BAB II
LANDASAN TEORI

2.1 Pengenalan Kriptografi

2.1.1 Sejarah Kriptografi

Kriptografi adalah ilmu yang mempelajari bagaimana suatu pesan atau dokumen kita aman, tidak bisa dibaca oleh pihak yang tidak berhak. Dalam perkembangannya, kriptografi juga digunakan untuk identifikasi pengirim pesan dengan tanda tangan digital dan keaslian pesan dengan sidik jari digital (*fingerprint*). Kriptografi mempunyai sejarah yang sangat panjang. Sejarah kriptografi dimulai pertama sekali dengan menggunakan metode pertukaran posisi untuk mengenkripsi suatu pesan. Dalam sejarah perkembangannya, Julius Caesar dalam mengirimkan pesan yang dibawa oleh hulubalangnya, sengaja mengacak pesan tersebut sebelum diberikan kepada kurir. Hal ini dilakukan untuk menjaga kerahasiaan pesan baik bagi kurir maupun bagi musuh jika kurir tertangkap di tengah jalan oleh musuh. Ada orang yang mengatakan bahwa apa yang dilakukan oleh Julius Caesar dianggap sebagai awal dari kriptografi.

Dalam sebuah buku yang berjudul *The Codebreaker* yang dikarang oleh David Kahn pada tahun 1963, disebutkan bahwa kriptografi digunakan pertama sekali oleh bangsa Mesir 4000 tahun yang lalu sampai saat sekarang ini. Sejak munculnya buku tersebut maka kriptografi pun mulai diperbincangkan secara luas. Peminat dari buku tersebut ialah peminat yang berhubungan dengan kemiliteran, layanan

diplomatik dan pemerintahan. Kriptografi digunakan sebagai suatu alat untuk melindungi rahasia dan strategi – strategi negara.

Sampai pada akhir Perang Dunia I, kriptografi merupakan disiplin ilmu matematika yang spesial. Penelitian dalam bidang ini tidak pernah sampai kepada umum sehingga tidaklah mengherankan kalau banyak orang tidak mengetahui keberadaan ataupun manfaat darinya. Kemudian pada Perang Dunia II, pihak militer pun mulai menyadari akan manfaat dari penggunaan kriptografi maupun kriptanalisis. Kriptografi memungkinkan untuk berkomunikasi dalam saluran yang aman (misalnya komunikasi melalui radio gelombang panjang) dengan cara membuatnya menjadi tidak dapat dimengerti oleh musuh. Kriptografi mencapai kemajuan yang pesat pada akhir Perang Dunia II. Akan tetapi kriptografi masih merupakan sesuatu yang sangat rahasia karena kriptografi telah menjadi bagian yang penting dalam komunikasi militer.

Perkembangan komputer dan sistem komunikasi pada tahun 1960-an mengakibatkan munculnya kebutuhan pihak swasta akan alat untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan informasi. Kriptografi digital dimulai pada tahun 1970 atas usaha *Feistel* dari IBM dan memuncak pada tahun 1977 dengan diadopsinya sistem kriptografi DES (*Data Encryption Standard*) oleh *U.S. Federal Information Processing Standard* untuk mengenkripsi informasi rahasia. DES merupakan mekanisme kriptografi yang paling

terkenal dalam sejarah dan tetap menjadi standar pengamanan data elektronik komersial pada kebanyakan institusi keuangan di seluruh dunia.

Perkembangan yang paling pesat dan berpengaruh dalam sejarah kriptografi ialah pada tahun 1976 dimana Whitfield Diffie dan Martin Hellman mempublikasikan sebuah tesis berjudul *New Direction in Cryptography*. Dalam tesis ini diperkenalkan konsep kunci publik kriptografi yang paling revolusioner dan juga menyediakan metode baru dalam pertukaran kunci, yaitu keamanan yang didasarkan atas logaritma diskrit. Walaupun penulis tesis tersebut tidak mempunyai praktek yang nyata akan bentuk skema enkripsi kunci publik pada saat itu akan tetapi ide tersebut memicu minat dan aktivitas yang besar dalam komunitas kriptografi. Pada tahun 1978, Rivest, Shamir, dan Adleman menemukan enkripsi kunci publik yang pertama dan sekarang ini dikenal dengan nama RSA (*Rivest, Shamir, and Adleman*). Skema RSA didasarkan pada permasalahan matematika sulit yang terdiri dari pemfaktoran terhadap bilangan yang besar nilainya. Karena adanya permasalahan matematika tersebut maka muncul usaha – usaha untuk mencari cara yang paling efisien dalam pemfaktoran bilangan. Skema kunci publik lainnya yang kuat dan praktis ditemukan oleh *ElGamal*. Skema ini juga berdasarkan masalah logaritma diskrit.

Salah satu sumbangan yang paling penting dari kriptografi kunci publik ialah tanda tangan digital. Pada tahun 1991, standar internasional yang pertama untuk tanda tangan digital yang dipergunakan adalah berdasarkan pada skema kunci publik

RSA. Pada tahun 1994 pemerintah Amerika Serikat mengadopsi standar tanda tangan digital yang didasarkan pada mekanisme skema kunci publik *ElGamal*.

Pencarian terhadap skema kunci publik yang baru dengan pengembangan dari mekanisme kriptografi yang sudah ada dan pembuktian keamanan berlangsung dengan cepat. Berbagai standar dan infrastruktur yang berhubungan dengan kriptografi sedang dibangun. Produk – produk keamanan sedang dikembangkan untuk memenuhi kebutuhan akan keamanan informasi pada masyarakat.

2.1.2 Definisi Kriptografi

Kriptografi (*cryptography*) berasal dari kata '**kryptos**' yang artinya tersembunyi dan '**grafia**' yang artinya sesuatu yang tertulis (bahasa Yunani) sehingga kriptografi dapat juga disebut sebagai sesuatu yang tertulis secara rahasia (tersembunyi).

Kriptografi adalah ilmu yang mempelajari teknik – teknik matematika yang berhubungan dengan aspek-aspek pada keamanan informasi misalnya kerahasiaan, integritas data, otentikasi pengirim / penerima data, dan otentikasi data. Dengan pengembangan bidang kriptografi, pembagian antara apa yang termasuk kriptografi dan apa yang tidak telah menjadi kabur. Dewasa ini, kriptografi dapat dianggap sebagai perpaduan antara studi teknik dan aplikasi yang tergantung kepada keberadaan masalah – masalah sulit.

Bagi kebanyakan orang, kriptografi lebih diutamakan dalam menjaga komunikasi agar tetap rahasia. Seperti yang telah diketahui dan disetujui bahwa perlindungan (proteksi) terhadap komunikasi yang sensitif telah menjadi penekanan kriptografi selama ini. Akan tetapi hal tersebut hanyalah sebagian dari penerapan kriptografi dewasa ini.

Terdapat dua proses penting di dalam kriptografi yang berperan dalam merahasiakan suatu informasi yakni enkripsi (*encryption*) dan dekripsi (*decryption*). Enkripsi adalah transformasi data (*plaintext*) ke dalam bentuk yang hampir tidak dapat dibaca (*ciphertext*) tanpa pengetahuan yang cukup. Tujuan dari enkripsi adalah untuk menjamin kerahasiaan dengan menjaga informasi tersembunyi dari siapapun yang bukan pemilik atau yang berkepentingan dengan informasi tersebut, bahkan bagi orang yang memiliki akses terhadap data yang telah dienkripsi. Sedangkan dekripsi adalah kebalikan dari enkripsi, yakni transformasi dari data yang telah dienkripsi (*ciphertext*) kembali ke bentuk semula (*plaintext*). Proses enkripsi dan dekripsi pada umumnya membutuhkan penggunaan sejumlah informasi yang rahasia, yang sering disebut kunci (*key*).

2.1.3 Tujuan Kriptografi

Menurut Stalling, ada beberapa tuntutan yang terkait dengan isu keamanan data yaitu :

1. *Confidentiality*

   Menjamin bahwa data-data tersebut hanya bisa diakses oleh pihak-pihak tertentu saja.

2. *Authentication*

   Baik pada saat mengirim atau menerima informasi, kedua belah pihak perlu mengetahui bahwa pengirim dari pesan tersebut adalah orang yang sebenarnya seperti yang diklaim.

3. *Integrity*

   Tuntutan ini berhubungan dengan jaminan setiap pesan yang dikirim pasti sampai pada penerimanya tanpa ada bagian dari pesan tersebut yang diganti, diduplikasi, dirusak, diubah urutannya, dan ditambahkan.

4. *Nonrepudiation*

   Nonrepudiation mencegah pengirim maupun penerima mengingkari bahwa mereka telah mengirimkan atau menerima suatu pesan/informasi. Jika sebuah pesan dikirim, penerima dapat membuktikan bahwa pesan tersebut memang dikirim oleh pengirim yang tertera. Sebaliknya, jika sebuah pesan diterima, pengirim dapat membuktikan bahwa pesannya telah diterima oleh pihak yang ditujunya.

5. *Access Control*

   Membatasi sumber-sumber data hanya kepada orang-orang tertentu.

