BAB II

LANDASAN TEORI

2.1 Cryptography (Kriptografi)

2.1.1 Definisi Cryptography

Cryptography berasal dari bahasa Yunani yang dibagi menjadi dua buah kata yaitu *crypto* yang bearti *hidden* atau *secret* (rahasia) dan *graphy* yang bearti *writing* (menulis) (Rivest Ronald L., 1990), sehingga dapat disimpulkan *cryptography* adalah ilmu yang mempelajari suatu teknik untuk melakukan komunikasi secara aman tanpa adanya gangguan dari pihak ketiga (AJ Menezes, 1996). Kriptografi secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita, selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Bruce Schneier, 2003).

Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *cryptology*. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak berhak. Dalam menjaga kerahasian data, kriptografi mentransformasikan data jelas (*plaintext*) kedalam bentuk data sandi (*ciphertext*) yang tidak dapat dikenali.

Proses transformasi dari *plaintext* menjadi *ciphertext* disebut proses *encipherment* atau enkripsi (*encryption*), sedangkan proses mentransformasikan kembali *ciphertext* menjadi *plaintext* disebut proses *decipherment* atau dekripsi (*decryption*). Suatu pesan tidak disandikan sebagai *plaintext* ataupun dapat disebut juga sebagai *cleartext*.

Kriptografi sebelum zaman modern sering kali disebut dengan enkripsi, yang merupakan suatu proses konversi informasi dari keadaan yang dapat dibaca menjadi sesuatu yang tidak dapat dibaca. Orang yang membuat pesan terenkripsi tentunya memiliki teknik dekripsi yang dibutuhkan untuk mengembalikan informasi menjadi seperti semula. Teknik dekripsi tersebut tentunya hanya diberikan pada penerima yang dituju sehingga hanya penerima yang mampu membaca pesan tersebut. Sejak Perang Dunia 1 dan munculnya komputer, metode yang digunakan untuk melakukan kriptologi menjadi semakin kompleks dengan aplikasi yang semakin luas (John Leyden, 2011).

2.1.2 Tujuan Cryptography

Empat tujuan mendasar dari ilmu kriptografi yang merupakan aspek keamanan informasi (Yusuf Kurniawan, 2004) yaitu:

1. *Data confidentiality*, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
2. *Data integrity*, berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan

untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.

3. *Authentication*, berhubungan dengan identifikasi, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan harus diauntentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. *Nonrepudiation*, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatau informasi oleh yang mengirimkan/membuat.

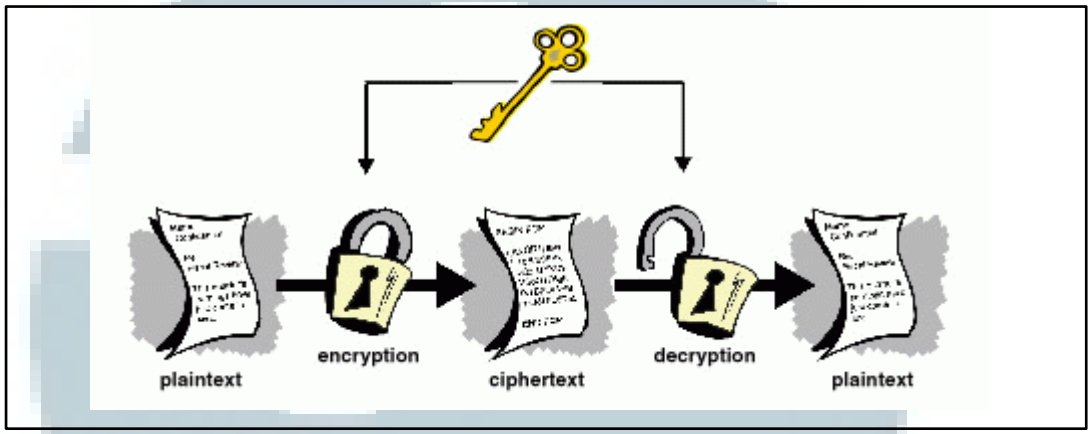
2.1.3 Jenis-Jenis Cryptography

Algoritma kriptografi dibagi menjadi tiga bagian berdasarkan kunci yang dipakainya (Dony Ariyus, 2008) yaitu:

1. Algoritma Kriptografi Kunci Simetris

   Pada algoritma kriptografi ini, kunci yang digunakan dalam proses dekripsi dan enkripsi merupakan kunci yang sama. Berdasarkan pemrosesan bit, algoritma kunci simetris dibagi menjadi dua bagian, yaitu: algoritma *cipher* blok (*block cipher*) yang melakukan pemrosesan bit per-blok, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya dan algoritma *cipher* aliran (*stream cipher*) yang memperoses blok secara mengalir atau per-bit, rangkaian bit dienkripsi dan didekripsi bit per bit. Algoritma yang memakai kunci simetris di antaranya

adalah *Data Encryption Standard* (DES), RC2, RC4, RC5, RC6, *International Data Encryption Algorithm* (IDEA), Blowfish, *One Time Pad* (OTP), A5 dan lain sebagainya. Algoritma kriptografi kunci simetris dapat digambarkan sebagai berikut.



Gambar 2.1 Algoritma Kriptografi Kunci Simetris

Sumber : http://library.thinkquest.org/C0126342/secret.htm/

(Diakses tanggal 8 Mei 2013)

Pada gambar diatas, pesan asli digambarkan dalam bentuk *plaintext*. Pesan asli mengalami proses enkripsi dengan menggunakan sebuah kunci menjadi sebuah pesan yang telah terenkripsi yang digambarkan dengan *ciphertext*. Untuk mengembalikan pesan yang telah terenkripsi atau *ciphertext* tersebut menjadi pesan asli, diperlukan kunci yang sama seperti pada waktu proses enkripsi. Proses mengembalikan pesan yang telah terenkripsi atau *ciphertext* menjadi pesan asli atau *plaintext* disebut proses dekripsi. Karena proses enkripsi dan dekripsi yang menggunakan sebuah kunci yang sama, maka algoritma kriptografi ini dinamakan algoritma kriptografi kunci simetris.

