BAB II

LANDASAN TEORI

2.1 Keamanan jaringan Komputer

Keamanan data atau informasi adalah sesuatu hal yang sangat penting dalam suatu jaringan komputer, karena pada suatu jaringan komputer memungkinkan kita untuk dapat saling bertukar data atau informasi. Untuk menjaga keamanan dalam pertukaran data atau informasi maka dibutuhkannya suatu sistem keamanan yang dapat menjaga keutuhan dari data tersebut dari serangan atau gangguan dari pihak – pihak yang tidak berhak untuk mengakses data atau informasi tersebut. Dalam hal ini keamanan jaringan yang sedang dibahas merupakan jaringan komputer *Local Area Network* (LAN), dengan menggunakan teknik keamanan kriptografi.

2.1.1 *Local Area Network* (LAN)

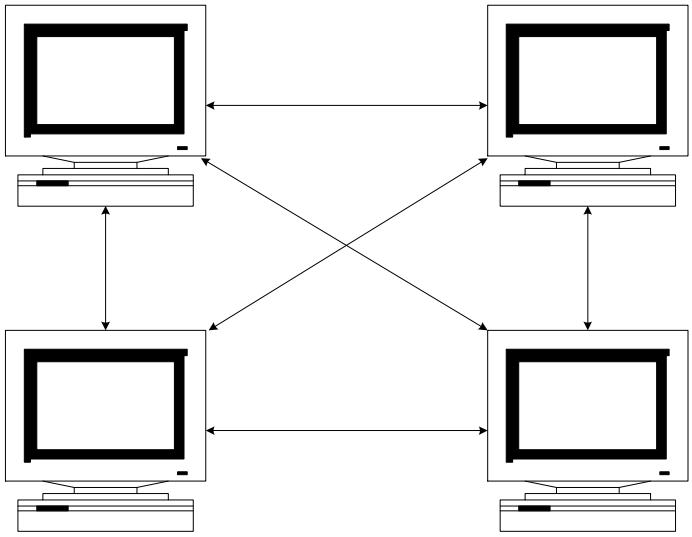
Kebutuhan akan komunikasi untuk komputer muncul pada tahun 1940-an dan 1950-an yang merupakan konsep dasar dari sebuah jaringan computer. Tahun 1960-an lahir konsep *timesharing,* dimana pengguna dihubungkan kekomputer melalui suatu *dumb terminal.*

Pada tahun 1970-an, teknologi IC (*Integrated Circuit*) dan mikroprosesor mulai berkembang sehingga memungkinkan munculnya komputer pribadi yang dapat dipasang di rumah-rumah. Dengan adanya perkembangan teknologi tersebut dapat mengubah cara pandang orang terhadap komputer. Tahun 1980-an muncul

teknologi jaringan lokal *(Local Area Network*-LAN*)* yang memiliki kemampuan berkomunikasi dengan komputer lainnya.

2.1.2 Model Hubungan *Peer-to-Peer*

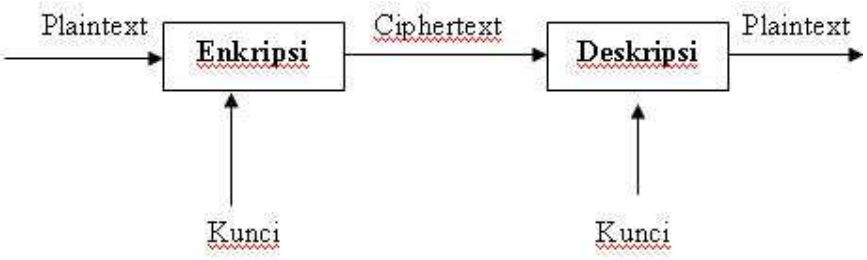
Model jaringan komputer *Peer-to-Peer* merupakan model jaringan yang tidak memiliki sebuah *file server* atau sumber daya yang terpusat. Didalam hubungan *Peer-to-Peer* ini, seluruh komputer adalah sama, yaitu mempunyai kemampuan yang sama untuk memakai sumber daya yang tersedia didalam jaringan (**Budhi Irawan** – Jaringan Komputer). Berikut dibawah ini pada gambar 2.1 merupakan gambaranm dari model hubungan jaringan *Peer-to-Peer* :



Gambar 2.1 Model *Peer-to-Peer*

2.2 Teori Dasar Kriptografi

Kriptografi dalam sejarahnya tercatat telah dipergunakan oleh bangsa Mesir 4000 tahun lalu. Kriptografi (*Cryptography*) berasal dari dua kata yaitu "*Crypto & graphy*" yang dalam sudut bahasa "*Crypto*" dapat diartikan rahasia (*secret*) dan *"graphy"* dapat diartikan tulisan (*writing*) jadi Kriptografi (*Cryptography*) dapat diartikan sebagai suatu ilmu atau seni untuk mengamankan pesan agar aman dan dilakukan oleh *"Cryptographer"*. Secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita [**bruce Schneier** - *Applied Cryptography*]. 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 [A. Menezes, P. van Oorschot and S. Vanstone - *Handbook of Applied Cryptography*]. Orang yang melakukan enkripsi terhadap suatu pesan atau praktisi kriptogragi disebut *"Cryptographer"*. Sebuah pesan yang tidak disandikan atau dienkripsi disebut sebagai *plaintext* atau disebut juga sebagai *cleartext.* Sedangkan pesan yang telah disandikan dengan sebuah algoritma kriptografi disebut sebagai *ciphertext.* Proses untuk mengubah *plaintext* ke *chipertext* disebut *encryption* atau *encipherment*. Sedang proses mengubah *chipertext* ke *plaintext* disebut *decryption* atau *decipherment*. Pada gambar 2.2 digambarkan diagram proses enkripsi dan dekripsi, dengan gambar sebagai berikut :



Gambar 2.2 diagram proses enkripsi dan dekripsi

Fasilitas untuk mengkonversikan sebuah *plaintext* ke *ciphertext* atau sebaliknya disebut *Cryptographic system* atau *Cryptosystem* dimana sistem tesebut terdiri dari algoritma–algorima tertentu yang tergantung pada sistem yang digunakan. Algoritma kriptografi (*cryptographic algorithm*) disebut *cipher* yang merupakan persamaan matematik yang digunakan dalam proses enkripsi dan deskripsi dimana proses tersebut diatur oleh satu atau lebih kunci kriptografi. Kunci-kunci tersebut secara umum digunakan untuk proses pengenkripsian dan pendekripsian tidak perlu identik, tergantung sistem yang digunakan. Dengan kata lain tugas utama dalam kriptografi adalah untuk menjaga *plaintext* (data asli) maupun *key* (kunci) tetap terjaga kerahasiaannya.

Proses enkripsi dan deskripsi secara matematis diterangkan sebagai berikut :

$$EK\ (M) = C \text{ (Proses Enkripsi)} \quad (2.1)$$

$$DK\ (C) = M \text{ (Proses Deskripsi)} \quad (2.2)$$

Keterangan :

EK : Enkripsi.