6. *Availability*

   Jika diperlukan setiap saat semua informasi pada sistem komputer harus tersedia bagi semua pihak yang berhak atas informasi tersebut.

Dari keenam aspek keamanan data tersebut, empat diantaranya dapat diatasi dengan menggunakan kriptografi yaitu *confidentiality, integrity, authentication,* dan *nonrepudiation.*

2.2 Jenis Sistem Kriptografi

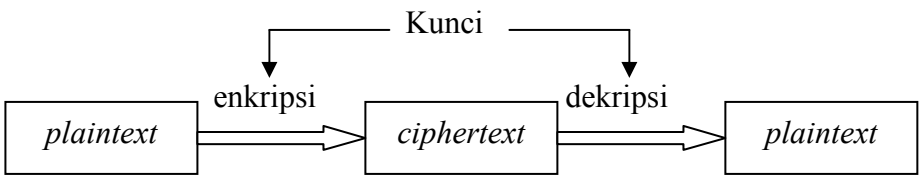
Berdasarkan pemakaian kunci maka sistem kriptografi (c*ryptosystems*) dapat digolongkan atas 2 jenis sistem yakni sistem kriptografi kunci publik (*public key cryptography*) dan sistem kriptografi kunci rahasia (*secret key cryptography*). Dalam sistem kriptografi kunci rahasia yang dikenal juga dengan *symmetric cryptosystems*, pihak pengirim dan penerima bersama-sama menyepakati sebuah kunci rahasia yang akan digunakan dalam proses enkripsi dan dekripsi tanpa diketahui oleh pihak lain. Sedangkan dalam sistem kriptografi kunci publik atau dikenal dengan *assymmetric cryptosystem*, pihak pengirim maupun pihak penerima mendapatkan sepasang kunci yakni kunci publik (*public key*) dan kunci rahasia (*private key*) dimana kunci publik dipublikasikan dan kunci rahasia tetap dirahasiakan. Enkripsi dilakukan dengan

menggunakan kunci publik sedangkan dekripsi dilakukan dengan menggunakan kunci rahasia.

2.2.1 Kriptografi Kunci Rahasia (*Secret Key Cryptography*)

Kriptografi kunci rahasia merupakan bentuk kriptografi yang lebih tradisional, dimana kunci tunggal dapat digunakan untuk enkripsi dan dekripsi suatu pesan. Kriptografi kunci rahasia tidak hanya digunakan untuk enkripsi, tetapi juga untuk otentikasi. Salah satu teknik untuk pekerjaan ini disebut *message authentication codes* (MAC).

Masalah utama dengan kriptografi kunci rahasia membuat pengirim dan penerima pesan setuju atas kunci rahasia yang digunakan tanpa orang lain mampu mendapatkan dan mengetahuinya. Atau dengan kata lain bagaimana memilih kunci rahasia yang benar-benar aman. Hal ini membutuhkan suatu metode dimana kedua pihak dapat berkomunikasi tanpa kekhawatiran akan tercecernya kunci tersebut. Akan tetapi, keuntungan dari kriptografi kunci rahasia adalah biasanya lebih cepat dibandingkan dengan kriptografi kunci publik. Metode yang paling umum untuk kriptografi kunci rahasia adalah *block ciphers*, *stream ciphers*, dan *message authentication codes* (MAC).



Gambar 2.1. Model sederhana dari Secret Key Cryptosystem

2.2.1.1 Block Cipher

Block cipher adalah bentuk algoritma enkripsi kunci simetri yang mentransformasikan satu blok data tertentu dari *plaintext* ke dalam satu blok data *ciphertext* dengan panjang blok yang sama. Transformasi ini berlangsung melalui penggunaan kunci rahasia yang disediakan oleh pemakai (*user*). Dekripsi dilakukan dengan menggunakan transformasi kebalikan terhadap blok *ciphertext* menjadi satu blok *plaintext* dengan kunci dan panjang blok yang sama. Panjang blok tertentu disebut ukuran blok (*block size*) dimana ukuran blok tersebut bervariasi misalnya 16 bit, 32 bit, 64 bit, 128 bit atau 256 bit tergantung dari teknik yang digunakan dan perkembangan kemampuan mikroprosesor selanjutnya.

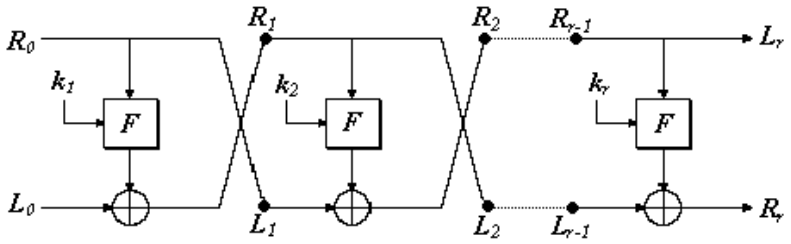
Karena blok *plaintext* yang berbeda dipetakan ke blok *ciphertext* yang berbeda (untuk memungkinkan dekripsi yang unik), suatu *block cipher* secara efektif menyediakan satu permutasi (korespondensi satu ke banyak) dari set pesan yang mungkin. Permutasi berpengaruh pada saat enkripsi tertentu yang sudah pasti rahasia, karena permutasi tersebut adalah fungsi dari kunci rahasia. Jika kita menggunakan satu *block cipher* untuk mengenkrip satu pesan dengan panjang sembarang, kita

menggunakan teknik yang dikenal sebagai modus operasi untuk *block cipher* tersebut. Agar dapat berguna, satu modus operasi setidak-tidaknya seefisien dan seaman *cipher* fundamental. Teknik enkripsi mungkin memiliki sifat-sifat tambahan terhadap sifat-sifat dasar yang dimiliki teknik biasa. Teknik standard DES telah dipublikasi dalam berbagai publikasi. Versi standard yang lebih umum menggabungkan 4 modus operasi dari DES untuk dapat diaplikasikan terhadap *block cipher* dengan ukuran blok sembarang. Standard tersebut adalah *Electronic Code Book*, *Cipher Block Chaining*, *Cipher Feedback*, dan *Output Feedback*. Berikut ini adalah bentuk umum dari *block cipher*,

1. *Iterated Block Cipher* (*Block Cipher* dengan Iterasi)

   Iterated Block Cipher melakukan enkripsi terhadap suatu blok *plaintext* dengan suatu proses yang memiliki beberapa putaran (*round*). Dalam setiap putaran transformasi yang sama (juga dikenal dengan *round function*) diaplikasikan terhadap data dengan menggunakan subkunci. Himpunan subkunci biasanya diturunkan/ diperoleh dari kunci yang disediakan pemilik dengan suatu fungsi khusus. Himpunan subkunci disebut *key schedule*. Jumlah putaran dalam iterasi tergantung pada tingkat keamanan yang diinginkan dan konsekuensi adanya hubungan dengan efisiensi waktu. Secara umum, peningkatan jumlah putaran akan meningkatkan keamanan, tetapi untuk beberapa teknik jumlah putaran dibutuhkan untuk mencapai tingkat keamanan yang cukup akan terlalu sulit untuk membuatnya menjadi praktis atau diwujudkan sesuai dengan keinginan.

Feistel Cipher adalah bentuk khusus dari *Iterated Block Cipher* dimana *ciphertext* diperoleh dari *plaintext* dengan penggunaan berulang-ulang transformasi atau fungsi putaran yang sama. *Feistel Cipher* sering disebut dengan *DES-like ciphers*. Dalam *Feistel cipher*, teks yang sedang dienkrip dipecah menjadi 2 bagian yang sama panjang yaitu *Left* dan *Right*. Fungsi *Round* f diaplikasikan terhadap salah satu potongan dengan menggunakan suatu subkunci dan hasilnya di-XOR-kan dengan potongan yang lainnya. Kemudian kedua potongan kiri dan kanan dipertukarkan. Setiap putaran mengikuti pola yang sama, kecuali putaran terakhir dimana tidak ada lagi pertukaran data.



Gambar 2.2 Feistel Cipher

Salah satu sifat yang baik dari suatu *Feistel cipher* adalah bahwa enkripsi dan dekripsi secara struktur adalah identik, walaupun subkunci yang digunakan selama proses enkripsi pada setiap putaran digunakan dengan urutan terbalik sewaktu proses dekripsi.

Untuk lebih spesifik dapat dikatakan input dalam proses dekripsi adalah pasangan (Rr, Lr) bukan pasangan (L0, R0) seperti pada saat enkripsi (notasi seperti terlihat pada Gambar 2.2), dan subkunci ke-i adalah k_{r-i+1} bukan k_i. Ini berarti kita akan

memperoleh pasangan (R_{r-i}; L_{r-i}) jadi bukanlah pasangan (L_i; R_i) setelah putaran ke-i. Sebagai contoh, R_1 diganti dengan,

$R_r \oplus F(k_r; L_r) = R_r \oplus F(k_r; R_{r-1}) = R_r \oplus (R_r \oplus L_{r-1}) = L_{r-1}$;

sehingga diperoleh,

$L_i = R_{i-1}$ dan

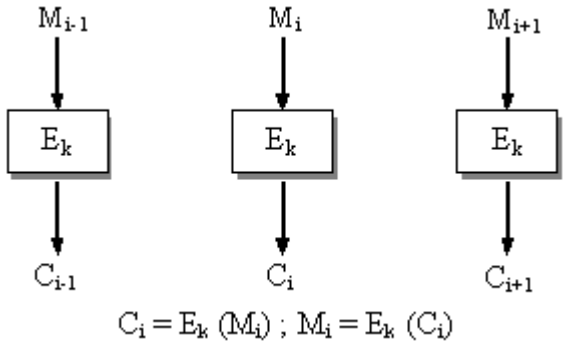
$R_i = L_{i-1} \oplus F(R_{i-1,}K_i)$; $\oplus$ adalah operator XOR.

