dengan perbedaan ΔP, kita berasumsi bahwa diferential pada putaran pertama independen terhadap diferential pada putaran kedua, dan probabilitas keduanya terjadi ditentukan oleh produk dari probabilitas.

Selanjutnya, kita dapat menggunakan perbedan S-Box pada putaran ketiga, S_{23} dan S_{33}, dan permutasi dari putaran ketiga untuk mendapatkan

ΔV_3 = [0000 0101 0101 0000]

dan

ΔU_4 = [0000 0110 0000 0110] (6)

dengan probabiltas $(6/16)^2$ untuk suatu ΔU_3 dan probabilitas 8/16 x 6/16 x $(6/16)^2$ = 27/1024 untuk suatu perbedaan plainteks ΔP dan juga kita berasumsi bahwa terdapat indepensi antara difference pair untuk semua S-Box pada seluruh putaran.

Pada proses kriptanalisi, banyak pasangan plainteks dimana ΔP = [0000 1011 0000 0000] akan dienkripsi. Dengan probabilitas tinggi, 27/1024, karakteristik differential yang diilustrasikan akan terjadi. Kita menyebut pasangan untuk ΔP sebagai *right pairs*. Difference pair plainteks dimana karakteristik tidak terjadi disebut sebagai *wrong pairs*.

3.4 Ekstrasi bit-bit kunci

Sekali karakteristik differential putaran R-1 ditemukan dengan probabilitas yang cukup besar, maka memungkinkan untuk menyerang chiper dengan cara merecover bit-bit dari subkey terakhir. Pada contoh kasus kita, memungkinkan untuk mengekstrak bit-bit tersebut dari K_5. Proses selanjutnya meliputi partial deskripsi dari chiper dan memeriksa input hingga putaran terakhir untuk menentukan apakah right pair mungkin telah terjadi. Selanjutnya, kita menyebut bit-bit subkey yang mengikuti putaran terakhir pada output dari S-Box di putaran terakhir yang dipengaruhi oleh *non-zero differences* pada output diferential sebagai *target partial subkey*. Partial dekripsi dari putaran terakhir akan terlibat, untuk seluruh S-Box pada putaran terakhir yang dipengaruhi oleh *non-zero differences* pada differential, XOR dari chiperteks dengan bit-bit target partial subkey dan mem*backward* data melalui S-Box, dimana semua kemungkinan nilai dari bit-bit target subkey akan dicoba.

Dekripsi partial dilakukan untuk setiap pasangan chiperteks yang berkorespondensi dengan pasangan plainteks yang digunakan untuk mengenerate perbedaan input ΔP untuk semua nilai taget partial subkey yang mungkin. Jumlahnya kemudian diincrement ketika perbedaan untuk input pada putaran terakhir berkorespondensi pada nilai yang diharapkan dari differential karakteristik. Nilai partial subkey yang mempunyai nilai terbesar diasumsikan sebagai indikasi dari nilai yang benar dari bit-bit subkey. Ini berhasil karena nilai partial subkey yang diasumsikan akan menghasilkan perbedaan pada putaran terakhir secara teratur seperti yang diharapkan dai karakteristik(contohnya terjadinya right pair) karen karakteristik mempunyai probilitas keterjadian yang tinggi. (Ketika wrong pair terjadi, walapun dengan deskripsi partial dengan subkey yang benar, jumlah dari correct subkey

sepertinya tidak akan diincrement). Subkey yang salah diasumsikan menghasilkan relative random guess pada bit-bit yang memasuki S-Box dari putaran terakir dan sebagai hasilnya, perbedaan akan menjadi seperti yang diharapkan dari karakteristik dengan probabilitas yang sangat rendah.