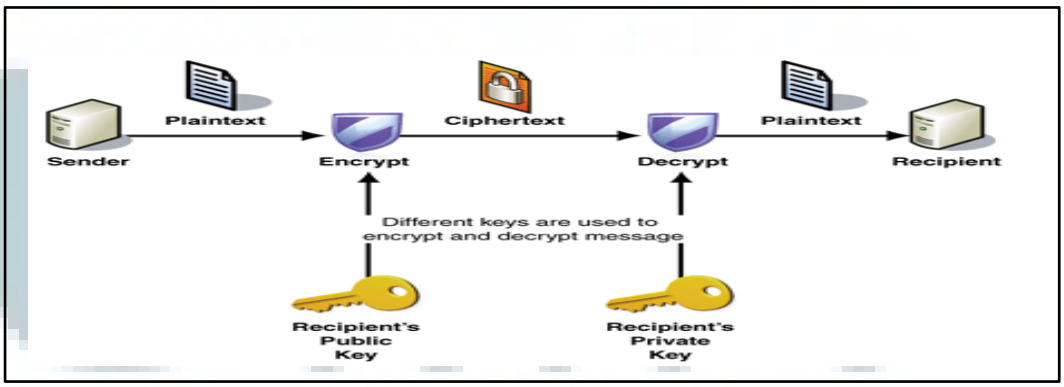
2. Algoritma Kriptografi Kunci Asimetris

Algoritma asimetris sering juga disebut dengan algoritma *public key*, dengan arti kata kunci yang digunakan untuk melakukan enkripsi dan dekripsi berbeda. Pada algoritma asimetri kunci terbagi menjadi dua bagian, yaitu:

a. Kunci umum (*public key*), yaitu kunci yang boleh semua orang tahu atau kunci yang dipublikasikan

b. Kunci rahasia (*private key*) yaitu kunci yang dirahasiakan atau hanya diketahui oleh satu orang.

Kunci-kunci tersebut berhubungan satu sama lain. Dengan kunci umum orang hanya dapat mengenkripsi pesan tetapi tidak dapat mendekripsinya. Hanya orang yang memiliki kunci rahasia yang dapat mendekripsi pesan tersebut. Algoritma yang memakai kunci simetris diantaranya adalah *Digital Signature Algoritma* (DSA), RSA, Elliptic Curve Cryptography (ECC), Kriptografi Quantum, dan lain sebagainya. Algoritma kriptografi kunci asimetris dapat digambarkan sebagai berikut.



Gambar 2.2 Algoritma Kriptografi Kunci Asimetris

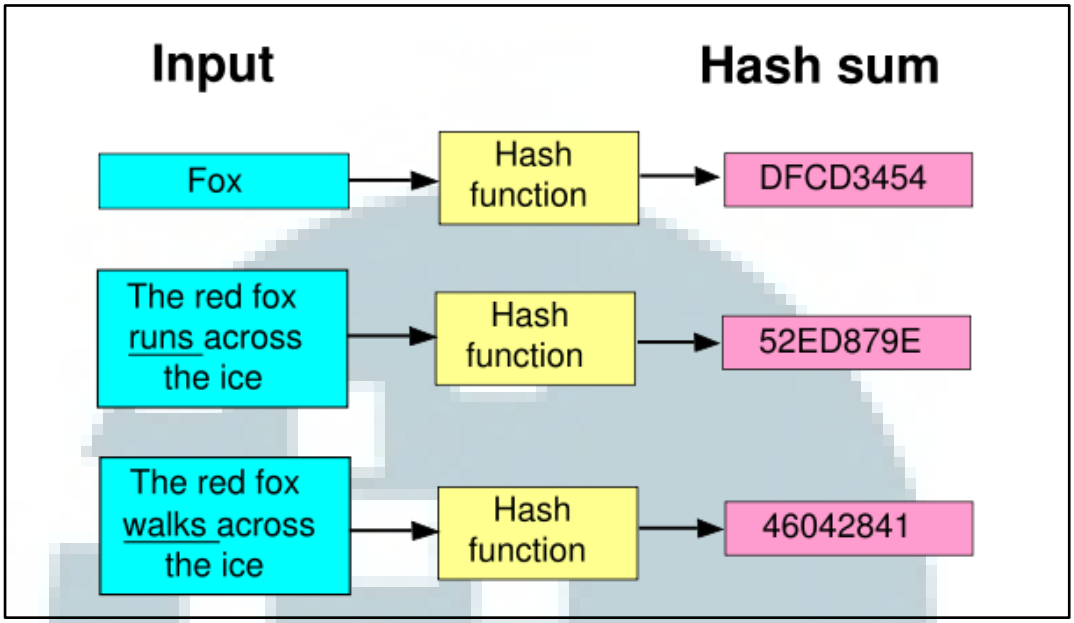
Sumber : http://msdn.microsoft.com/en-us/library/ff650720.aspx/

(Diakses tanggal 8 Mei 2013)

Pada gambar diatas, pesan asli atau *plaintext* mengalami proses enkripsi dengan menggunakan sebuah kunci umum atau *public key* oleh pengirim atau *sender*. Setelah mengalami proses enkripsi, *ciphertext* tersebut dikirimkan oleh pengirim atau *sender* kepada penerima atau *recipient*. Untuk mengembalikan *ciphertext* yang dikirimkan oleh *sender* menjadi sebuah *plaintext*, penerima atau *recipient* akan melakukan proses dekripsi dengan menggunakan sebuah kunci yang berbeda dengan kunci yang digunakan pada waktu proses enkripsi. Kunci yang digunakan pada waktu proses dekripsi tersebut dinamakan kunci rahasia atau *private key*. Karena proses enkripsi dan dekripsi yang menggunakan dua buah kunci yang berbeda, maka algoritma kriptografi ini dinamakan algoritma kriptografi kunci asimetris.

3. Funsi Hash

Fungsi Hash sering disebut dengan fungsi Hash satu arah (*one-way function*), *message diges, fingerprint,* fungsi kompresi dan *message authentication code* (MAC), merupakan satu fungsi matematika yang mengambil masukan panjang variabel dan mengubahnya ke dalam urutan biner dengan panjang yang tetap. Fungsi Hash biasanya diperlukan bila ingin membuat sidik jadi dari suatu pesan. Sidik jadi pada pesan merupakan suatu tanda bahwa pesan tersebut benar-benar berasal dari orang yang di inginkan. Fungsi hash dapat digambarkan sebagai berikut.