Tentu saja dapat dirancang *iterative ciphers* yang bukan merupakan *Feistel ciphers*, tetapi proses enkripsi dan dekripsinya (setelah melakukan pengaturan ulang atau perhitungan ulang terhadap variabel-variabel yang terlibat) secara struktur adalah sama.

2. *Electronic Code Book (ECB) Mode*

   Dalam mode ECB (Gambar 2.3), setiap blok *plaintext* dienkripsi secara independen dengan blok *cipher*.

   Mode ECB memiliki tingkat keamanan sama seperti *block cipher*. Akan tetapi, pola *plaintext* tidak diberitahukan. Setiap blok *plaintext* yang indentik menghasilkan blok *ciphertext* yang identik. *Plaintext* dapat dengan mudah dimanipulasi dengan menghilangkan, mengulang atau saling mempertukarkan blok. Kecepatan masing-masing operasi enkripsi identik dengan kecepatan *block cipher*. ECB dapat dengan mudah melakukan proses paralel untuk menghasilkan performansi yang lebih tinggi. Akan tetapi, sangat disayangkan tidak ada proses sebelum sebuah blok dibaca (kecuali untuk pembentukan kunci).

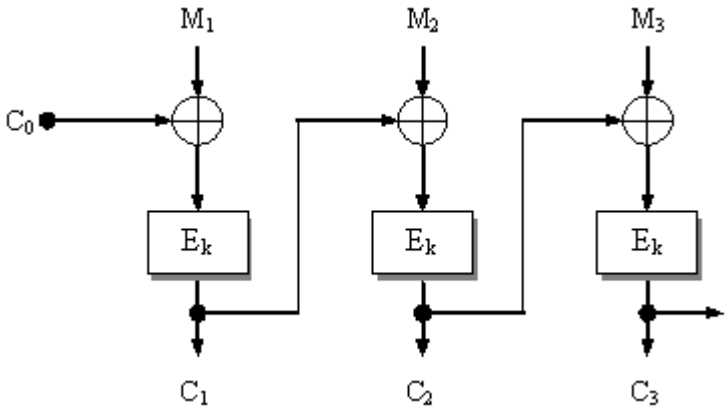


Gambar 2.3 Electronic Code Book Mode

3. *Cipher Block Chaining (CBC) Mode*

   Dalam mode CBC, setiap blok *plaintext* di-XOR-kan dengan blok *ciphertext* sebelumnya dan kemudian dienkripsi. Vektor inisialisasi C_0 digunakan sebagai bibit (seed) untuk proses enkripsi.

   Mode CBC memiliki tingkat keamanan yang setara dengan *block cipher* terhadap serangan standard. Sebagai tambahan, pola sembarang dalam *plaintext* disembunyikan dengan melakukan operasi XOR terhadap blok *ciphertext* sebelumnya dengan blok *plaintext*. Satu hal yang perlu dicatat bahwa *plaintext* tidak dapat dimanipulasi secara langsung dengan menghilangkan blok dari awal atau akhir dari *ciphertext*. Vektor inisialisasi harus berbeda untuk dua sembarang pesan yang di-enkrip dengan kunci sama dan lebih baik jika dipilih secara acak. Tidak ada keharusan untuk melakukan enkripsi dan dapat ditransmisikan dengan *ciphertext*.



$$C_i = E_k(C_{i-1} \oplus M_i) \; ; \; M_i = C_{i-1} \oplus D_k(C_i)$$

Gambar 2.4 Cipher Block Chaining Encryption Mode

Akan tetapi, perhatikan juga kelemahan yang dimiliki seperti kecepatan enkripsi sama dengan block cipher, tetapi proses enkripsi sulit untuk diparalelkan, walaupun proses dekripsi dapat diparalelkan.

Mode PCBC adalah variasi pada mode CBC dimana operasi dirancang untuk memperbesar atau menyebarkan satu bit error dalam *ciphertext*. Cara ini memungkinkan kesalahan dalam transmisi dapat ditangkap dan plaintext yang dihasilkan akan ditolak. Metode enkripsi dituliskan sebagai berikut ,

$$C_i = E_k \, (C_{i-1} \oplus M_{i-1} \oplus M_i)$$

Dan dekripsi dicapai dengan menghitung ,

$$M_i = Ci_{-1} \oplus M_{i-1} \oplus D_k \, (C_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.1)$$

Terdapat kesalahan/kelemahan dalam PCBC, yang mungkin dimaksudkan sebagai suatu contoh arahan (pengujian) dalam *cryptanalysis* tentang *block cipher*. Jika dua blok *ciphertext* C_{i-2} dan C_{i-1} dipertukarkan, maka hasil dari langkah ke-i

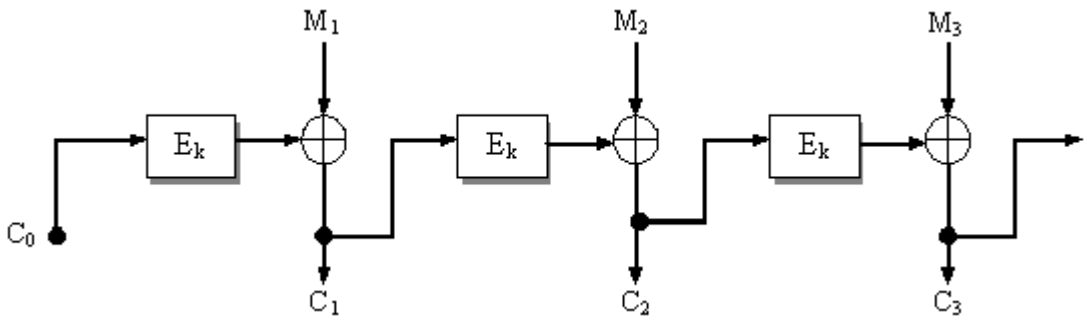
dalam proses dekripsi masih menghasilkan blok *plaintext* yang benar. Untuk lebih jelasnya, dengan persamaan (2.1) kita mendapatkan,

$$M_i = D_k (C_i \oplus (C_{i-1} \oplus D_k (C_{i-1})) \oplus D_k (C_{i-2})) \oplus C_{i-3} \oplus M_{i-3} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2.2)$$

Sebagai konsekuensinya, mempertukarkan dua blok berdekatan dari *ciphertext* (atau, lebih umum mengacak k blok berurutan dari *ciphertext*) tidak mempengaruhi apapun kecuali bahwa proses dekripsi blok *plaintext* yang bersangkutan. Walaupun konsekuensi praktis dari kesalahan tersebut tidak nampak dengan jelas, PCBC digantikan dengan mode CBC dalam Kerberos versi 5. Dalam kenyataannya, mode ini belum dipublikasikan secara formal sebagai suatu standar federal atau standar nasional (USA).

4. *Cipher Feedback (CFB) Mode*

   Dalam mode *Cipher Feedback* (CFB) blok *ciphertext* sebelumnya dienkripsi dan outputnya digabungkan dengan blok *plaintext* dengan menggunakan XOR untuk menghasilkan blok *ciphertext* sekarang. Kita dapat mendefinisikan mode tersebut sedemikian hingga mode tersebut menggunakan *feedback* yang lebih kecil dari 1 blok penuh data. Sebuah vektor inisialisasi C_0 digunakan sebagai sebuah "seed" untuk prosesnya, seperti terlihat pada Gambar 2.5.



$C_i = E_k (C_{i-1}) \oplus M_i$; $M_i = E_k (C_{i-1}) \oplus C_i$

Gambar 2.5 Cipher Feedback Mode

CFB *mode* memiliki tingkat keaamanan yang setara dengan *cipher* yang ada dan pola *plaintext* disembunyikan dalam *ciphertext* dengan menggunakan operasi XOR. *Plaintext* tidak dapat dimanipulasi secara langsung kecuali dengan menghilangkan blok dari awal atau akhir dari *ciphertext*. Dengan mode CFB dan *full feedback*, jika dua blok *ciphertext* adalah identik, output dari operasi blok *cipher* pada tahap berikutnya juga adalah identik. Hal ini memungkinkan informasi tentang blok *plaintext* bocor. Konsiderasi keamanan untuk vektor inisialisasi pada mode CFB sama seperti pada mode CBC, kecuali *attack* tidak berlaku. Akan tetapi blok terakhir dari blok *ciphertext* dapat diserang (*attacked*).