DK : Dekripsi.

M : *Message* (Pesan sebelum dienkripsi).

C : *Cipher* (Pesan setelah dienkripsi).

Berikut adalah istilah – istilah yang digunakan dalam ilmu kriptografi :

a. *Plaintext* adalah pesan asli yang akan dienkripsi yang biasa disimbolkan dengan (M).
b. *Chipertext* adalah sebuah pesan yang telah dienkripsi atau disandikan dan disimbolkan dengan (C).
c. Enkripsi adalah proses untuk merubah *plaintext* menjadi *chipertext* disimbolkan dengan (E).
d. Dekripsi adalah proses untuk merubah *chipertext* menjadi *plaintext*, atau merubah sebuah data yang tersandikan menjadi data asli. Disimbolkan dengan (D).
e. *Key /* kunci adalah sebuah bilangan yang dirahasiakan dalam suatu proses enkripsi dan dekripsi.

Secara matematis, proses enkripsi merupakan pengoperasian fungsi E (enkripsi) menggunakan e (kunci enkripsi) pada M *(plaintext)* sehingga dihasilkan C *(ciphertext),* notasinya :

$$E_e(M) - C \qquad (2.3)$$

Sedangkan untuk proses dekripsi, merupakan pengoperasian fungsi D (dekripsi) menggunakan d (kunci dekripsi) pada C *(ciphertext)* sehingga dihasilkan M *(plaintext),* notasinya :

$$D_d(C) = M \qquad (2.4)$$

Sehingga dari dua hubungan diatas berlaku :

$$D_d(E_e(M)) = M \qquad (2.5)$$

Tujuan dari adanya enkripsi adalah untuk meningkatkan keamanan data tetapi juga berfungsi untuk :

1. Melindungi data agar tidak dapat dibaca oleh orang-orang yang tidak berhak.
2. Mencegah agar orang-orang yang tidak berhak, menyisipkan atau mengahapus data.

Sedangkan tujuan dari sistem kriptografi adalah sebagai berikut :

1. *Confidentiality*

   Memberikan kerahasiahan pesan dan menyimpan data dengan menyembunyikan informasi lewat teknik-teknik enkripsi.

2. *Message Integrity*

   Memberikan jaminan untuk tiap bagian bahwa pesan tidak akan mengalami perubahan dari saat ia dibuat sampai saat ia dibuka.

3. *Non-repudiation*

   Memberikan cara untuk membuktikan bahwa suatu dokumen datang dari seseorang apabila ia mencoba menyangkal memiliki dokumen tersebut.

4. *Authentication*

   Memberikan dua layanan. Pertama mengidentitifikasi keaslian suatu pesan dan memberikan jaminan keontentikannya. Kedua untuk menguji identitas seseorang apabila ia akan memasuki sebuah sistem.

Terdapat tiga kategori enkripsi yaitu :

1. Kunci enkripsi rahasia / *private*, dalam hal ini terdapat sebuah kunci yang digunakan untuk mengenkripsi dan juga sekaligus mendeskripsikan informasi.
2. Kunci enkripsi *public*, dalam hal ini terdapat dua kunci yang digunakan, satu untuk proses enkripsi, satu lagi untuk proses deskripsi.
3. Fungsi *one-way*, dimana informasi dienkripsi untuk menciptakan "*signature*" dari informasi asli yang bisa digunakan untuk keperluan autentifikasi.

Dalam *Cryptosystem* menurut teknik enkripsinya dapat digolongkan menjadi dua buah, yaitu :

1. *Symmetric Cryptosystem* (Enkripsi Konvensional)

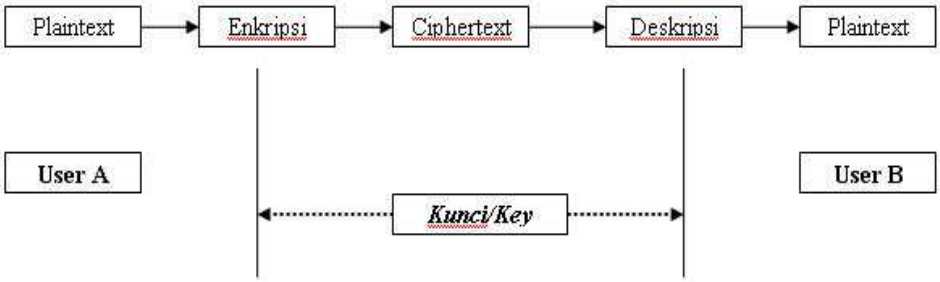
   Dalam *symmetric cryptosystem*, kunci yang digunakan dalam proses enkripsi dan dekripsi adalah sama atau pada prinsipnya identik. Kunci ini pun bisa diturunkan dari kunci lainnya. Oleh karena itu sistem ini sering disebut *secret-key ciphersystem.*

   Jumlah kunci yang dibutuhkan umumnya adalah :

$$ {}_nC_2 = \frac{n.(n-1)}{2} \qquad (2.6) $$

   Dimana n adalah banyaknya pengguna.

   Kunci yang menggunakan teknik enkripsi ini harus betul-betul dirahasiakan, berikut dibawah ini merupakan penjelasan dari proses enkripsi konvensional yang digambarkan pada gambar 2.3 :



Gambar 2.3 proses enkripsi konvensional

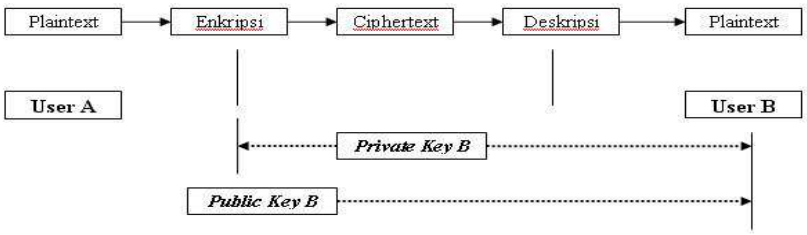
2. *Assymmetric Cryptosystem* (Enkripsi *public-key*)

   Dalam Assymmetric cryptosystem, kunci yang digunakan terdapat dua buah. Satu kunci yang dapat dipublikasikan disebut kunci publik (*public key*), satu lagi kunci yang harus dirahasiakan disebut kunci privat (*private key*).

   Secara sedehana proses tersebut diterangkan sebagai berikut :

   a. A mengirimkan pesan kepada B.
   b. A menyandikan pesannya dengan menggunakan kunci publik B.
   c. Bila B ingin membaca pesan dari A, ia harus menggunakan kunci privat-nya untuk mendekripsikan pesan yang tersandikan itu.