Gambar 2.3 Algoritma Kriptografi Fungsi Hash

Sumber : https://en.wikipedia.org/wiki/File:Hash_function.png/

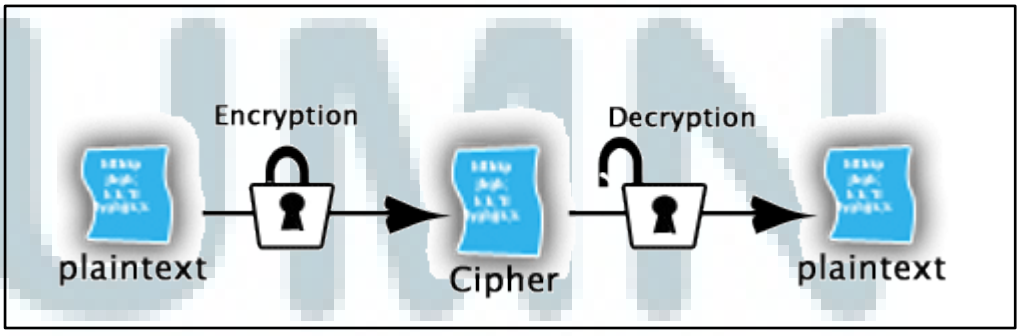
(Diakses tanggal 8 Mei 2013)

Pada gambar diatas, input yang dimasukkan akan mengalami proses fungsi hash dan menghasilkan sederetan karakter acak yang memiliki jumlah karakter yang sama. Fungsi hash berbeda dengan teknik enkripsi pada kriptografi. Tujuan fungsi hash adalah mengubah sebuah pesan yang dapat dibaca oleh manusia atau *plaintext* menjadi sebuah pesan acak yang tidak dapat dibaca oleh manusia atau *ciphertext* yang sama seperti proses enkripsi, tetapi yang menjadi perbedaan antara fungsi hash dengan proses enkripsi pada kriptografi adalah pada fungsi hash, pesan yang telah diubah menjadi *ciphertext* tidak dapat diubah kembali menjadi pesan yang dapat dibaca oleh

manusia atau *plaintext*. Inilah mengapa fungsi hash disebut juga sebagai *one-way function*.

2.1.4 Definisi Enkripsi dan Dekripsi

Encryption adalah proses menyandikan *plaintext* menjadi *ciphertext*, sedangkan proses mengembalikan *ciphertext* menjadi *plaintext* disebut dengan *decryption*. Enkripsi digunakan untuk mengamankan komunikasi diberbagai Negara untuk menjaga kerahasiaan dari pesan yang ingin disampaikan. Saat ini enkripsi telah digunakan pada sistem secara luas seperti internet *e-commerce*, jaringan telepon dan ATM pada bank. Enkripsi dapat digunakan untuk tujuan keamanan, tetapi teknik lain masih diperlukan untuk membuat komunikasi yang aman, terutama untuk memastikan integritas dan autentikasi dari sebuah pesan, contohnya *Message Authentication Code* (MAC) atau *digital signature*. Penggunaan yang lain yaitu untuk melindungi dari analisis jaringan komputer. (Goldreich, 2004). Proses enkripsi dan dekripsi dapat digambarkan sebagai berikut.



Gambar 2.4 *Encryption and Decryption*

Sumber : http://www.pwinfotech.com/2010/10/cryptographic-algorithm.html/

(Diakses tanggal 8 Mei 2013)

Pada gambar diatas, *plaintext* menyatakan sebuah pesan asli yang dapat dibaca oleh manusia. Apabila *plaintext* mengalami proses enkripsi, maka *plaintext* akan berubah menjadi *ciphertext* yang menyatakan sebuah pesan asli yang sudah mengalami proses enkripsi dan tidak dapat dibaca oleh manusia. Untuk mengembalikan *ciphertext* menjadi *plaintext* diperlukan sebuah proses dekripsi sehingga pesan tersebut dapat dibaca kembali.

2.1.5 Jenis-Jenis Cipher Simetri

Algoritma kriptografi (*cipher*) simetri dapat dikelompokkan menjadi dua kategori (Lung dan Munir, 2005), yaitu:

1. *Cipher* aliran (*stream cipher*)

   Algoritma kriptografi beroperasi pada *plaintext*/*ciphertext* dalam bentuk bit tunggal, yang dalam hal ini rangkaian bit dienkripsikan/didekripsikan bit per bit. Contoh algoritma kriptografi yang menggunakan cipher aliran adalah RC4 A5, A2, SEAL, dan lain-lain

2. *Cipher* blok (*block cipher*)

   Algoritma kriptografi beroperasi pada *plaintext*/*ciphertext* dalam bentuk blok bit, yang dalam hal ini rangkaian bit dibagi menjadi blok-blok bit yang panjangnya sudah ditentukan sebelumnya. Pada *cipher* blok, rangkaian bit-bit *plainteks* dibagi menjadi blok-blok bit dengan panjang sama, biasanya 64 bit, namun seiring dengan kemajuan teknologi, ukuran blok plaintext berkembang menjadi 128 bit, 256 bit bahkan menjadi 512 bit. Algoritma enkripsi

menghasilkan blok cipherteks yang pada kebanyakan sistem kriptografi simetri berukuran sama dengan blok plainteks. Dengan blok *cipher*, blok plainteks yang sama akan dienkripsi menjadi blok cipherteks yang sama bila digunakan kunci yang sama pula. Ini berbeda dengan *cipher* aliran dimana bit-bit plainteks yang sama akan dienkripsi menjadi bit-bit cipherteks yang berbeda setiap kali dienkripsi. Contoh algoritma kriptografi yang menggunakan *cipher* blok adalah DES, 3DES, RC5, AES, Blowfish, Twofish, Serpent, dan lain-lain. (Munir, 2006).

2.2 Algoritma Kriptografi Blowfish

2.2.1 Definisi Algoritma Blowfish

Blowfish diciptakan oleh seorang *Cryptanalyst* bernama Bruce Schneier, Presiden perusahaan Counterpane Internet Security, Inc (perusahaan konsultan tentang kriptografi dan keamanan komputer) dan dipublikasikan tahun 1994. Dibuat untuk digunakan pada komputer yang mempunyai *microprocessor* besar (32-bit keatas dengan *cache* data yang besar). Blowfish merupakan algoritma yang tidak dipatenkan dan *licensefree*, dan tersedia secara gratis untuk berbagai macam kegunaan (Syafari, 2007).