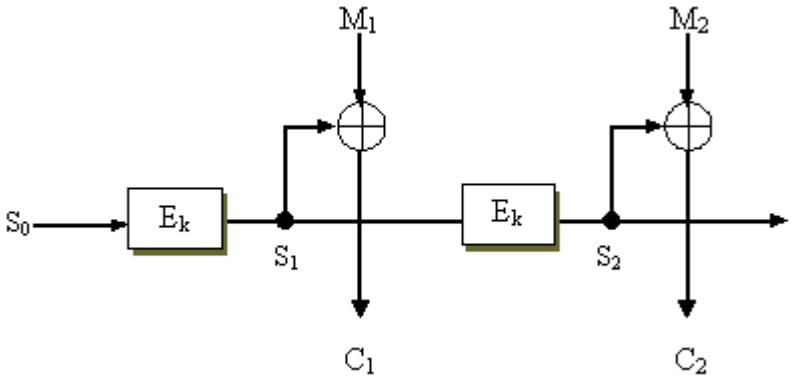
Dengan menggunakan umpan balik penuh (*full feedback*), kecepatan proses enkripsi sama seperti yang dimiliki metode *block cipher* yang lain, tetapi proses enkripsi sulit untuk dibuat paralel.

5. *Output Feedback (OFB) Mode*

Mode *Output Feedback* mirip dengan mode CFB kecuali bahwa jumlah operasi XOR dengan setiap blok *plaintext* dihasilkan secara independen dari *plaintext*

maupun *ciphertext*. Sebuah vektor inisialisasi C_0 digunakan sebagai suatu "*seed*" untuk sebarisan blok data S_i, dan setiap blok data S_i diperoleh dari proses enkripsi terhadap blok data S_{i-1} sebelumnya. Proses enkripsi blok *plaintext* diperoleh dengan melakukan operasi XOR antara blok *plaintext* dengan blok data yang relevan.

Lebar umpan balik lebih kecil dari blok penuh tidak direkomendasikan untuk alasan keamananan. Mode OFB memiliki kelebihan dibandingkan dengan mode CFB dimana setiap bit error yang mungkin terjadi selama transmisi tidak ikut dikirimkan untuk mempengaruhi proses dekripsi blok berurutan. Konsiderasi keamanan untuk vektor inisialisasi sama seperti pada mode CFB. Satu masalah dengan mode OFB adalah bahwa *plaintext* dapat dimanipulasi. Yakni, seorang *attacker* yang mengetahui satu blok *plaintext* M_i dapat menggantikannya dengan *plaintext* x yang salah atau dengan melakukan x XOR M_i terhadap blok *ciphertext* C_i yang terkait.



$C_i = M_i \oplus S_i$; $M_i = C_i \oplus S_i$; $S_i = E_k (S_{i-1})$

Gambar 2.6 Output Feedback Mode

Terdapat serangan yang mirip terhadap mode CBC dan CFB, tetapi serangan tersebut beberapa blok *plaintext* akan dimodifikasi dalam suatu cara yang tidak dapat diprediksi oleh penyerang. Tetapi blok *ciphertext* paling depan (vektor inisialisasi) dalam mode CBC dan blok terakhir *ciphertext* dalam CFB sangat lemah terhadap serangan seperti blok dalam mode OFB. Serangan jenis ini dapat dicegah dengan menggunakan misalnya skema tanda tangan digital (*digital signature scheme*) atau skema MAC.

Kecepatan proses enkripsi sama seperti yang dimiliki oleh *block cipher* yang lain. Namun proses tidak dapat diparalelkan, waktu bisa dihemat dengan menghasilkan aliran kunci (*keystream*) sebelum data ada untuk proses enkripsi. Untuk mengatasi kelemahan yang dimiliki mode OFB, Diffie telah mengusulkan penambahan mode operasi, yang disebut dengan mode *counter*. Mode ini berbeda dengan mode OFB dalam cara blok data yang berkelanjutan dihasilkan untuk enkripsi berturut-turut. Dengan tidak menurunkan satu blok data sebagaimana enkripsi untuk blok data sebelumnya, *Diffie* mengusulkan mengenkripsi jumlah i + IV mod 2^{64} untuk blok data ke-i, dimana IV adalah Vektor Inisialisasi (*Initial Vector*).

2.2.1.2 Stream Cipher

Stream cipher adalah jenis algoritma enkripsi simetri yang mentransformasikan data secara karakter per karakter. *Stream ciphers* dapat dibuat

sangat cepat sekali, jauh lebih cepat dibandingkan dengan algoritma *block cipher* yang manapun. Sementara algoritma *block cipher* secara umum digunakan untuk unit *plaintext* yang berukuran besar sedangkan *stream cipher* digunakan untuk blok data yang lebih kecil, biasanya ukuran bit. Proses enkripsi terhadap *plaintext* tertentu dengan algoritma *block cipher* akan menghasilkan *ciphertext* yang sama jika kunci yang sama digunakan. Dengan *stream cipher*, transformasi dari unit *plaintext* yang lebih kecil ini berbeda antara satu dengan lainnya, tergantung pada kapan unit tersebut ditemukan selama proses enkripsi.

Suatu *stream cipher* akan menghasilkan apa yang disebut suatu *keystream* yaitu suatu barisan bit yang digunakan sebagai kunci. Proses enkripsi dicapai dengan menggabungkan *keystream* dengan *plaintext* biasanya dengan operasi bitwise XOR. Pembentukan *keystream* dapat dibuat independen terhadap *plaintext* dan *ciphertext,* menghasilkan apa disebut dengan *synchronous stream cipher*, atau dapat dibuat tergantung pada data dan enkripsinya, dalam hal mana *stream cipher* disebut sebagai *self-synchronizin*g. Kebanyakan bentuk *stream cipher* adalah *synchronous stream cipher*s.

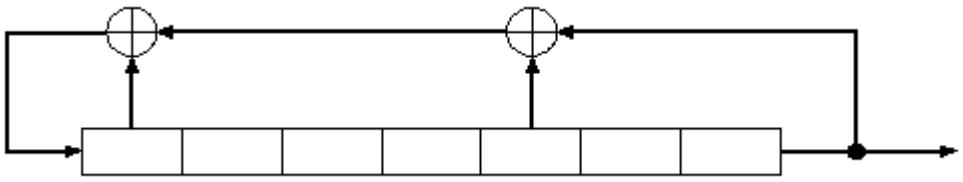
Konsentrasi dalam *stream ciphers* pada umumnya berkaitan dengan sifat-sifat teoritis yang menarik dari *one-time pad*. Suatu *one-time pad*, kadang-kadang disebut *Vernam cipher,* menggunakan sebuah string dari bit yang dihasilkan murni secara acak. *Keystream* memiliki panjang sama dengan pesan *plaintext;* string acak digabungkan dengan menggunakan bitwise XOR dengan *plaintext* untuk

menghasilkan *ciphertext*. Karena *keystream* seluruhnya adalah acak, walaupun dengan sumber daya komputasi tak terbatas seseorang hanya dapat menduga *plaintext* jika dia melihat *ciphertext*. Metode *cipher* seperti ini disebut memberikan kerahasiaan yang sempurna (*perfect secrecy*), dan analisis terhadap *one-time pad* dipandang sebagai salah satu landasan kriptografi modern. Sementara *one-time pad* yang digunakan semasa perang melalui saluran diplomatik membutuhkan tingkat keamanan yang sangat tinggi, fakta bahwa kunci rahasia (yang hanya dapat digunakan satu kali) dianggap rahasia sepanjang pesan memperkenalkan masalah manajemen kunci yang *strict*. Sedangkan keamanan sempurna, *one-time pad* secara umum adalah tidak praktis.

Stream ciphers dikembangkan sebagai satu aproksimasi terhadap tindakan dari *one-time pad*. Sementara *stream cipher* modern tidak mampu menyediakan tingkat keamanan *one-time pad* yang memadai secara teori, tetapi setidaknya praktis. Sampai saat ini belum ada *stream cipher* sebagai standard secara de facto. Metode *stream cipher* yang umum digunakan adalah RC4. Satu hal yang menarik bahwa mode operasi tertentu dari suatu *block cipher* dapat mentransformasikan secara efektif hasil operasi tersebut ke dalam satu *keystream generator* dan dalam hal ini, *block cipher* apa saja dapat digunakan sebagai suatu *stream cipher*; seperti dalam DES, CFB atau OFB. Akan tetapi, *stream ciphers* dengan desain khusus biasanya jauh lebih cepat. Berikut ini adalah jenis – jenis dari *stream cipher*,

1. *Linear Feedback Shift Register*

Suatu *Linear Feedback Shift Register* (LFSR) adalah suatu mekanisme untuk menghasilkan sekuens bit biner. Register (Gambar 2.7) memiliki sebarisan sel yang ditentukan oleh vektor inisialisasi yakni, biasanya, menjadi kunci rahasia. Tingkah laku register diatur oleh sebuah *counter* (*clock*). Pada setiap saat isi sel dari register digeser (*shift*) ke kanan sejauh satu posisi, dan hasil operasi XOR terhadap subset dari isi sel ditempatkan pada sel paling kiri. Satu bit dari output biasanya diturunkan selama prosedur *update* ini.