Dibawah ini merupakan penjelasan dari proses enkripsi *public key* pada gambar 2.4 berikut ini :



Gambar 2.4 proses enkripsi *public key*

2.2.1 Kriptanalisis

Cryptanalysis atau analisis sandi adalah ilmu untuk mendapatkan *plaintext* pesan tanpa harus mengetahui kunci secara wajar [**Yusuf kurniawan** – kriptografi keamanan internet dan jaringan komunikasi]. Pemecahan sandi rahasia yang berhasil akan menghasilkan *plaintext* atau kunci. Analisis sandi juga dapat menemukan kelemahan dalam kriptosistem, dengan kata lain analisis sandi merupakan kebalikan dari kriptografi. Usaha analisis sandi disebut juga dengan *attack* (serangan). Asumsi dasar dalam analisis sandi diungkapkan pertama kali oleh Dutchman A Kerckhoffs pada abad ke – 19, yaitu bahwa kerahasiaan harus terletak pada kunci. Kerckhoffs mengasumsikan bahwa analisis sandi mempunyai detil lengkap algoritma kriptografi dan implementasinya.

Lars Knudsen menggolongkan berbagai macam jenis pemecahan algoritma :

1. *Total break,* seorang analis berhasil menemukan kunci, K yang digunakan untuk melindungi data – data, sedemikian sehingga $D_k(C) = P$.
2. *Global deduction,* analis sandi mendapatkan algoritma alternative, A, yang ekivalen dengan $D_k(C)$, tanpa mengetahui K.
3. *Instance (local)deduction,* analis sandi mendapatkan *plaintext* atau *chipertext* yang disadap.
4. *Information deduction,* analis sandi memperoleh beberapa informasi mengenai kunci atau *plaintext*.

Terdapat beberapa cara untuk mengukur kompleksitas serangan, yaitu :

1. Data *Complexity*, yaitu jumlah data yang diperlukan sebagai input *attack*.
2. *Processing Complexity*, lama waktu yang tersedia untuk melakukan *attack*.
3. *Storage Requipments*, jumlah memori yang dibutuhkan untuk melakukan *attack*.

2.3 Kriptografi Klasik Dan Kriptografi *Modern*

2.3.1 Kriptografi klasik

Kriptografi klasik merupakan kriptografi yang berbasis karakter, sebutan kriptografi klasik merupakan kriptografi yang dipakai pada jaman romawi. Sejak jama romawi yang dipimpin oleh Yulius Caesar telah menggunakan teknik kriptografi ini yang sekarang dianggap kuno dan sangat mudah untuk dibobol. Dalam kriptografi klasik jaman romawi masih menggunakan pena dan kertas saja karena belum terdapat komputer pada saat itu. Kriptografi klasik tergolong kedalam kriptografi kunci simetrik.

Teknik kriptografi yang digunakan pada kriptografi klasik yaitu :

1. Teknik subtitusi

   Teknik subtitusi merupakan penggantian setiap karakter *plaintext* dengan karakter lain. Didalam teknik subtitusi kriptografi klasik terdapat beberapa istilah, diantaranya yaitu :

   a. *Monoalfabet*, merupakan setiap karakter *chipertext* mengganti satu macam karakter pada *plaintext.*

b. *Polyalfabet*, setiap karakter *chipertext* dapat mengganti lebih dari satu macam karakter *plaintext.*

c. *Monograf / unilateral*, satu enkripsi dilakukan terhadap satu karakter *plaintext.*

d. *Polygraf / multilateral*, satu enkripsi dilakukan terhadap lebih dari satu karakter *plaintext* sekaligus.

Chipper subtitusi yang paling lama dikenal dengan nama subtitusi Yulius Caesar. Berikut dibawah ini merupakan contoh dari subtitusi, Dalam hal ini kuncinya adalah jumlah pergeseran huruf (yaitu $k = 3$) :

Tabel substitusi:
pi : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ci : **D E F G H I J K L M N O P Q R S T U V W X Y Z A B C**
Plaintext (pesan yang akan ditulis) : **KRIPTOGRAFI**

Maka *chipertext* (pesan yang telah dienkripsi) menjadi : **NULSWRJUDIL**

2. Teknik Transposisi

Pada *cipher* transposisi, plainteks tetap sama, tetapi urutannya diubah. Dengan kata lain, algoritma ini melakukan *transpose* terhadap rangkaian karakter di dalam teks. Nama lain untuk metode ini adalah permutasi, karena *transpose* setiap karakter di dalam *teks* sama dengan mempermutasikan karakter-karakter tersebut. Untuk meng-enkripsi pesan, *plaintext* ditulis secara horizontal dengan lebar kolom tetap, misal selebar 6 karakter (kunci $k = 6$), sedangkan *chipertext* dibaca secara vertikal.

Plaintext : ILMU KRIPTOGRAFI DUA

Maka *Chipertext* : ILMUKR

IPTOGR

AFIDUA

Maka *chipertext* dibaca vertikal menjadi :

IIALPFMTIUODKGURRA

2.3.2 Kriptografi *Modern*

Kriptografi modern merupakan kriptografi yang sering digunakan di era komputer modern sekarang ini. Kriptografi modern dalam pengoperasiannya beroperasi pada *bit*, dan kunci, *plaintext, chipertext* diproses dalam rangkaian bit. Perkembangan kriptografi modern didorong karena penggunaan komputer digital untuk keamanan suatu data atau pesan.

Dalam kriptografi modern, pesan dalam rangkaian bit dipecah menjadi beberapa blok. Seperti contoh dibawah ini dalam pemecahan bit kedalam blok :

Plaintext : 110010100010

Bila dibagi kedalam blok 4 – bit maka menjadi

1100 1010 0010

Maka setiap blok menyatakan bilangan 0 sampai dengan 15, menjadi :

12 10 2

2.4 Jenis – jenis Algoritma Kriptografi

2.4.1 Algoritma Simetris

Algoritma simetris *(symmetric algorithm)* adalah suatu algoritma dimana kunci enkripsi yang digunakan sama dengan kunci dekripsi sehingga algoritma ini disebut juga sebagai *single-key algorithm.*

Sebelum melakukan pengiriman pesan, pengirim dan penerima harus memilih suatu kunci tertentu yang sama untuk dipakai bersama, dan kunci ini haruslah rahasia bagi pihak yang tidak berkepentingan sehingga algoritma ini disebut juga algoritma kunci rahasia *(secret-key algorithm).* Yang termasuk dalam algoritma kunci simetri adalah OTP, DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5, Kasumi dan lain – lain.

Berikut kelebihan dari algoritma simetris :

a. Kecepatan operasi lebih tinggi bila dibandingkan dengan algoritma asimetris.
b. Karena kecepatannya yang cukup tinggi, maka dapat digunakan pada sistem *real-time.*

Berikut kelemahan dari algoritma simetris :

a. Untuk tiap pengiriman pesan dengan pengguna yang berbeda dibutuhkan kunci yang berbeda juga, sehingga akan terjadi kesulitan dalam manajemen kunci tersebut.
b. Permasalahan dalam pengiriman kunci itu sendiri yang disebut *"key distribution problem"*

2.4.2 Algoritma Asimetris

Algoritma asimetris *(asymmetric algorithm)* adalah suatu algoritma dimana kunci enkripsi yang digunakan tidak sama dengan kunci dekripsi. Pada algoritma ini menggunakan dua kunci yakni kunci publik *(public key)* dan kunci privat *(private key).* Kunci publik disebarkan secara umum sedangkan kunci privat disimpan secara rahasia oleh si pengguna. Walau kunci publik telah diketahui namun akan sangat sukar mengetahui kunci privat yang digunakan.yang termasuk dalam algoritma kunci asimetri adalah ECC, LUC, RSA, El Gamal dan DH.