Blowfish atau disebut juga *OpenPGP.Cipher.4* adalah algoritma kunci simetrik cipher blok dengan panjang blok tetap 64 bit dan menerapkan teknik kunci berukuran sembarang serta bebas lisensi dan dirancang pada tahun 1993 oleh Bruce Schneier untuk menggantikan DES (*Data Encryption Standard*). Algoritma blowfish dibuat untuk digunakan pada komputer yang mempunyai

microprosesor besar (32-bit keatas dengan *cache* data yang besar) (Bruce Schneier, 1996).

Blowfish dirancang dan diharapkan mempunyai kriteria perancangan yang diinginkan (Bruce Schneier, 1996) sebagai berikut:

1. Cepat, Blowfish melakukan enkripsi data pada *microprocessor* 32-bit dengan *rate* 26 *clock cycles* per *byte*.
2. *Compact*, Blowfish dapat dijalankan pada *memory* kurang dari 5K.
3. Sederhana, Blowfish hanya menggunakan operasi-operasi sederhana, Blowfish hanya menggunakan operasi-operasi sederhana, seperti penambahan, XOR, dan *lookup* tabel pada operan32-bit.
4. Memiliki tingkat keamanan yang bervariasi, panjang kunci yang digunakan oleh Blowfish dapat bervariasi dan bisa sampai sepanjang minimal 32-bit, maksimal 448 -bit, *Multiple* 8 bit, *default* 128 bit

Dalam penerapannya sering kali algoritma ini menjadi tidak optimal, karena strategi implementasi yang tidak tepat. Algoritma Blowfish akan lebih optimal jika digunakan untuk aplikasi yang tidak sering berganti kunci, seperti jaringan komunikasi atau enkripsi *file* otomatis. Selain itu, karena algoritma ini membutuhkan memori yang besar, maka algoritma ini tidak dapat diterapkan untuk aplikasi yang memiliki memori kecil seperti *smart card*. Panjang kunci yang digunakan juga mempengaruhi keamanan penerapan algoritma ini.

2.2.2 Jaringan Feistel (Feistel Network)

Hampir semua algoritma *cipher* blok bekerja dalam model jaringan Feistel. Jaringan Feistel ditemukan oleh Horst Feistel tahun 1970. Model jaringan Feistel adalah sebagai berikut:

1. Bagi blok yang panjangnya *n* bit menjadi dua bagian, kiri (*L*) dan kanan (*R*), yang masing-masing panjangnya *n*/2 (hal ini mensyaratkan *n* harus genap).
2. Definisikan *cipher* blok berulang dimana hasil dari putaran ke-*i* ditentukan dari hasil putaran sebelumnya, yaitu

$$L_i = R_{i+1}$$

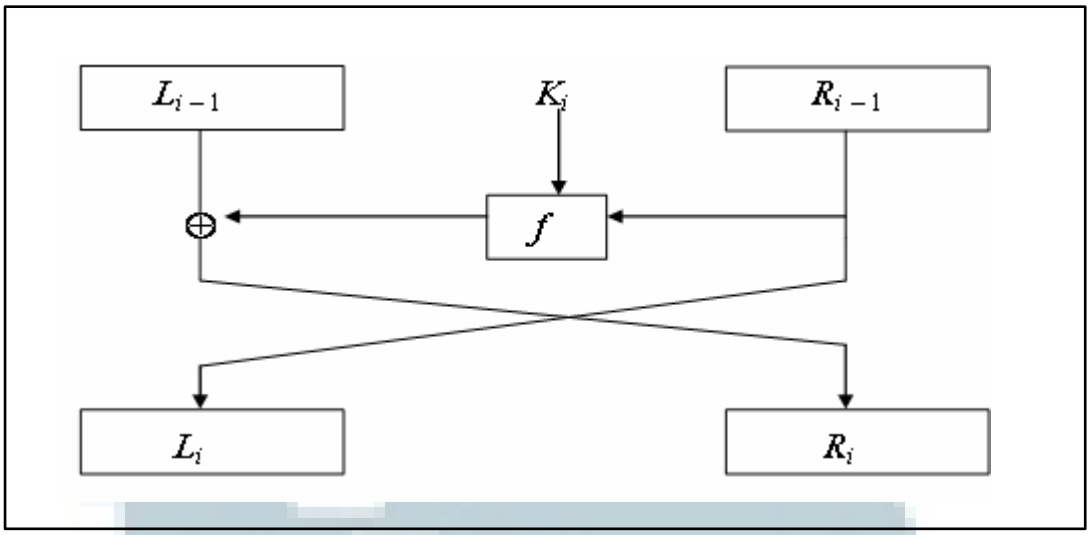
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Keterangan:

$i = 1, 2, \ldots, r$ (r adalah jumlah putaran).

K_i = subkunci (*subkey*) pada putaran ke-*i*

f = fungsi transformasi (di dalamnya terdapat fungsi substitusi, permutasi, dan/atau ekspansi, kompresi).



Gambar 2.5 Jaringan Feitsel (*Feitsel Network*)

(Sumber : Patarin, 2003)

Plainteks adalah gabungan *L* dan *R* awal, atau secara formal dinyatakan dengan (L_0, R_0), sedangkan *cipherteks* didapatkan dari *L* dan *R* hasil dari putaran terakhir setelah terlebih dauhulu dipertukarkan, atau secara formal dinyatakan sebagai (R_r, L_r).

Jaringan Feistel banyak dipakai pada algoritma kriptografi DES (*Data Encryption Standard*), LOKI, GOST, FEAL, Lucifer, Blowfish, Khufu, Khafre, dan lain-lain karena model ini bersifat *reversible* untuk proses enkripsi dan dekripsi. Sifat *reversible* ini membuat kita tidak perlu membuat algoritma baru unruk mendekripsi *cipherteks* menjadi *plainteks*. Karena operator XOR mengkombinasikan setengah bagian kiri dengan hasil dari fungsi transformasi *f*, maka dihasilkan persamaan sebagai berikut:

$$L_{i-1} \oplus f(R_{i-1}, K_i) \oplus f(R_{i-1}, K_i) = L_{i-1}$$

Sifat *reversible* tidak bergantung pada fungsi *f* sehingga fungsi *f* dapat dibuat serumit mungkin.