Gambar 2.7 Linear Feedback Shift Register (LFSR)

LFSR cepat dan mudah untuk diimplementasikan baik secara *hardware* maupun secara *software*. Dengan pemilihan yang cermat terhadap alat kontrol (*pad*) *feedback* (bit tertentu yang digunakan, dalam Gambar 2.7 di atas *tap* adalah bit pertama dan bit kelima) sekuens yang dihasilkan dapat memiliki performansi yang baik secara statistik. Akan tetapi, sekuens yang dihasilkan oleh LFSR tunggal tidak cukup aman karena sebuah kerangka matematik yang kuat telah dikembangkan selama bertahun-tahun yang memungkinkan analisis yang mudah terhadap hasil tersebut. Namun demikian, LFSR sangat berguna sebagai blok bangunan dalam sistem yang lebih aman.

Suatu *shift register cascade* merupakan sebuah set dari LFSR yang dihubungkan bersama-sama dengan cara tertentu sedemikian hingga perlakuan dari LFSR sebelumnya dalam bentuk aliran yang menyerupai air terjun. Sifat dan tingkah laku yang saling tergantung ini biasanya ditempuh dengan menggunakan satu LFSR untuk mengontrol LFSR berikutnya. Sebagai contoh, satu register mungkin melangkah lebih dulu satu fase jika output register di depannya adalah 1 dan maju dua langkah jika hasil = 0. Beberapa konfigurasi yang berbeda mungkin dan pemilihan parameter tertentu nampaknya menawarkan tingkat keamanan yang baik.

The shrinking generator dikembangkan oleh Coppersmith, Krawczyk, and Mansour. Ini adalah suatu *stream cipher* yang didasarkan pada interaksi sederhana antara output dari dua LFSR. Bit-bit dari satu output digunakan untuk menentukan apakah bit yang terkait dengan output register kedua akan digunakan sebagai bagian dari *keystream* secara keseluruhan. *The shrinking generator* adalah sederhana dan dapat diukur, dan memiliki sifat-sifat keamanan yang baik. Salah satu kelemahan dari *the shrinking generator* adalah bahwa tingkat output *keystream* tidak akan konstan lain hal jika proteksi dilakukan. Salah satu varian dari *shrinking generator* adalah *the self-shrinking generator*, dimana dalam proses kerjanya ia tidak menggunakan satu output dari salah satu LFSR melainkan ia melakukan penyusutan output dari register yang satunya (seperti pada *the shrinking generator*), output dari LFSR tunggal digunakan untuk mengekstraksi bit-bit dari output yang sama. Belum ditemukan hasil kriptanalisis terhadap kedua teknik.

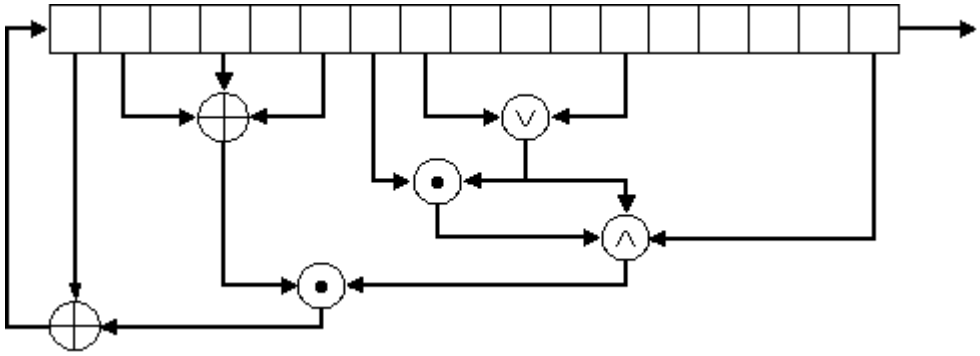
2. *Non Linear Feedback Shift Register (NLFSR)*

Mudah untuk membayangkan suatu rangkaian umpan balik (*feedback*) yang lebih rumit daripada yang digunakan LFSRs atau FCSRs. Masalahnya adalah tidak terdapat suatu teori matematika untuk menganalisanya. Secara khusus, beberapa masalah dengan rangkaian umpan balik *shift register nonlinear* adalah ,

a. Mungkin terdapat prasangka (bias), seperti terdapat lebih banyak 1 daripada 0 atau lebih sedikit langkah daripada yang diharapkan pada keluaran.
b. Periode maksimum dari rangkaian mungkin jauh lebih rendah daripada yang diharapkan.
c. Periode rangkaian dapat berbeda untuk nilai awal yang berbeda.
d. Rangkaian dapat muncul secara acak untuk sesaat, tetapi kemudian terjadi "*dead end*" pada suatu nilai tunggal. Ini dapat diselesaikan dengan meng-XOR-kan fungsi *nonlinear* dengan bit paling kanan.

Sebaliknya, jika tidak ada teori untuk menganalisis sisi keamanan umpan balik *nonliniear shift register*, terdapat sedikit perangkat (*tool*) kriptanalisis *stream cipher* yang didasarkan padanya. Kita dapat menggunakan umpan balik *nonlinear shift register* pada rancangan *stream cipher*, tetapi kita harus berhati-hati.

Dalam sebuah umpan balik *nonlinear shift register*, fungsi umpan balik dapat berupa apa saja yang diinginkan yang terlihat pada Gambar 2.8 berikut ini.



Gambar 2.8 Non Linear Feedback Shift Register

2.2.1.3 Desain Cipher

Terdapat dua prinsip dasar untuk menghasilkan *cipher* yang aman, yaitu *confusion* dan *diffusion*. Tanpa memperhatikan hal ini, *cipher* kita mungkin akan sangat mudah dipecahkan sandinya.

Confusion berarti mengaburkan hubungan antara *plaintext* dan *ciphertext*. Ini akan membuat frustasi usaha untuk mencari keteraturan dan pola statistik antara *plaintext* dan *ciphertext*. Cara paling mudah untuk melakukan hal ini adalah dengan substitusi. Substitusi modern menggunakan cara yang sangat komplek. Namun cara ini belum cukup. *Cipher* Jerman, *Enigma*, yang menggunakan algoritma substitusi yang komplek dipecahkan oleh Sekutu dalam perang dunia kedua.

Diffusion berarti menghilangkan redundansi *plaintext* dengan menyebarkan masukan ke seluruh *ciphertext*. Diperlukan waktu yang lebih lama untuk memecahkan sandi rahasia ini, bila *diffusion* digunakan. Cara paling mudah untuk melakukan *diffusion* adalah transposisi atau permutasi.

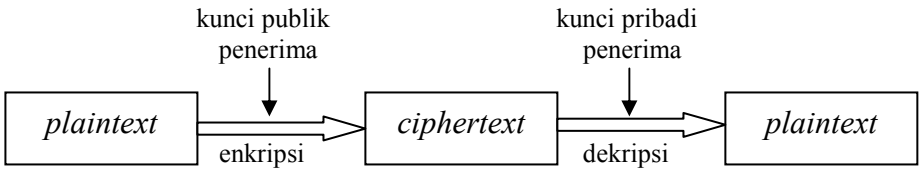
Dalam dunia kriptografi modern, *confusion* dan *diffusion* ini dilakukan secara sangat intensif dengan bantuan komputer.

2.2.2 Kriptografi Kunci Publik (*Public Key Cryptography*)

Kriptografi kunci publik diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1976. Kriptografi kunci publik memiliki dua penggunaan utama, yakni enkripsi dan tanda tangan digital (*encryption and digital signatures*). Dalam sistem kriptografi kunci publik, masing-masing pihak mendapat sepasang kunci, satu disebut kunci publik (*public key*) dan satu lagi disebut kunci rahasia (*private key*). Kunci publik dipublikasikan, sementara kunci rahasia tetap dirahasiakan. Keharusan penggunaan kunci secara bersama antara pengirim dan penerima pesan rahasia dihilangkan, semua komunikasi hanya melibatkan kunci publik, dan tidak ada kunci rahasia yang ditransmisikan atau digunakan bersama. Dalam sistem ini, tidak ada lagi kecurigaan terhadap keamanan dari sistem komunikasi. Satu-satunya kebutuhan bahwa kunci publik dikaitkan dengan penggunanya dalam lingkup yang saling mempercayai (contoh dalam suatu *trusted directory*). Seseorang dapat mengirimkan pesan rahasia dengan hanya menggunakan informasi yang umum (kunci publik), tetapi pesan tersebut hanya mungkin didekrip dengan menggunakan kunci rahasia, dimana satu-satunya yang memiliki kunci rahasia tersebut hanyalah orang yang diharapkan menerima pesan tersebut. Kriptografi kunci publik tidak hanya digunakan

untuk merahasiakan pesan, tetapi juga untuk otentikasi (tanda tangan digital) dan teknik lainnya.