Berikut kelebihan algoritma asimetris :

a. Masalah keamanan pada distribusi kunci dapat lebih baik
b. Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit.

Kelemahan dari algoritma asimetris :

a. Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris
b. Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris.

2.5 Blok *Chiper* Dan *Stream Chiper*

2.5.1 Blok *Chiper*

Chipper blok adalah algoritma kriptografi yang beroperasi dalam bentuk blok bit. Proses enkripsi dilakukan dalam blok bit *plaintext* menggunakan kunci yang berukuran sama dengan ukuran blok *plaintext.* Pada algoritma ini akan menghasilkan *chipertext* yang sama dengan blok *plaintext.*

2.5.1.1 Mode Operasi Blok *Chiper*

Pada algoritma kriptografi yang beroperasi pada blok dikenal dengan beberapa mode operasi, yaitu :

1. *Electronic Code Book* (ECB)

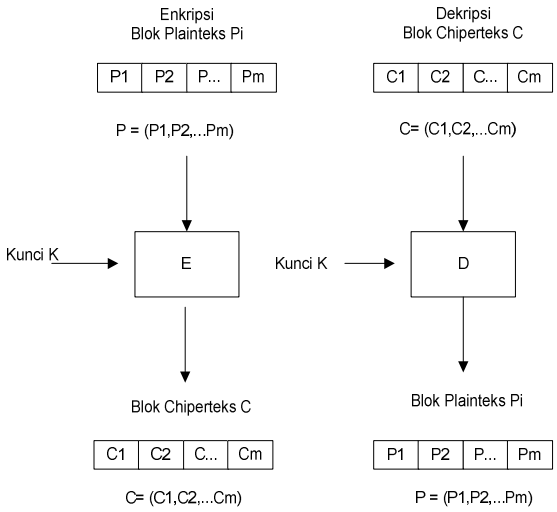
   Pada mode *Electronic Code Book* (ECB) sebuah blok input pada *plaintext* dienkripsi secara *individual* dan *independen* menjadi blok *chipertext*. Secara matematis proses enkripsi dan dekripsi pada mode ECB yaitu :

   Proses enkripsi : $Ci = E_k(Pi)$ (2.7)

   dan proses dekripsi : $Pi = D_k(Ci)$ (2.8)

   Dalam hal ini Pi dan Ci merupakan blok *plaintext* dan *chipertext* ke – i.

   Berikut dibawah ini penjelasan dari skema enkripsi dan dekripsi pada mode ECB yang dijelaskan pada gambar 2.5 :



Gambar 2.5 skema enkripsi dan dekripsi pada mode ECB

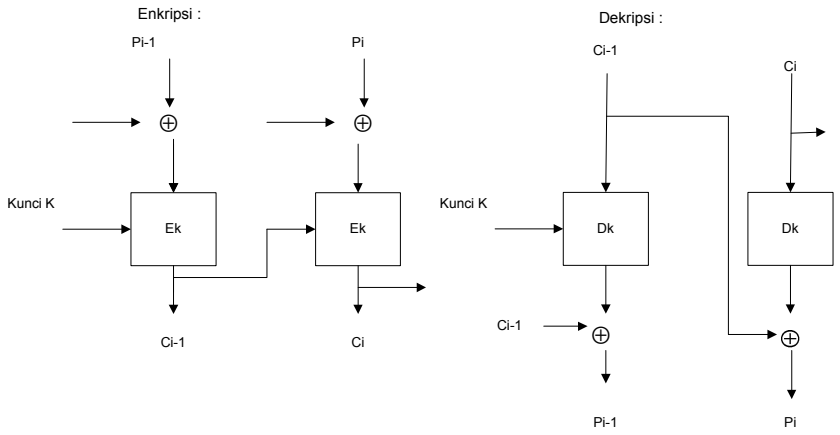
2. *Chipper Blok Chaining* (CBC)

Pada mode operasi CBC, hasil dari proses enkripsi dari blok sebelumnya mempengaruhi hasil enkripsi selanjutnya, atau enkripsi sebelumnya menjadi *feedback* pada enkripsi blok saat itu., jadi tiap blok *ciphertext* bergantung bukan hanya pada blok *plaintext*-nya tapi bergantung pula pada blok-blok *plaintext* sebelumnya. Sehingga untuk *plaintext* yang sama, belum tentu menghasilkan *ciphertext* yang sama pula. Secara matematis proses enkripsi dan dekripsi dapat dinyatakan sebagai berikut :

Proses Enkripsi $Ci = E_K(Pi \oplus Ci\text{-}1)$ (2.9)

Proses dekripsi $Pi = D_K(Ci) \oplus Ci\text{-}1$ (2.10)

Berikut dibawah ini merupakan skema dari mode operasi CBC, yang digambargakan pada gambar 2.6 :



Gambar 2.6 Skema enkripsi dan dekripsi mode operasi CBC

3. *Chiper Feed Back* (CFB)

   Jika pada mode *CBC*, plaintext sebesar n bit diproses dalam sekali waktu (menggunakan sebuah n bit cipher blok), beberapa aplikasi mengharuskan r bit plaintext untuk dienkripsi terlebih dahulu dan ditransmisikan bebas delay, untuk r < n (biasanya r = 1 atau r = 8); dalam kasus ini *CBF* digunakan. Dalam mode ini juga melibatkan penggunaan *initializing vector (IV*). Secara matematis proses enkripsi dan dekripsinya dinyatakan sebagai berikut :

Proses enkripsi :

$$Ci = Pi \oplus MSBm(EK(Xi)) \quad (2.11)$$

$$Xi+1 = LSBm\text{-}n(Xi) \,||\, Ci \quad (2.12)$$

Proses dekripsi :

$$Pi = Ci \oplus MSBm(DK(Xi)) \quad (2.13)$$

$$Xi+1 = LSBm\text{-}n(Xi) \,||\, Ci \quad (2.14)$$

Keterangan:

Xi = isi antrian dengan X1 adalah IV

E = fungsi enkripsi

K = kunci

M = panjang blok enkripsi

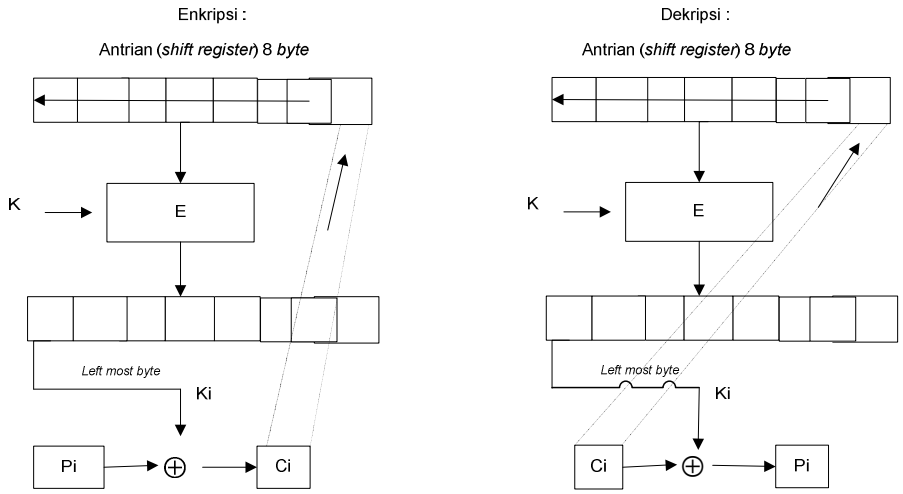
N = panjang unit enkripsi

|| = operator penyambungan (concatenation)

MSB = Most Significant Byte

LSB = Least Significant Byte

Gambar 2.7 dibawah ini merupakan penjelasan dari mode operasi CFB :



Gambar 2.7 skema enkripsi dan dekripsi mode operasi CFB

4. *Output Feed Back* (OFB)

Pada mode OFB bekerja mirip dengan mode CFB, kecuali n – bit dari hasil enkripsinya. Secara matematis proses enkripsi dan dekripsi pada mode OFB n - bit dapat dinyatakan sebagai berikut :

Proses enkripsi :

$$Ci = Pi \oplus MSBm(EK(Xi)) \quad (2.15)$$

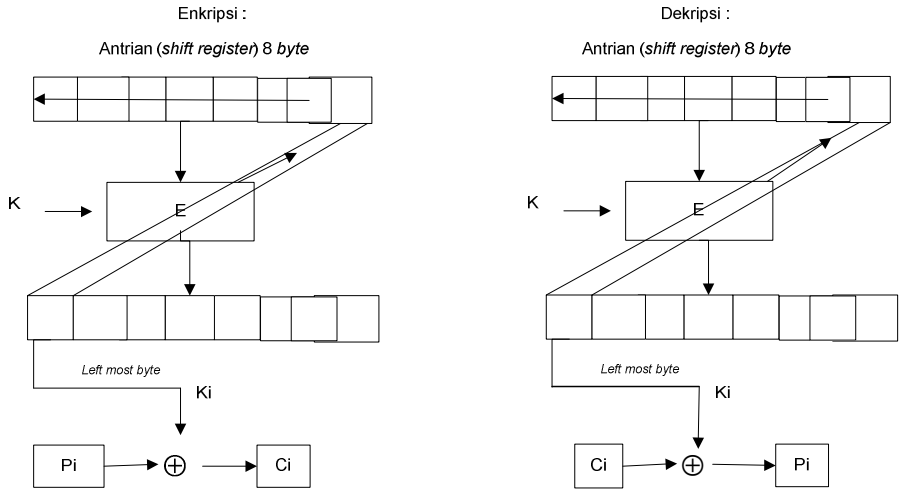
$$Xi+1 = LSBm-n(Xi) \,||\, MSBm(EK(Xi)) \quad (2.16)$$

Proses dekripsi :

$$Pi = Ci \oplus MSBm(DK(Xi)) \quad (2.17)$$

$$Xi+1 = LSBm-n(Xi) \,||\, MSBm(EK(Xi)) \quad (2.18)$$

Gambar 2.8 dibawah ini merupakan penjelasan dari mode operasi OFB :



Gambar 2.8 skema enkripsi dan dekripsi mode operasi OFB

2.5.2 *Stream Chiper*

Stream Chiper adalah *chipper* yang berasal dari hasil XOR antara setiap *bit plaintext* dengan setiap *bit* kuncinya. jadi format data berupa aliran dari bit untuk kemudian mengalami proses enkripsi dan dekripsi.

2.5.3 Desain *Chiper*

Terdapat dua buah prinsip dasar dalam mengghasilkan *Chiper* yang aman, yaitu :

1. *Confusion* yaitu mengaburkan hubungan antara *plaintext* dan *chipertext.* cara yang dapat dilakukan dalam prinsip *confusion* adalah dengan substitusi.
2. *Diffusion* yaitu menghilangkan redudansi *plaintext* dengan menyebarkan masukan keseluruh *chipertext.* Diperlukan waktu yang lama dalam memecahkan sandi rahasia ini, cara yang dilakukan dalam prinsip *diffusion* yaitu transposisi dan permutasi.

2.6 Blowfish

2.6.1 Deskripsi Blowfish

Blowfish atau "*OpenPGP.Cipher.4*" merupakan enkripsi yang termasuk dalam golongan *Symetric Cryptosystem*, metoda enkripsinya mirip dengan DES (DES-like Cipher) ditemukan 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 *microprocesor* besar (32-bit keatas dengan *cache* data yang besar).

Blowfish dikembangkan untuk memenuhi kriteria desain yang cepat dalam implementasinya dimana pada keadaan optimal dapat mencapai 26 *clock cycle per byte*, dimana dapat berjalan pada memori kurang dari 5 KB, sederhana dalam algoritmanya sehingga mudah diketahui kesalahannya, dan keamanan yang variabel

dimana panjang kunci bervariasi (minimum 32 *bit*, maksimum 448 *bit*, *Multiple* 8 *bit*, *default* 128 *bit*). *Blowfish* dioptimasikan untuk berbagai aplikasi dimana kunci tidak sering berubah, seperti pada jaringan komunikasi atau enkripsi secara otomatis. Dalam pengimplementasiannya dalam komputer bermicroprosesor 32-*bit* dengan *cache* data yang besar (*Pentium* dan *Power* PC) *Blowfish* terbukti jauh lebih cepat dari DES. Tetapi *Blowfish* tidak cocok dengan aplikasi dengan perubahan kunci yang sering atau sebagai fungsi *hash* satu arah seperti pada aplikasi *packet switching*. *Blowfish* pun tidak dapat digunakan pada aplikasi kartu pintar (*smart card*) karena memerlukan memori yang besar.

Blowfish termasuk dalam enkripsi *block Cipher* 64-bit dangan panjang kunci yang bervariasi antara 32-bit sampai 448-bit.Algoritma *Blowfish* terdiri atas dua bagian :

1. *Key-Expansion*

   Berfungsi merubah kunci (Minimum 32-bit, Maksimum 448-bit) menjadi beberapa array subkunci (*subkey*) dengan total 4168 *byte*.

2. Enkripsi Data

   Terdiri dari iterasi fungsi sederhana (*Feistel Network*) sebanyak 16 kali putaran. 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 (*table lookup*) array berindeks untuk setiap putaran.

2.6.2 Algoritma Blowfish

Blowfish merupakan algoritma kriptografi kunci simetrik *chiper* blok dengan panjang blok tetap sepanjang 64 *bit. Blowfish* menerapkan teknik kunci yang berukuran sembarang. Ukuran kunci yang dapat diterima adalah sepanjang 32 *bit* sampai 448 *bit*, dengan ukuran *default* sebesar 128 *bit*.