2.2.3 Kotak-S (S-box)

Kotak-*S* adalah matriks yang berisi substitusi sederhana yang memetakan satu atau lebih bit dengan satu atau lebih bit yang lain. Pada kebanyakan algoritma *cipher* blok, kotak-*S* memetakan *m* bit masukan menjadi *n* bit keluaran, sehingga kotak-*S* tersebut dinamakan kotak $m \times n$ *S-box*. Kotak-*S* merupakan satu-satunya langkah nirlanjar di dalam algoritma, karena operasinya adalah *look-up table*. Masukan dari operasi *look-up table* dijadikan sebagai indeks kotak-*S*, dan keluarannya adalah *entry* di dalam kotak-*S*.

Sebagai contoh, Kotak-*S* di dalam algoritma DES adalah 6×4 *S-box* yang berarti memetakan enam bit masukan menjadi empat bit keluaran. Salah satu kotak-*S* yang ada di dalam algoritma DES adalah sebagai berikut:

Tabel 2.1 Salah Satu Kotak-S pada Algoritma DES

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Baris diberi nomor dari 0 sampai 3.

Kolom diberi nomor dari 0 sampai 15.

Masukan untuk proses substitusi adalah enam bit, $b_1b_2b_3b_4b_5b_6$.

Nomor baris dari tabel ditunjukkan oleh *string* bit b_1b_6 (menyatakan 0 sampai 3 desimal). Nomor kolom ditunjukkan oleh *string* bit $b_2b_3b_4b_5$ (menyatakan 0 sampai 15). Misalkan masukan adalah 110100, nomor baris tabel = 10 (artinya baris dua desimal), nomor kolom tabel = 1010 (artinya kolom 10 desimal). Jadi, substitusi untuk 110100 adalah *entry* pada baris dua dan kolom 10, yaitu 0100 (atau empat desimal). Perancangan kotak-S menjadi isu penting karena kotak-S harus dirancang sedemikian sehingga kekuatan kriptografinya bagus dan mudah diimplementasikan. Ada empat cara (pendekatan) yang dapat digunakan dalam mengisi kotak-*S*:

1. Dipilih secara acak

   Untuk kotak-*S* yang kecil, cara pengisian secara acak tidak aman, namun untuk kotak-*S* yang besar cara ini cukup bagus.

2. Dipilih secara acak lalu diuji

   Sama seperti cara nomor satu, namun nilai acak yang dibangkitkan diuji apakah memenuhi sifat tertentu.

3. Dibuat oleh orang (*man-made*)

   Entry di dalam kotak-*S* dibangkitkan dengan teknik yang lebih intuitif.

4. Dihitung secara matematis (*math-made*)

   Entry di dalam kotak-*S* dibangkitkan berdasarkan prinsip matematika yang terbukti aman dari serangan kriptanalis.

2.2.4 Struktur Algoritma Blowfish

Struktur dari Algoritma Blowfish terdiri atas dua buah bagian (Schneier, 1996), yaitu:

1. *Key-expansion* (ekspansi kunci)

   Berfungsi merubah kunci (minimum 32-bit, maksimum 448-bit) menjadi beberapa *array* subkunci (*subkey*) dengan total 4168 *byte* (18x32-bit untuk P-*array* dan 4x256x32-bit untuk S-*box* sehingga totalnya 33344 bit atau 4168 *byte*). Kunci disimpan dalam K-*array*:

   K1, K2, …Kj 1 ≤ j ≤ 14

   Kunci-kunci ini yang dibangkitkan (*generate*) dengan menggunakan subkunci yang harus dihitung terlebih dahulu sebelum enkripsi atau dekripsi data. Sub-sub kunci yang digunakan terdiri dari:

   P-*array* yang terdiri dari 18 buah 32-bit subkunci,

   P1, P2, …, P18

   S-*box* yang terdiri dari empat buah 32-bit, masing-masing memiliki 256 entri:

   S1, 0, S1, 1, …, S1, 255

   S2, 0, S2, 1, …, S2, 255

   S3, 0, S3, 1, …, S3, 255

   S4, 0, S3, 1, …, S4, 255

   Langkah-langkah perhitungan atau pembangkitan subkunci tersebut adalah sebagai berikut:

   a. Inisialisasi P-*array* yang pertama dan juga empat S-*box*, berututan, dengan *string* yang telah pasti.

String tersebut terdiri dari digit-digit heksadesimal dari *phi*, tidak termasuk angka tiga di awal.

Contoh :

P1= 0x243f6a88

P2= 0x85a308d3

P3= 0x13198a2e

P4= 0x03707344

dan seterusnya sampai S-*box* yang terakhir.

b. XOR-kan P1 dengan 32-bit awal kunci, XOR-kan P2 dengan 32-bit berikutnya dari kunci, dan seterusnya untuk semua bit kunci. Ulangi siklus seluruh bit kunci secara berurutan sampai seluruh P-*array* ter-XOR-kan dengan bit-bit kunci. Atau jika disimbolkan : P1 = P1 _ K1, P2 = P2 _K2, P3 = P3 _ K3, . . . P14 = P14 _ K14, P15 = P15 _ K1, . . . P18 = P18 _ K4. Keterangan : _ adalah simbol untuk XOR.
c. Enkripsikan *string* yang seluruhnya nol (*all-zero string*) dengan algoritma Blowfish, menggunakan subkunci yang telah dideskripsikan pada langkah satu dan dua.
d. Gantikan P1 dan P2 dengan keluaran dari langkah 3.
e. Enkripsikan keluaran langkah 3 menggunakan algoritma Blowfish dengan subkunci yang telah dimodifikasi.
f. Gantikan P3 dan P4 dengan keluaran dari langkah 5.
g. Lanjutkan langkah-langkah di atas, gantikan seluruh elemen P-*array* dan kemudian keempat S-*box* secara berurutan, dengan hasil keluaran

algoritma Blowfish yang terus-menerus berubah. Total keseluruhan, terdapat 521 iterasi untuk menghasilkan subkunci-subkunci dan membutuhkan memori sebesar 4KB.