Dalam kriptografi kunci publik, kunci rahasia selalu berhubungan secara matematis terhadap kunci publik. Oleh karena itu, selalu dimungkinkan untuk menembus (menyerang) sistem kunci publik dengan menurunkan kunci rahasia dari kunci publik. Biasanya, cara untuk menangkal kemungkinan tersebut adalah membuat sesulit mungkin untuk menghasilkan kunci privat dari kunci publik. Sebagai contoh, beberapa kriptosistem kunci publik dibuat sedemikian hingga penurunan kunci rahasia (privat) dari kunci publik mengharuskan penyerang melakukan faktorisasi terhadap bilangan yang sangat besar, dalam hal ini sangat sulit untuk melakukan penurunan. Inilah ide di belakang RSA *public-key cryptosystem*.



Gambar 2.9 Model Sederhana dari Public Key Cryptography

2.3 Cryptanalysis

2.3.1 Definisi Cryptanalysis

Cryptanalysis adalah sisi lain dari *cryptography*, yang merupakan ilmu untuk memecahkan kode, mendecode rahasia, mematahkan skema otentikasi, dan secara umum menembus dan memecahkan protokol *cryptography*. Agar diperoleh algoritma

enkripsi atau protokol *cryptography* yang handal (robust), kita harus menggunakan *cryptanalysis* untuk mencari dan memperbaiki kelemahan yang dimiliki. Inilah alasan sebenarnya mengapa algoritma enkripsi yang paling dipercaya adalah algoritma yang telah dibuat publik. Sebagai contoh, DES telah diberikan ke publik selama bertahun-tahun, dan oleh karena itu telah dipercaya sepenuhnya (well-trusted), sementara Skipjack merupakan proyek rahasia untuk waktu yang lama dan dengan demikian kurang dipercaya (less-trusted).

Adalah merupakan prinsip dasar dari *cryptology* dimana keamanan dari suatu algoritma seharusnya tidak tergantung pada kerahasiaannya. Satu hal yang tidak dapat dihindarkan bahwa algoritma tersebut akan ditemukan dan kelemahannya (jika ada) akan dipaparkan ke khayalak ramai. Berbagai teknik dalam *cryptanalysis* mencoba melakukan kompromi terhadap *cryptosystems* yang dianggap sebagai penyerangan (attack). Sebagian ancaman (serangan) adalah bersifat umum, sementara yang lainnya hanya berlaku pada jenis *cryptosystems* tertentu.

2.3.2 Bentuk dasar dari cryptanalytic attack

Cryptanalytic attacks umumnya diklasifikasikan ke dalam enam kategori yang membedakan informasi yang dibutuhkan oleh *cryptanalyst* untuk dapat melakukan serangan.

1. *Ciphertext-only attack*

   adalah serangan dimana *cryptanalyst* mendapatkan contoh dari *ciphertext*, tanpa *plaintext* yang berhubungan dengan *ciphertext* yang dimilikinya. Data ini relatif mudah untuk diperoleh dalam beberapa skenario, tetapi suatu serangan *ciphertext-only* yang berhasil umumnya sulit, dan membutuhkan contoh ciphertext yang sangat besar.

2. *Known-plaintext attack*

   adalah serangan dimana *cryptanalyst* medapatkan sampel dari *ciphertext* dan *plaintext* terkait..

3. C*hosen-plaintext attack*

   adalah serangan dimana *cryptanalyst* dapat memilih sebagian *plaintext* dan kemudian mendapatkan *ciphertext* terenkripsi terkait.

4. *Adaptive-chosen-plaintext attack*

   adalah kasus khusus dari *chosen-plaintext attack* dimana *cryptanalyst* dapat memilih sampel *plaintext* secara dinamis, dan mengubah pilihannya berdasarkan hasil dari enkripsi sebelumnya.

5. *Chosen-ciphertext attack*

   adalah serangan dimana *cryptanalyst* boleh memilih potongan *ciphertext* dan mencoba untuk mendapatkan *plaintext* yang didekripsi. Jenis serangan ini umumnya paling dapat digunakan untuk *public-key cryptosystems*.

6. *Adaptive-chosen-ciphertext*

   adalah versi adaptif dari *chosen-ciphertext attack* . Seorang *cryptanalyst* dapat melancarkan serangan jenis ini dalam suatu skenario dimana dia memiliki penggunaan bebas terhadap hardware untuk dekripsi, tetapi tidak dapat mengekstraksikan kunci dekripsi dari hardware tersebut.

Tujuan dari *cryptanalyst* dalam semua kasus adalah untuk melakukan dekripsi potongan baru dari *ciphertext* tanpa informasi tambahan. Kondisi ideal bagi seorang *cryptanalyst* adalah mendapatkan kunci rahasia. *Cryptanalytic attacks* dapat dilancarkan bukan hanya terhadap algoritma enkripsi, tetapi juga terhadap algoritma tanda tangan digital, Algoritma MACing dan *pseudo-random number generators*.

2.3.3 Exhaustive key search

Exhaustive key search, atau *brute-force search*, adalah teknik dasar dari mencoba setiap kemungkinan kunci secara berturut-turut hingga kunci yang benar ditemukan. Untuk mengidentifikasikan kunci yang benar mungkin perlu untuk memiliki *plaintext* dan *ciphertext* terkait atau jika *plaintext* mengandung beberapa karakter yang dikenal, *ciphertext* sendiri mungkin cukup. *Exhaustive key search* dapat dilakukan pada sembarang *cipher* dan kadangkala kelemahan dalam penjadwalan kunci dari *cipher* dapat membantu untuk meningkatkan efisiensi dari *exhaustive key search attack*. *Exhaustive key search* dapat juga dilakukan dalam program yang dijalankan dengan workstation standard atau dalam PC.

Kemajuan dalam teknologi dan performansi komputasi akan selalu membuat *exhaustive key search* menjadi ancaman serius terhadap kunci panjang tertentu. Ketika DES dirancang, algoritma ini dianggap aman terhadap *exhaustive key search* tanpa investasi besar dalam hardware. Akan tetapi, setelah beberapa tahun berlalu jenis serangan terhadapnya menjadi hal yang menarik terhadap musuh potensial. *Exhaustive search* untuk DES dengan kunci 56-bit akan membutuhkan puluhan atau ratusan tahun dengan *general purpose computer* paling cepat yang ada saat ini, perkembangan internet telah memungkinkan untuk menggunakan ribuan mesin dalam mode terdistribusi dengan melakukan partisi kunci dan mendistribusikan bagian kecil kunci ke masing-masing komputer dalam jumlah besar. Dengan cara ini dan dengan menggunakan supercomputer yang dirancang khusus, suatu kunci untuk DES memang dapat dipecahkan dalam waktu 22 jam pada bulan Januari 1999. Tingkat kecepatan dalam peningkatan kemampuan komputasi sedemikian hingga kunci dengan panjang 80-bit harus memberikan level keamanan yang dapat diterima untuk selama 10 atau 15 tahun ke depan.Tidak adanya terobosan berarti dalam komputasi kuantum, adalah suatu hal yang mustahil untuk memecahkan dengan cara *exhaustive search* kunci 128-bit, seperti yang digunakan dalam IDEA dan AES.

2.4 Landasan Matematis Kriptografi

2.4.1 Aritmatika Modular

Aritmatika modular merupakan operasi matematika yang banyak diimplementasikan pada metode kriptografi. Pada metode MMB, operasi aritmatika modular yang dipakai adalah operasi perkalian modulo $2^{32} - 1$. Operasi modulo ini melibatkan bilangan 0 dan 1 saja sehingga identik dengan bit pada komputer.

Contoh :

(12457865 * 12456) mod ($2^{32} - 1$) = 155175166440 mod 4294967295 = 556343820

2.4.2 Inverse Perkalian

Inverse perkalian yang digunakan pada metode MMB tidak seperti *inverse* pada operasi perkalian dalam matematika. *Inverse* perkalian ini menggunakan algoritma yang hampir sama dengan *inverse* perkalian pada metode IDEA. Perbedaannya hanya terletak pada besar nilai modulonya saja. Pada metode IDEA, digunakan aritmatika modulo $2^{16} + 1$ sedangkan pada metode MMB digunakan aritmatika modulo $2^{32} - 1$. *Inverse* perkalian ini tidak dapat dijelaskan secara matematis, tetapi dengan menggunakan algoritma berikut ini :

```
Fungsi Inverse(A)
    n = 4294967295
    G0 ← n
    G1 ← A
    V0 ← 0
    V1 ← 1
    Ketika (G1 <> 0)
```

```
        Y ← Int(G0 / G1)
        G2 ← G0 - Y * G1
        G0 ← G1
        G1 ← G2
        V2 ← V0 - Y * V1
        V0 ← V1
        V1 ← V2
    End Ketika
    Jika (V0 >= 0) Maka
        Inverse ← V0
    Jika tidak,
        Inverse ← V0 + n
    End Jika
End Fungsi
```

Algoritma ini hanya diimplementasikan pada waktu mencari besar konstanta c_0 untuk proses dekripsi. Pada proses enkripsi c_0 yang digunakan bernilai sebesar $(025F1CDB)_{16}$, maka nilai c_0 yang digunakan pada proses dekripsi adalah sebesar $(0DAD4694)_{16}$.