Blowfish adalah algoritma yang menerapkan jaringan Feistel (*Feistel Network*) yang terdiri dari 16 putaran. Input adalah elemen 64-*bit,*X.Untuk alur algoritma enkripsi dengan metoda *Blowfish* dijelaskan sebagai berikut :

1. Bentuk inisial P-array sebanyak 18 buah (P1,P2,.....P18) masing-msing bernilai 32-bit.

   Array P terdiri dari delapan belas kunci 32-bit subkunci :

   $P_1,P_2,.....,P_{18}$ (2.19)

2. Bentuk S-box sebanyak 4 buah masing-masing bernilai 32-bit yang memiliki masukan 256.

   Empat 32-bit S-box masing-masing mempunyai 256 entri :

   $S_{1,0},S_{1,1},.....,S_{1,255}$ (2.20)

   $S_{2,0},S_{2,1},.....,S_{2,255}$ (2.21)

$S_{3,0}, S_{3,1}, \ldots\ldots\ldots\ldots\ldots, S_{3,255}$ (2.22)

$S_{4,0}, S_{4,1}, \ldots\ldots\ldots\ldots\ldots, S_{4,255}$ (2.23)

3. *Plaintext* yang akan dienkripsi diasumsikan sebagai masukan, *Plaintext* tersebut diambil sebanyak 64-*bit*, dan apabila kurang dari 64-*bit* maka kita tambahkan bitnya, supaya dalam operasi nanti sesuai dengan datanya.

4. Hasil pengambilan tadi dibagi 2, 32-bit pertama disebut XL, 32-bit yang kedua disebut XR.

5. Selanjutnya lakukan operasi XL = XL xor Pi dan XR = F(XL) xor XR

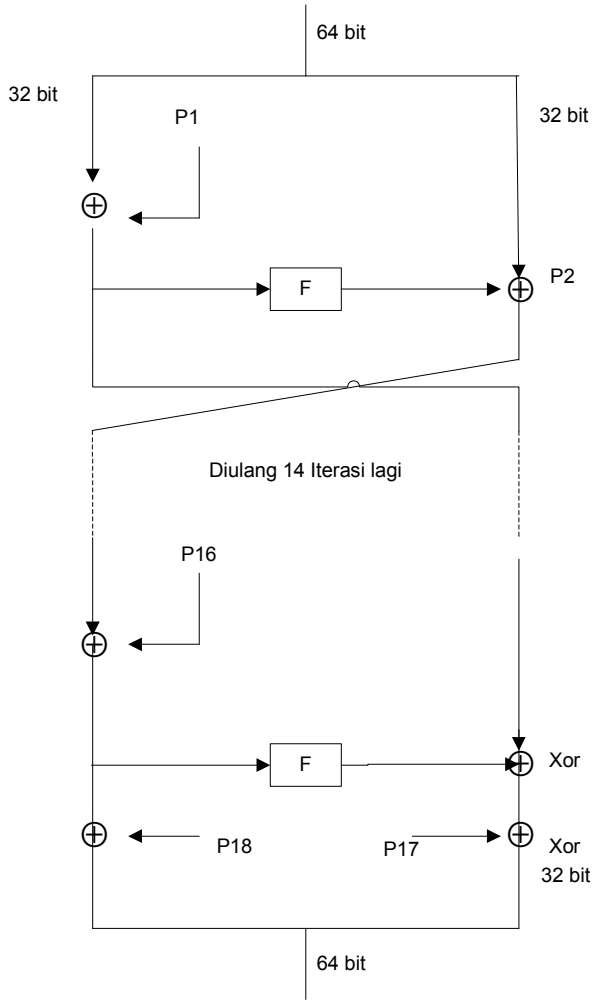
6. Hasil dari operrasi diatas ditukar XL menjadi XR dan XR menjadi XL.

7. Lakukan sebanyak 16 kali, perulangan yang ke-16 lakukan lagi proses penukaran XL dan XR.

8. Pada proses ke-17 lakukan operasi untuk XR = XR xor P17 dan XL = XL xor P18.

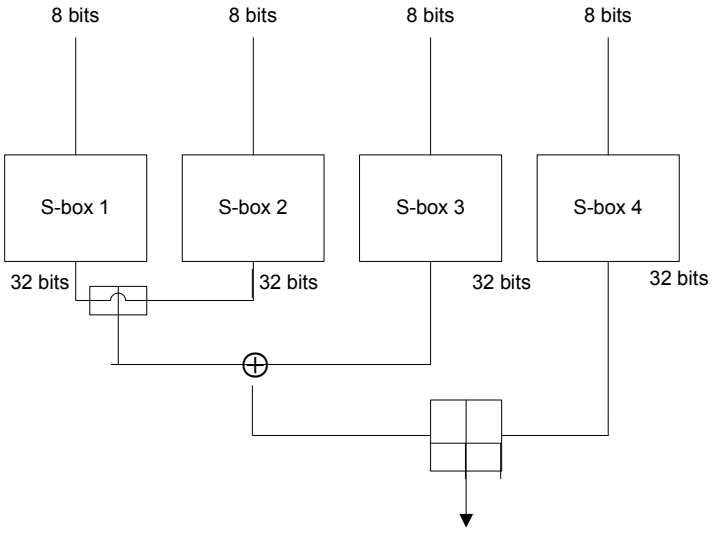
9. Proses terakhir satukan kembali XL dan XR sehingga menjadi 64-bit kembali.

Di bawah ini merupakan gambar dari struktur algoritma *blowfish*, yang digambarkan pada gambar 2.9 :



Gambar 2.9 Struktur Algoritma *Blowfish*

Di bawah ini merupakan gambar dari skema fungsi F dari algoritma *blowfish,* yang digambarkan pada gambar 2.10 :



Gambar 2.10 skema fungsi F pada Algoritma *Blowfish*

2.6.3 Keamanan Blowfish

Sampai saat ini algoritma Blowfish belum ditemukan kelemahan yang berarti hanya adanya *weak key* dimana dua entri dari S-box mempunyai nilai yang sama. Belum ada cara untuk mengecek *weak key* sebelum melakukan *key expansion*, tetapi hal ini tidak berpengaruh terhadap hasil enkripsi. Hasil enkripsi dengan algoritma Blowfish sangat tidak mungkin dan tidak praktis untuk di terjemahkan tanpa bantuan kunci. Sampai kini belum ada *Cryptanalysis* yang dapat membongkar pesan tanpa kunci yang enkripsi oleh Blowfish. Agar aman dari pembongkaran pesan maka dalam algoritmanya harus menggunakan 16 putaran agar pesan tersebut tidak dapat dibongkar. Algoritma Blowfish pun da-

pat digabungkan dengan algoritma-algoritma enkripsi yang lain dalam pengkripsian sebuah pesan untuk lebih menjamin isi dari pesan tersebut.