2. Enkripsi data

   Terdiri dari iterasi fungsi sederhana (*Feistel Network*) sebanyak 16 kali putaran (iterasi), masukannya adalah 64-bit elemen data X. Setiap putaran terdiri dari permutasi kunci dependent dan substitusi kunci dan data dependent. Semua operasi adalah penambahan (*addition*) dan XOR pada variabel 32-bit. Operasi tambahan lainnya hanyalah empat penelusuran tabel *array* berindeks untuk setiap putaran. Langkahnya adalah seperti berikut:

   a. Bagi X menjadi dua bagian yang masing-masing terdiri dari 32-bit: XL, XR.
   b. Lakukan langkah berikut

      For i = 1 to 16:

      XL = XL XOR Pi

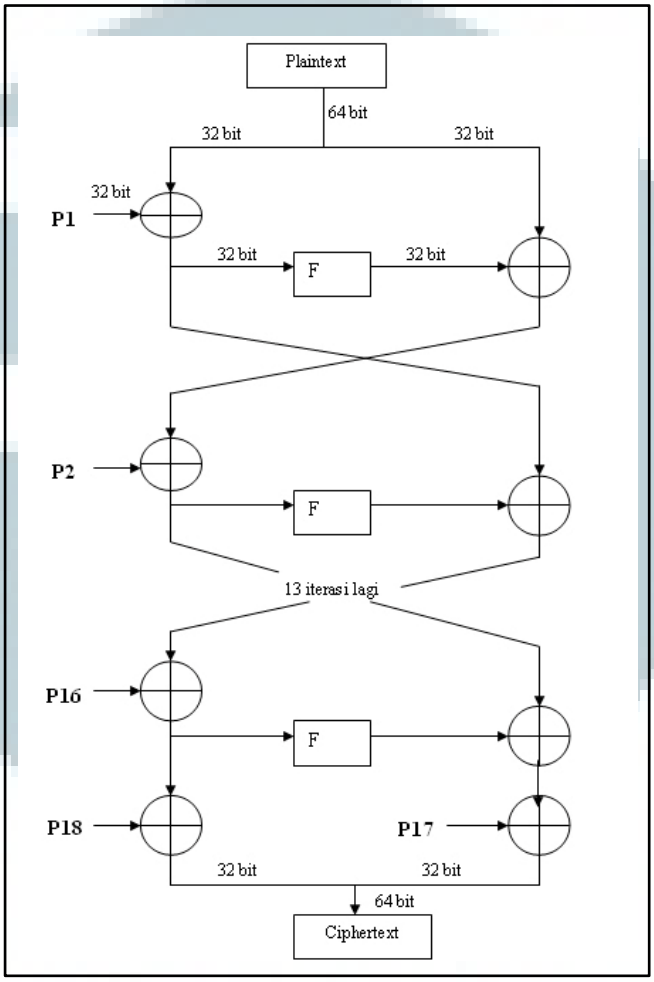
      XR = F(XL) XOR XR

      Tukar XL dan XR
   c. Setelah iterasi ke-16, tukar XL dan XR lagi untuk melakukan membatalkan pertukaran terakhir.
   d. Lalu lakukan

      XR = XR XOR P17

      XL = XL XOR P18
   e. Terakhir, gabungkan kembali XL dan XR untuk mendapatkan cipherteks.

Untuk mempermudah pemahaman, gambaran tahapan pada jaringan feistel yang digunakan algoritma Blowfish adalah seperti pada gambar berikut.



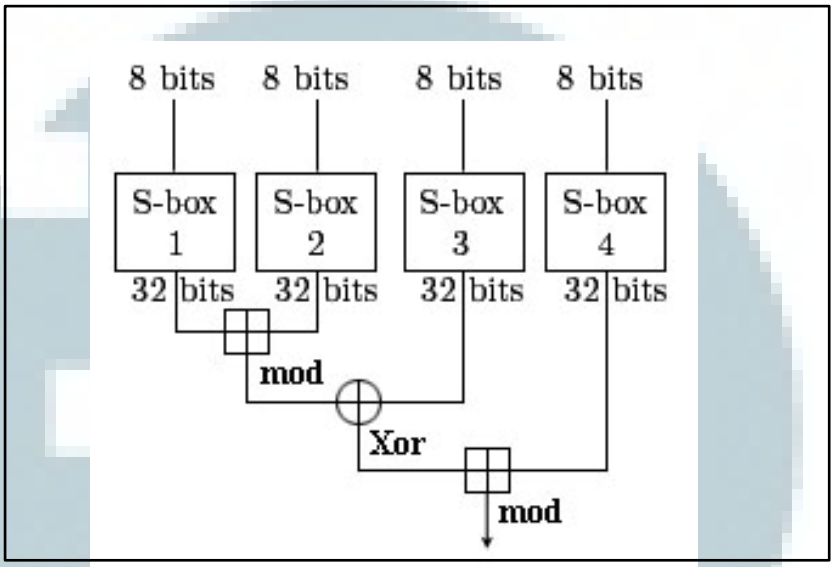
Gambar 2.6 Blok Diagram Algoritma Blowfish

(Sumber : Schneier, 1996)

Pada langkah kedua, telah dituliskan mengenai penggunaan fungsi F. Fungsi F adalah bagi XL menjadi empat bagian 8-bit: a,b,c dan d.

$$\mathbf{F(XL) = ((S1,a + S2,b \bmod 2^{32}) \text{ XOR } S3,c) + S4,d \bmod 2^{32}}$$

Agar dapat lebih memahami fungsi F, tahapannya dapat dilihat pada gambar berikut ini.