2.4.3 Operasi XOR

XOR adalah operasi *Exclusive*-OR yang dilambangkan dengan tanda "⊕". Operasi XOR akan menghasilkan nilai bit "0" (nol) jika meng-XOR-kan dua buah bit yang sama nilainya dan akan menghasilkan nilai bit "1" (satu) jika meng-XOR-kan dua buah bit yang masing – masing nilai bitnya berbeda. Aturan yang berlaku untuk operasi XOR dapat dilihat pada Tabel 2.1 berikut ini,

Tabel 2.1 Aturan Operasi XOR

A	**B**	**A ⊕ B**
0	0	0
0	1	1
1	0	1
1	1	0

Nilai A jika di-XOR-kan dengan nilai B sebanyak dua kali maka akan didapatkan nilai A kembali. Karena sifat istimewa yang dimiliki operasi XOR tersebut sehingga operasi XOR cenderung dipakai dalam proses enkripsi dan dekripsi yang memiliki algoritma yang sama.

$$P \oplus K = C \; ; \; C \oplus K = P$$

Keterangan,

P = *Plaintext*

K = *Key*

C = *Ciphertext*

Berikut ini adalah contoh operasi XOR :

```
1101 0110 0001 0100
1000 0001 1110 0000 ⊕
---------------------
0101 0111 1111 0100
```

2.4.4 Sifat-Sifat Operasi XOR

Operasi XOR ini memiliki sifat-sifat sebagai berikut :

1. Sifat komutatif.

   Hasil dari operasi XOR akan bernilai sama walaupun letak kedua operand yang akan dilakukan operasi dibalik urutannya. Hal ini membuktikan bahwa operasi XOR mendukung sifat komutatif. Sifat ini dapat dirumuskan dalam pernyataan berikut,

   $$A \oplus B = B \oplus A$$

2. Sifat asosiatif.

   Untuk operand yang lebih dari dua buah, hasil dari operasi XOR untuk berbagai macam kombinasi tanda kurung tetap akan bernilai sama. Hal ini membuktikan bahwa operasi XOR mendukung sifat asosiatif. Untuk tiga buah operand, sifat ini dapat dirumuskan sebagai berikut :

   $$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

2.4.5 Fungsi Linier

Fungsi linier adalah sebuah fungsi dimana variabel *input* x bertambah dengan pemangkatan 1 dalam setiap persamaan. Fungsi ini akan membentuk grafik yang berupa garis lurus dan oleh karena itu, dinamakan linier.

Secara umum, sebuah fungsi linier adalah sebuah fungsi yang dapat didefinisikan dengan menggunakan bentuk persamaan seperti berikut :

$$f(x) = y = mx + b$$

2.4.6 Fungsi Non Linier

Non linier berarti bahwa grafik yang dibentuk bukan berupa garis lurus. Sesuai dengan pengertian tersebut, maka fungsi non linier adalah sebuah fungsi dimana pertambahan variabel inputnya bukan berupa pemangkatan 1 dan grafik fungsinya tidak membentuk garis lurus. Fungsi-fungsi yang termasuk fungsi non-linier antara lain fungsi logaritma, exponential, aritmatika modular, perkalian, parabola (fungsi kuadrat) dan sebagainya.

2.5 Metode MMB

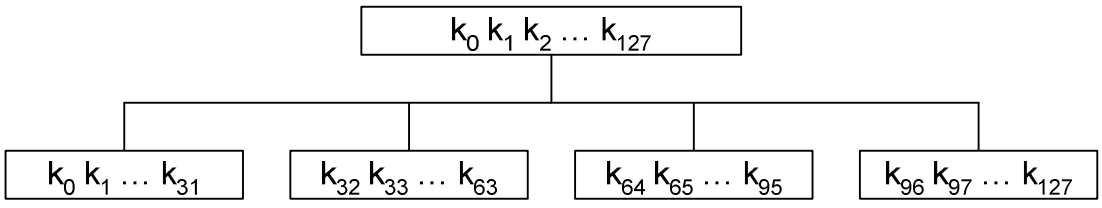
Kelemahan metode IDEA yang menggunakan *plaintext* 64 bit dan operasi perkalian modulo $2^{16} + 1$, diperbaiki oleh Joan Daemen dalam sebuah algoritma yang dinamakan MMB (*Modular Multiplication-based Block cipher*). Dengan menggunakan *plaintext* 64 bit (4 buah 16 bit *subblock text*), metode IDEA hanya dapat diimplementasikan pada prosesor 16 bit, sehingga dinilai tidak dapat mengikuti perkembangan teknologi pada saat ini yang kebanyakan telah menggunakan prosesor 32 bit. Kriptografi metode MMB menggunakan *plaintext* 128 bit dan algoritma iteratif yang terdiri dari langkah-langkah linier (seperti XOR dan aplikasi kunci) serta

aplikasi paralel dari empat substitusi non linier besar yang dapat dibalik. Substitusi ini ditentukan oleh sebuah operasi perkalian modulo $2^{32} - 1$ dengan faktor konstan, yang memiliki tingkat sekuritas lebih tinggi bila dibandingkan dengan metode IDEA yang hanya menggunakan operasi perkalian modulo $2^{16} + 1$. MMB menggunakan 32 bit *subblock text* (x_0, x_1, x_2, x_3) dan 32 bit *subblock* kunci (k_0, k_1, k_2, k_3). Hal ini membuat algoritma tersebut sangat cocok diimplementasikan pada prosesor 32 bit. Sebuah fungsi non linier, f, diterapkan enam kali bersama dengan fungsi XOR.

2.5.1 Pembentukan Kunci

Metode MMB menggunakan kunci sepanjang 128 bit. Proses pembentukan kunci pada metode MMB ini sangat sederhana. Kunci yang di-*input* hanya dibagi menjadi 4 buah *subblock* kunci dengan panjang masing-masing 32 bit.

Proses pembentukan kunci pada metode MMB ini dapat dilihat pada bagan berikut ini :



Gambar 2.10 Proses Pembentukan Kunci pada Metode MMB

2.5.2 Enkripsi

Metode MMB menggunakan *plaintext* dan kunci dengan panjang 128 bit. Inti proses enkripsi dari metode MMB adalah sebagai berikut :

1. *Plaintext* dibagi menjadi 4 *subblock* yang sama besar (x_0, x_1, x_2, x_3).
2. Lakukan proses berikut ini sebanyak 2 kali : (Index dilakukan operasi modulo 4)

    for i = 0 to 3

        $x_i = x_i$ XOR k_i

    next i

    $f(x_0,x_1,x_2,x_3)$

    for i = 0 to 3

        $x_i = x_i$ XOR k_{i+1}

    next i

    $f(x_0,x_1,x_2,x_3)$

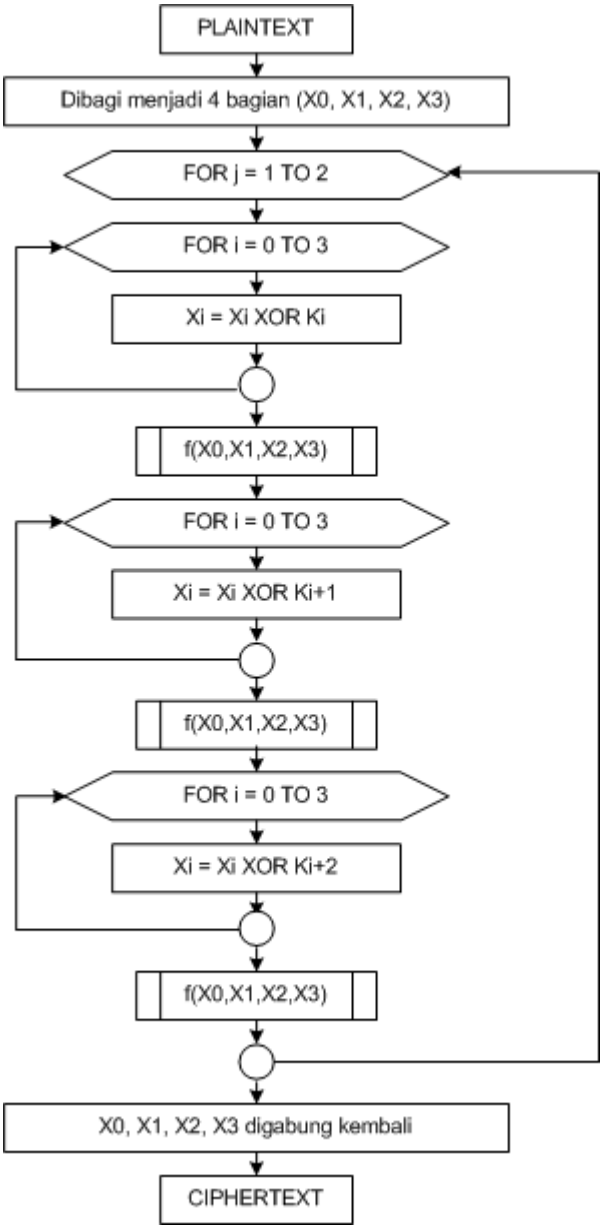
    for i = 0 to 3

        $x_i = x_i$ XOR k_{i+2}

    next i

    $f(x_0,x_1,x_2,x_3)$

Proses enkripsi tersebut dapat ditunjukkan dalam bentuk bagan seperti berikut ini :