Perhatikan penyerangan pada contoh chiper, karakteristik differential mempengaruhi input pada S-Box S_{42} dan S-Box$_{44}$ pada putaran terakhir. Untuk setiap pasangan chiperteks , kita akan mencoba ke 256 nilai yang mungkin untuk [$K_{5,5}$..$K_{5,8}$ $K_{5,13}$..$K_{5,16}$]. Untuk setiap nilai partial subkey, kita akan mengincrement jumlahnya ketika perbedaan input pada putaran terakhir yang ditentukan oleh deskripsi partial sama dengan(6), dimana kita menentukan nilai dari [$\Delta U_{4,5}$.. $\Delta U_{4,8,}$ $\Delta U_{4,13}$.. $\Delta U_{4,16}$] dengan membackward data melalui partial subkey dan S-Box S_{22} dan S_{44}. Untuk setiap nilai partial subkey, count merupakan jumlah terjadinya perbedaan yang konsisten dengan right pairs (dengan asumsi partial subkey benar). Count yang terbesar diambil sebagai nilai yang benar karena kita berasumsi bahwa kita melihat probilitas keterjadian yang tinggi dari right pair.

Perhatikan bahwa tidak penting untuk mengeksekusi partial deksripsi untuk semua pasangan chiperteks. Karena perbedaan input pada putaran terakhir hanya mempengaruhi 2 S-Box, ketika karakteristik telah terjadi (contohnya untuk right pair) perbedaan bit chiperteks yang berkorespondensi pada S-Box S_{41} dan S_{43} adalah 0. Kita dapat membuang banyak pasangan dengan cara menolak pasangan chiperteks dimana 0 nya tidak muncul pada subblock yang cocok dari suatu perbedaan chiperteks. Pada kasus ini, karena chiperteks tidak berkorespondensi dengan right pair, maka tidak butuh untuk memeriksa [$\Delta U_{4,5}$..$\Delta U_{4,8}$, $\Delta U_{4,13}$..$\Delta U_{4,16}$].

Kita telah mensimulasikan penyerangan basic chiper keyed dengan menggunakan subkey yang tergenerate secara acak dengan men-generate 5000 pasang chosen plainteks/chiperteks (misalnya 10000 enkripsi dengan pasangan plainteks yang memenuhi ΔP = [0000 1011 0000 0000]) dan mengikuti proses seperti yang telah disebutkan di atas. Nilai target partial subkey yang benar ialah [$K_{5,5}$ .. $K_{5,8}$, $K_{5,13}$ .. $K_{5,16}$] = [0010, 0100] = $[2,4]_{hex}$. Seperti yang diharapkan, count yang paling besar teramati untuk nilai partial subkey $[2,4]_{subkey}$, mengkonfirmasikan bahwa penyerangan berhasil diturunkan dari bit-bit subkey. tabel dibawah ini menhighlight ringkatan partial dari data yang diturunkan dari count subkey. (keseluruhan data terdiri dari 256 data entry, satu dari setiap nilau partial subkey). nilai pada tabel mengindikasikan bahwa estimasi probabilitas keterjadian right pair untuk kandidat partial subkey diturunkan dari

Prob = count / 5000

Dimana count adalah count yang berkorespondendsi dengan suatu partial subkey. Dapat dilihat bahwa dari contoh hasil dari tabel , probabilitas terbesar terjadi untuk nilai partial subkey [$K_{5,5}$ .. $K_{5,8}$, $K_{5,13}$ .. $K_{5,16}$] = $[2,4]_{hex}$ dan pengamatan ini ditemukan benar untuk suatu set nilai partial subkey.

Pada contoh ini, kita mengharapkan proabilitas keterjadiaan right pair menjadi p_D = 27/1024 = 0.0264 dan kita menemukan secara eksperimen probabilitas untuk nilai subkey [2,4] memberi nilai p_D = 0.0244. Perhatikan bahwa terkadang nilai count besar lainnya terjadi pada target partial subkey yang salah. Ini mengindikasikan bahwa pengujian target partial subkey yang salah tidak memeberikan niali ekivalensi yang tepat sama dengan random differences terhadap nilai diferential yang diharapkan. Ada beberapa faktor yang mempengaruhi count menjadi lain dari ekspetasi teori kita termasuk properti S-Box yang mempengaruhi partial dekripsi untuk differential partial subkey, impresisi dari asumsi independen yang dibutuhkan untuk menentukan probabilista karakteristik, dan konsep bahwa differential dibentuk dari beberapa karakteristik differential.