Gambar 2.7 Fungsi F pada Algoritma Blowfish

(Sumber : Schneier, 1996)

Pada Algoritma Blowfish, proses dekripsi memiliki proses yang sama dengan enkripsi, hanya saja P1, P2, …, P18 digunakan pada urutan yang berbalik (*reverse*). Algoritmanya dapat dinyatakan sebagai berikut (Schneier, 1996):

for i = 1 to 16 do

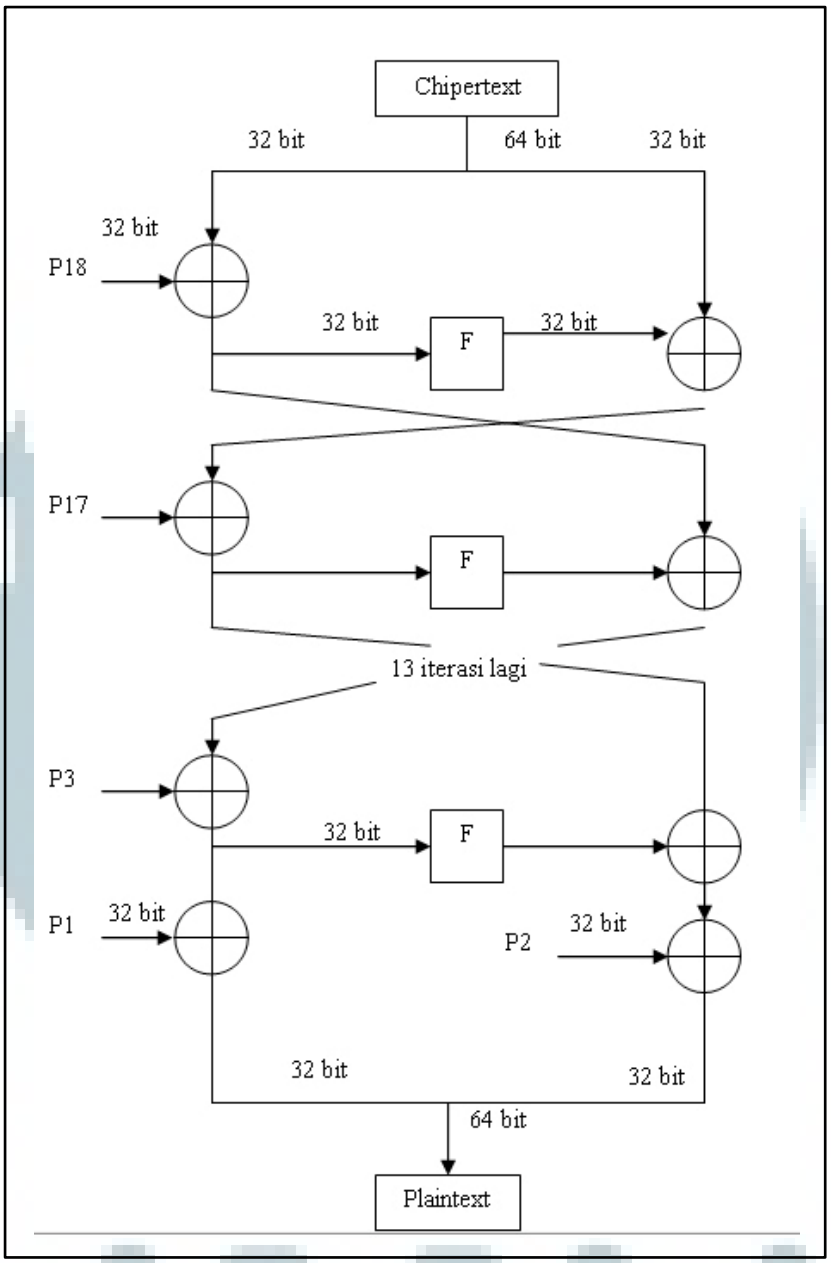
XRi = XLi-1 XOR P19-i;

XLi = F[XRi] XOR XRi-1;

XL17 = XR16 XOR P1;

XR17 = XL16 XOR P2;

Untuk lebih jelasnya dapat digambarkan pada gambar berikut ini.



Gambar 2.9 Blok Diagram Dekripsi Blowfish

(Sumber : Schneier, 1996)

2.3 MPEG-1 Audio Layer 3 (MP3)

MPEG-1 *Audio Layer* III atau biasa disebut dengan MP3 adalah salah satu berkas pengodean suara yang mempunyai kompresi yang baik (*lossy*) sehingga ukuran berkas menjadi lebih kecil. MP3 merupakan format audio yang umum dan telah menjadi standar kompresi audio digital untuk proses memutar musik pada pemutar audio digital. MP3 adalah format audio khusus yang dirancang oleh *Moving Picture Experts Group* (MPEG) sebagai bagian dari MPEG-1 dan kemudian dikembangkan menjadi MPEG-2. MP3 dikembangkan oleh seorang insinyur Jerman bernama Karlheinz Brandenburg. MP3 menggunakan algoritma kompresi *lossy* dengan tujuan untuk mengurangi sebagian besar data yang mewakili suatu audio akan tetapi menghasilkan audio yang sama dengan sebelum terkompresi, karena algoritma tersebut hanya mengilangkan komponen-komponen suara yang tidak terdengar oleh manusia. (Ewing, 2007)

2.4 Database (Basis Data)

2.4.1 Definisi Database

Secara etimologis basis data terdiri dari dua kata yaitu basis dan data yang dapat diartikan sebagai markas atau gudang, tempat bersarang atau berkumpul. Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang dan sebagainya yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, atau kombinasinya. Menurut pengertian lain Basis data (*database*) merupakan sekumpulan data yang saling terintegrasi satu sama lain

dan terorganisasi berdasarkan sebuah skema atau struktur tertentu dan tersimpan pada sebuah *hardware* komputer (M.Rudyanto Arief, 2005).

2.4.2 Database Management System (DBMS)

Database management system (DBMS) adalah konsep basis data yang menyimpan semua data dalam bentuk table-tabel. Sebuah tabel menyimpan informasi mengenai sebuah subjek tertentu. Dengan DBMS sebuah basis data akan dengan mudah dikelola walaupun jumlah data banyak dan sangat kompleks. Pada prinsipnya DBMS terbagi menjadi tiga data (Jeffrey, 1997), yaitu :

1. *Data Definition*

   Mendefinisikan jenis data yang akan dibuat, cara relasi data, validasi data dan lainnya.

2. *Data Manipulation*

   Data yang telah dibuat dan didefinisikan akan dilakukan beberapa pengerjaan, seperti *sharing data*, proses *query* dan lain sebagainya.

3. *Data control*

   Pada bagian ini berhubungan dengan cara mengendalikan data, seperti siapa yang dapat melihat isi data, bagaimana data bisa digunakan oleh banyak *user* dan sebagainya.