John Kelsey mengembangkan sebuah metode serangan yang dapat memecahkan Blowfish dengan tiga putaran, tetapi tidak dapat mengembangkan lebih dari itu. Penyerangan ini mengeksploitasi fungsi F. Vikramjit Singh Chabra juga telah mencari cara yang efisien untuk mengimplementasikan mesin pencarian kunci dengan cara lempang (*brute force*).

Serge Vaudenay melakukan pemeriksaan terhadap Blowfish dengan kotak-S diketahui dan putaran sebanyak r. Proses pemeriksaan yang dilakukan dengan serangan diferensial dapat menghasilkan *P-array* dengan 28r+1 *chosen plainteks* [SCH95]. Untuk kunci lemah tertentu yang menghasilkan kotak-S yang buruk, serangan yang sama hanya membutuhkan 24r+1 *chosen* plainteks untuk menghasilkan *P-array*. Kemungkinan untuk mendapatkan kunci lemah ini sendiri adalah 1 berbanding 214. Tanpa diketahuinya kotak-S yang digunakan, serangan ini dapat mendeteksi lemah tidaknya kunci yang digunakan, tetapi tidak dapat menentukan kunci.

2.7 Twofish

2.7.1 Deskripsi Twofish

Pada tahun 1972 dan 1974, US the National Bureau of Standards (sekarang bernama the National Institute of Standards and Technology, atau NIST) mengeluarkan publikasi pertama untuk sebuah standar enkripsi, yang menghasilkan al-

goritma data Encryption Standard (DES) [NBS77], yang tidak dapat disangkal sebagai algoritma kriptografi yang sangat terkenal dan sangat berhasil.

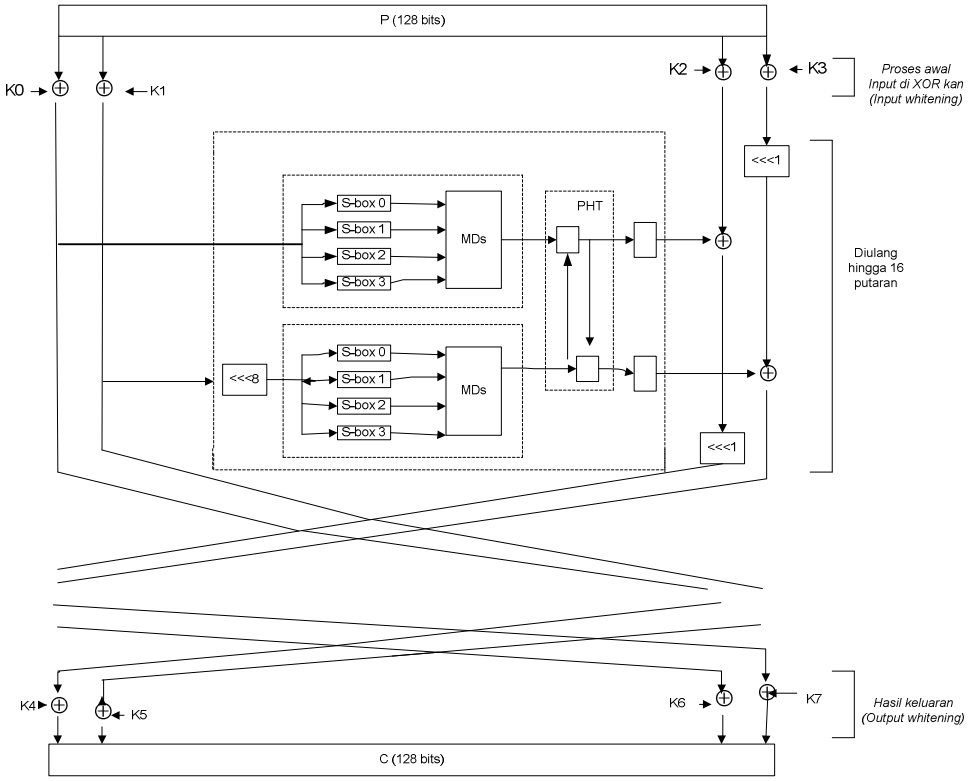
Twofish merupakan 128-*bit* block sandi/*cipher* yang bisa menerima panjang varibel kunci/key sebesar 256 *bit*. *Cipher* tersebut berasal 16-*round* jaringan Feitsel dengan fungsi bijektif F yang dilanjutkan dengan empat *key-dependent 8-by-b-bit S-boxes*, satu *fixed 4-by-4 maximum distance separable matrix over* GF(2^8), satu *pseudo-Hadamard transform*, satu rotasi *bitwise* dan satu desain *key schedule*. Suatu implementasi *Twofish* yang dioptimalkan mengenksripsi pada Pentium Pro dengan 17,8 siklus *clock per byte*, dan pada *smartcard* akan mengenksripsi pada 1660 siklus *clock per byte*. *Twofish* dapat diimplemetsikan pada pada perangkat keras dengan 14000 gerbang. *Design round function* dan penjadwalan kunci mengakibatkan adanya *trade off* antara kecepatan, ukuran *software*, waktu *setup key*, jumlah gerbang dan *memory*.

2.7.2 Algoritma Twofish

Twofish menggunakan struktur Feistel 16-*round* dengan *whitening* tambahan dalam input dan outputnya. Satu-satunya elemen yang bukan Feistel adalah rotasi 1 *bit*. Rotasi tersebut dapat dipindahkan ke fungsi *F* untuk menciptakan *output* berjalan.

Plaintext dipecah menjadi empat buah *word* 32-*bit*. Pada *whitening input*, keempat *word* itu di XOR-kan dengan empat *key word*. Dan di ikuti dengan ke enam belas *round*. Dalam tiap *round*, dua *word* di kiri digunakan sebagai *input* fungsi *g* (Salah satunya dirotasikan dengan 8 *bit* terlebih dahulu). Gambar 2.11

dibawah ini merupakan gambar dari struktur algoritma *Twofish* yang dikutip dari sebuah jurnal yang berjudul "*Implementation of the Twofish chiper using FPGA devices"* yang ditulis oleh Pawel Chodoweic, Kris Gaj :



Gambar 2.11 Struktur Algoritma *Twofish*

Langkah-langkah algoritma twofish adalah

sebagai berikut:

1. Masukan satu blok *plainteks* adalah 128 bit. Satu blok tersebut dibagi menjadi 4 buah sub-blok yang masing-masing sepanjang 32 bit (A, B, C, dan D).
2. Masing-masing sub-blok tersebut melalui proses *whitening* dengan meng-Xor-kan dengan kunci K0, K1, K2, dan K3.

$$R_{0,i} = P_i \oplus K_i \quad \rightarrow i = 0...3 \tag{2.24}$$

Proses pada *input whitening* :

$$R_0,i = P_i \text{ XOR } Ki \rightarrow i = 0\ldots3 \tag{2.25}$$

$$(Fr,0, Fr,1) = F (Rr,0,Rr,1,r) \tag{2.26}$$

$$Rr+1,0 = ROR (Rr,2 \text{ XOR } Fr,0,1) \tag{2.27}$$

$$Rr+1,1 = ROL (Rr,3,1) \text{ XOR } Fr,1 \tag{2.28}$$

$$Rr+1,2 = Rr,0 \tag{2.29}$$

$$Rr+1,3 = Rr,1 \tag{2.30}$$

Proses pada *output whitening* :

$$Ci=R16\ (i+2) \bmod 4 \text{ XOR } Ki+4 \rightarrow i = 0...3 \tag{2.31}$$

Langkah-langkah 1 putaran adalah sebagai berikut:

1. 2 buah 32 bit yang kiri (A dan B) merupakan input dari fungsi g (yang merupakan bagian dari fungsi f), yang salah satunya (B) di geser ke kiri sejauh 8 bit dahulu.
2. Fungsi g memiliki 4 buah kotak substitusi yang dibangkitkan oleh kunci.
3. Keluaran fungsi kotak substitusi dilakukan percampuran linear menggunakan kotak Most Distance Separable
4. Keluaran fungsi g dimasukkan ke fungsi transformasi pseudo-Hadamard, kemudian ditambahkan dengan 2 buah 32 bit dari kunci.
5. Dua buah 32 bit hasil kemudian di xorkan dengan C dan D. Hasil xor dengan C digeser ke kanan sejauh 1 bit. Dan untuk D sebelum dixorkan digeser ke kiri sejauh 1 bit.

6. 2 buah 32 bit kiri dan kanan dipertukarkan (A dan B dipertukarkan dengan C dan D).

Fungsi F adalah permutasi yang bergantung pada kunci dengan nilai 64 bit. Fungsi ini menerima 3 argumen, dua buah 32 bit R0 dan R1, dan nomor putaran untuk menentukan subkunci mana yang dipakai. R0 akan diserahkan ke fungsi g yang akan mengembalikan T0. R1 akan digeser sejauh 8 bit yang kemudian di berikan juga ke fungsi g yang akan mengembalikan T1. Hasil T0 dan T1 kemudian dikombinasikan ulang menggunakan transformasi pseudo-Hadamard, yang kemudian ditambahkan dengan dua buah 32 bit dari kunci.

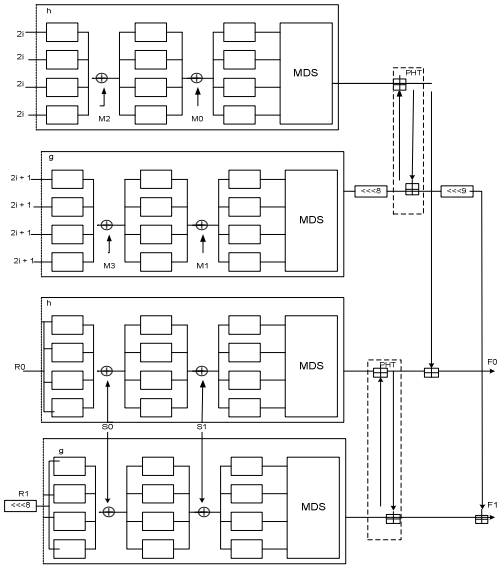
T0 = g(R0); (2.32)

T1 = g(shiftLeft(R1,8)); (2.33)

F0 = (T0+T1+K2r+8) mod 232; (2.34)

F1 = (T0+2T1+K2r+9) mod 232; (2.35)

F0 dan F1 adalah hasil dari F, yang masing – masing sepanjang 32 bit. Hasil keluaran ini nantinya akan dipertukarkan dan dimasukkan kembali ke putaran selanjutnya. Gambar 2.12 Dibawah ini merupakan skema dari fungsi F pada algoritma *Twofish* :



Gambar 2.12 Skema Fungsi F pada Algoritma *Twofish*

Fungsi g merupakan jantung dari keseluruhan algoritma twofish. 32 bit masukan X dari fungsi F dipecah menajdi 4 buah yang masing - masing sepanjang 8 bit. Setiap 8 bit kemudian diproses dengan kotak S yang bersesuaian. Setiap kotak S bersifat bijektif, yaitu menerima 8 bit dan mengeluarkan 8 bit pula. 4 buah 8 bit hasil keluaran kemudian dikalikan dengan matriks *Most Distance Separable* (MDS) 4x4. Hasil pengalian kemudian diartikan sebagai 32 bit, yang merupakan keluaran dari fungsi g, yang kemudian akan dikembalikan kembali ke fungsi F.

2.7.3 Proses *Whitening*

Whitening adalah suatu teknik yang terdapat dalam algoritma kriptografi *Twofish* dengan cara meng-XOR-kan 128-*bit* sub-kunci sebelum sebelum *round* pertama (*input whitening*) dan 128-*bit* setelah *round* terakhir (*output whitening*). Dalam struktur algoritma *Twofish* terdapat *input whitening* yang merupakan masuk-

kan dari kunci yang dipakai dan di XOR-kan, sedangkan untuk *output whitening* merupakan hasil keluaran *whitening* terhadap kunci setelah melakukan *whitening round* terakhir.

.

2.7.4 Keamanan Twofish

Studi mengenai keamanan dan kriptanalisis algoritma *Twofish* sudah banyak dilakukan. Walaupun begitu semua studi ini dilakukan terhadap algoritma *Twofish* yang sudah disederhanakan atau diperlemah.

Untuk *Twofish* dengan lima putaran, tanpa proses *whitening* pada awal dan akhir proses dibutuhkan 222,5 pasangan *chosen* plainteks dan 251 usaha. Metode kriptanalisis untuk *Twofish* ini masih terus dikembangkan. Tetapi tidak ada metode kriptanalisis untuk Twofish dengan jumlah putaran di atas sembilan buah. Selain dengan serangan *chosen* plainteks, telah dilakukan juga kriptanalisis dengan serangan *related-key*. Metode kriptanalisis yang dilakukan adalah serangan *chosen-key* parsial pada *Twofish* dengan 10 putaran tanpa proses *whitening* pada awal dan akhir proses. Untuk melakukan proses kriptanalisis ini harus disiapkan pasangan kunci-kunci yang berhubungan. Kemudian pilih 20 dari 32 *byte* dari setiap kunci. Dua puluh *byte* yang dipilih berada di bawah kendali kriptanalisis. Sedangkan dua belas *byte* sisanya tidak dikatahui, tetapi kriptanalisis dapat mengetahui bahwa keduanya adalah sama untuk kedua kunci. Proses yang harus dilakukan sebanyak 264 *chosen* plainteks untuk setiap kunci yang dipilih dan dilakukan sekitar 234 usaha, untuk mendapatkan 12 *byte* kunci yang belum diketahui. Selain studi-studi di atas, dilakukan juga serangan terhadap algoritma yang telah di-

kurangi jumlah putarannya dengan penyederhanaan fitur-fitur tertentu, seperti: *Twofish* dengan kotak-S statis, *Twofish* tanpa rotasi 1 bit, dan yang lainnya. Tetapi hasil studi ini menunjukkan bahwa kriptanalisis tidak dapat memecahkan *Twofish* walaupun dengan penyederhanaan-penyederhanaan yang telah dilakukan [SCH01].