partial subkey [$K_{5,5}$...$K_{5,8}$, $K_{5,13}$...$K_{5,16}$]	prob	*partial subkey* [$K_{5,5}$...$K_{5,8}$, $K_{5,13}$...$K_{5,16}$]	prob
1 C	0.0000	2 A	0.0032
1 D	0.0000	2 B	0.0022
1 E	0.0000	2 C	0.0000
1 F	0.0000	2 D	0.0000
2 0	0.0000	2 E	0.0000
2 1	0.0136	2 F	0.0000
2 2	0.0068	3 0	0.0004
2 3	0.0068	3 1	0.0000
2 4	**0.0244**	3 2	0.0004
2 5	0.0000	3 3	0.0004
2 6	0.0068	3 4	0.0000
2 7	0.0068	3 5	0.0004
2 8	0.0030	3 6	0.0000
2 9	0.0024	3 7	0.0008

Hasil percobaan untuk differential attack

3.5 Kompleksitas penyerangan

Pada differential kriptanalis, kita menyebut S-Box yang terlibat pada karakteristik yang mempunyai non-zero input difference (dan maka menghasilkan non-zero output difference) disebut sebagai aktif S-Box. Secara umum, makin besar probabilitas differntial suatu aktif S-Box maka makin besar probabilitas karakteristik untuk keseluruhan chiper. Dan juga, makin sedikit aktif S-Box, makin besar probailitas karakteristiknya. Seperti linear cryptanalis, kita menuju data yang digunakan untuk penyerangan ketika menentukan kompleksitas kriptanalisis. Oleh karena itu, kita berasumsi nahwa jika kita dapat mempunyai N_D plainteks maka kita dapat memproses mereka. Secara umum, sangatlah kompleks untuk menentukan secara tepat jumlah chosen plainteks pairs yang dibutuhkan untuk penyerangan. Namun, hal it dapat di tunjukkan bahwa *good rule of thumb* untuk jumlah chosen plainteks pair, N_D, dibutuhkan untuk menentukan right pair ketika mencoba kandidat subkey adalah

$N_D \approx c/p_D$

Dimana p_D adalah probabilitas karakteristik differential dari putaran R-1 dari keseluruhan R putaran chiper. Dan c adalah konstanta kecil. Mengasumsikan bahwa keterjadian difference pair pada aktif S-Box adalah independen, probabilitas karakteristik differnetial nya

$$p_D = \prod_{i=1}^{\gamma} \beta_i$$

dimana γ adalah jumlah S-Box aktif dan β_i adalah keterjadian suatu difference pair pada aktif S-Box ke-i dari suatu probabilitas karakteristik .

Daftar Pustaka

[1] E. Biham and A. Shamir, "Differential Cryptanalysis of DES-like Cryptosystems", Journal of Cryptology, vol. 4, no. 1, pp. 3-72, 1991.

[2] K. Nyberg, "Linear Approximations of Block Ciphers", Advances in Cryptology - EUROCRYPT '94 (Lecture Notes in Computer Science no. 950), Springer-Verlag, pp. 439-444, 1995.

[3] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", Advances in Cryptology - EUROCRYPT '93 (Lecture Notes in Computer Science no. 765), Springer-Verlag, pp. 386-397, 1994

[4] http://www.engr.mun.ca/~howard/Research/Papers/ldc_tutorial.html. Tanggal akses 28 September 2006.

[5] http://www.jya.com/dfa.htm. Tanggal akses 28 September 2006

[6] http://en.wikipedia..org/wiki/Linear_cryptanalis. Tanggal akses 28 oktober 2006.

[7] http://en.wikipedia..org/wiki/Differential_cryptanalis. Tanggal akses 28 oktober 2006.

[8] http://cryptome.org/cracking-des/crackingdes.htm