2.4.3 Structure Query Language

Structure Query Language (SQL) adalah salah satu bahasa generasi level ke-4 (4th GL) yang awalnya dikembangkan oleh IBM di *San Jose Research Laboratory*. SQL merupakan bahasa basis data relasional yang terdiri atas sekumpulan perintah untuk mendefinisikan, manipulasi dan mengontrol data. SQL sendiri terbagi atas dua bagian (Alan, 2009), yaitu:

1. *Data Definition Language* (DDL)

   DDL adalah bahasa yang memiliki kemampuan untuk mendefinisikan data yang berhubungan dengan pembuatan dan penghapusan objek seperti tabel, indeks, bahkan basis datanya sendiri. Misal *CREATE*, *DROP* dan *ALTER*.

2. *Data Manipulation Language* (DML)

   DML adalah bahasa yang berhubungan dengan proses manipulasi data pada tabel, *record*. Misalnya, *INSERT*, *UPDATE*, *SELECT*, dan *UPDATE*. DML terbagi atas dua tipe, yaitu:

   a. DML *prosedural*, dimana pengguan harus menspesifikasikan data apa yang dibutuhkan dan bagaimana mendapatkan data tersebut.

   b. DML deklaratif atau *non-prosedural* DML, dimana pengguna hanya menspesifikasikan data yang dibutuhkan tanpa menspesifikasikan bagaimana cara mendapatkan data itu. DML jenis ini adalah DML yang secara umum dikenal, contohnya adalah *SQL Language*.

2.5 Serangan Terhadap Kriptografi

2.5.1 Jenis-Jenis Serangan

Serangan-serangan terhadap kriptografi dapat dikelompokkan dengan beberapa cara (Munir, 2006), yaitu:

1. Berdasarkan keterlibatan penyeran dalam komunikasi, serangan dapat dibagi menjadi dua macam, yaitu:
   a. Serangan pasif (*passive attack*)

      Pada serangan ini, penyerang tidak terlibat dalam komunikasi antara pengirim dan penerima, namun penyerang menyadap semua pertukaran pesan antara kedua entitas tersebut. Tujuannya adalah untuk mendapatkan sebanyak mungkin informasi yang digunakan untuk kriptanalisis. Beberapa metode penyadapan antara lain:

      i. *Wiretapping*, adalah penyadap mencegat data yang ditransmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras.

      ii. *Electro eavesdropping*, adalah penyadap mencegat data yang ditransmisikan melalui saluran wireless, misalnya radio dan microwave.

      iii. *Acoustic eavesdropping*, adalah menangkap gelombang suara yang dihasilkan oleh suara manusia.

   b. Serangan aktif (*active attack*)

      Pada jenis serangan ini, penyerang mengintervensi komunikasi dengan ikut mempengaruhi sistem untuk keuntungan dirinya. Misalnya

penyerang mengubah aliran pesan seperti menghapus sebagian *ciphertext*, mengubah *ciphertext*, menyisipkan potongan *ciphertext* palsu, mengulangi pesan lama, mengubah informasi yang tersimpan, dan sebagainya.

2. Berdasarkan banyaknya informasi yang diketahui oleh kriptanalis, maka serangan dapat dikelompokkan menjadi lima jenis, yaitu:

   a. *Ciphertext-only attack*

      Ini adalah jenis serangan yang paling umum namun paling sulit, karena informasi yang tersedia hanyalah *ciphertext* saja. Kriptanalis memiliki beberapa *ciphertext* dari beberapa pesan, semuanya dienkripsi dengan algoritma yang sama, untuk itu kriptanalisis menggunakan beberapa cara seperti mencoba semua kemungkinan kunci secara *exhaustive search*. Menggunakan analisis frekuensi, membuat terkaan berdasarkan informasi yang diketahui, dan sebagainya.

   b. *Known-plaintext attack*

      Ini adalah jenis serangan dimana kriptanalis memiliki pasangan *plainteks* dan *cipherteks* yang berkoresponden.

   c. *Chosen-plaintext attack*

      Serangan jenis ini lebih hebat dari pada *known-plaintext attack*, karena kriptanalis dapat memilih *plainteks* yang dimilikinya untuk dienkripsikan, yaitu *plainteks-plainteks* yang lebih mengarahkan penemuan kunci.

d. *Chosen-ciphertext attack*

Ini adalah jenis serangan dimana kriptanalis memilih *cipherteks* untuk didekripsikan dan memiliki akses ke *plainteks* hasil dekripsi.

e. *Chosen-text attack*

Ini adalah jenis serangan yang merupakan kombinasi *chosen-plaintext attack* dan *chosen-chiphertext attack*.

3. Berdasarkan teknik yang digunakan dalam menemukan kunci, maka serangan dapat dibagi menjadi empat, yaitu:

a. *Exhaustive attack* atau *brute force attack*

Ini adalah serangan untuk mengungkap *plainteks* atau kunci dengan menggunakan semua kemungkinan kunci. Diasumsikan kriptanalis mengetahui algoritma kriptografi yang digunakan oleh pengirim pesan. Selain itu kriptanalis memiliki sejumlah *cipherteks* dan *plainteks* yang bersesuaian.

b. *Analytical attack*

Pada jenis serangan ini, kriptanalis tidak mencoba-coba semua kemungkinan kunci tetapi menganalisis kelemahan algoirtma kriptografi untuk mengurangi kemungkinan kunci yang tidak ada. Diasumsikan kriptanalis mengetahui algoritma kriptografi yang digunakan oleh pengirim pesan. Analisis dapat menggunakan pendekatan matematik dan statistik dalam rangka menemukan kunci.

c. *Related-key attack*

Kriptanalis memiliki cipherteks yang dienkripsi dengan dua kunci berbeda. Kriptanalis tidak mengetahui kedua kunci tersebut namun ia mengetahui hubungan antara kedua kunci, misalnya mengetahui kedua kunci hanya berbeda 1 bit.

d. *Rubber-hose cryptanalysis*

Ini mungkin jenis serangan yang paling ekstrim dan paling efektif, dimana peenyerang melakukan tindakan mengancam, mengirim surat gelap, atau melakukan penyiksaan sampai orang yang memegang kunci memberinya kunci untuk mendekripsi pesan.