Gambar 2.11 Proses Enkripsi pada Metode MMB

Fungsi f yang digunakan memiliki 3 langkah yaitu :

- for i = 0 to 3

  $x_i = c_i * x_i$

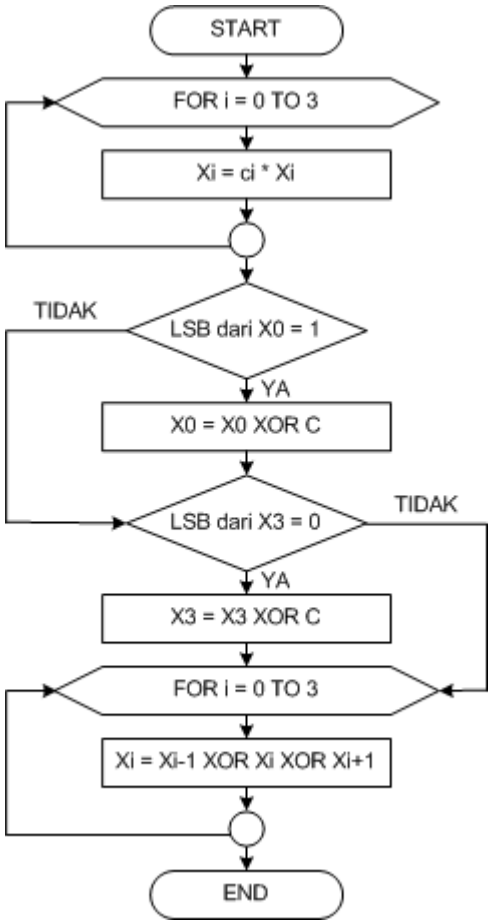
next i

- Jika LSB (*Least Significant Bit*) dari $x_0 = 1$, maka $x_0 = x_0$ XOR C. Jika LSB dari $x_3 = 0$, maka $x_3 = x_3$ XOR C.
- for i = 0 to 3

  $x_i = x_{i-1}$ XOR x_i XOR x_{i+1}

  next i

Fungsi f diatas dapat digambarkan dalam bentuk bagan seperti berikut ini :



Gambar 2.12 Fungsi f pada Proses Enkripsi Metode MMB

Operasi perkalian yang digunakan merupakan operasi perkalian modulo $2^{32} - 1$. Sedangkan konstanta yang digunakan dapat dirincikan sebagai berikut :

- $C = (2AAAAAAA)_{16}$
- $c_0 = (025F1CDB)_{16}$
- $c_1 = 2 * c_0$
- $c_2 = 2^3 * c_0$
- $c_3 = 2^7 * c_0$

2.5.3 Dekripsi

Algoritma yang digunakan pada proses dekripsi agak sedikit berbeda dengan proses enkripsi. Inti proses dekripsi dari metode MMB dapat dijabarkan seperti berikut :

1. *Ciphertext* dibagi menjadi 4 *subblock* yang sama besar (x_0, x_1, x_2, x_3).
2. Lakukan proses berikut ini sebanyak 2 kali : (Index dilakukan operasi modulo 4)

   $f(x_0,x_1,x_2,x_3)$

   for i = 0 to 3

   $x_i = x_i$ XOR k_{i+2}

   next i

   $f(x_0,x_1,x_2,x_3)$

   for i = 0 to 3

   $x_i = x_i$ XOR k_{i+1}

next i

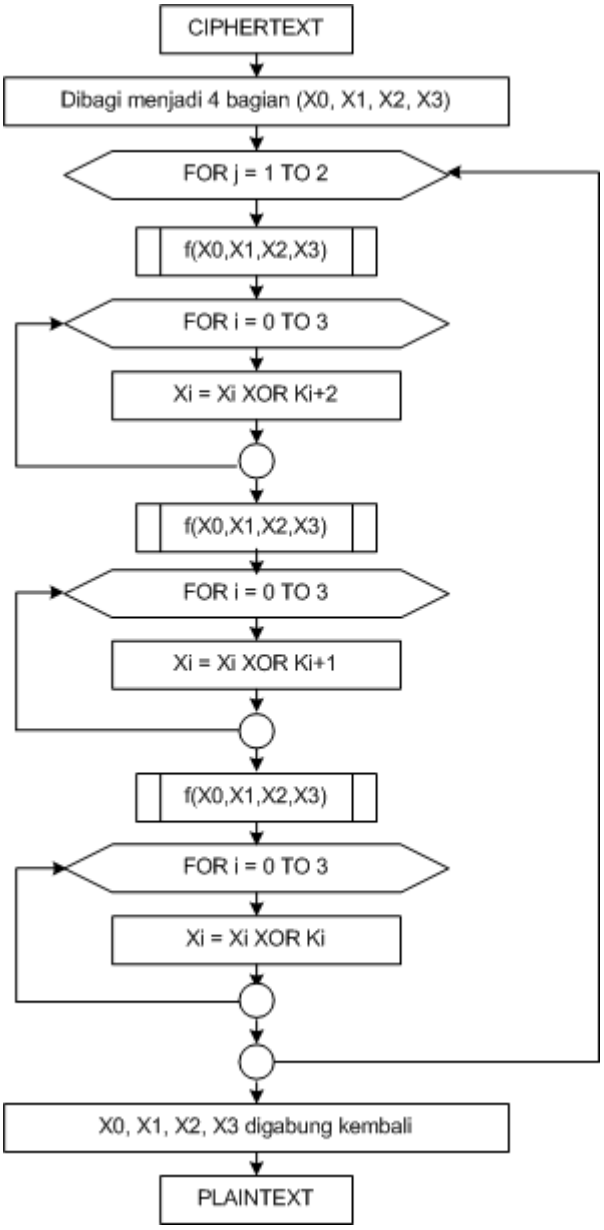
$f(x_0,x_1,x_2,x_3)$

for i = 0 to 3

$x_i = x_i$ XOR k_i

next i

Proses dekripsi tersebut dapat ditunjukkan dalam bentuk bagan seperti berikut ini :



Gambar 2.13 Proses Dekripsi pada Metode MMB

Fungsi f yang digunakan juga berbeda dan dapat dijabarkan seperti berikut :

- for i = 3 to 0 step -1

  $x_i = x_{i-1}$ XOR x_i XOR x_{i+1}

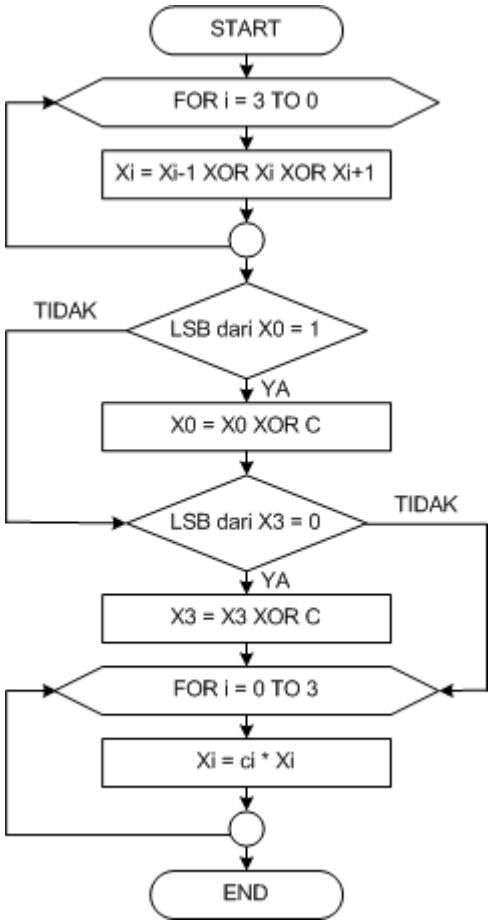
next i

- Jika LSB (*Least Significant Bit*) dari $x_0 = 1$, maka $x_0 = x_0$ XOR C. Jika LSB dari $x_3 = 0$, maka $x_3 = x_3$ XOR C.
- for i = 0 to 3

  $x_i = c_i * x_i$

  next i

Fungsi f diatas dapat digambarkan dalam bentuk bagan seperti berikut ini :



Gambar 2.14 Fungsi f pada Proses Dekripsi Metode MMB

Operasi perkalian yang digunakan merupakan operasi perkalian modulo $2^{32} - 1$. Sedangkan konstanta yang digunakan dapat dirincikan sebagai berikut :

- $C = (2AAAAAAA)_{16}$
- $c_0^{-1} = (0DAD4694)_{16}$
- $c_1^{-1} = 2^{-1} * c_0^{-1}$
- $c_2^{-1} = 2^{-3} * c_0^{-1}$
- $c_3^{-1} = 2^{-7} * c_0^{-1}$

2.5.4 Perbandingan Antara MMB dan IDEA

Beberapa perbedaan mendasar antara MMB dan IDEA dapat dijabarkan sebagai berikut :

MMB	IDEA
Panjang *plaintext* dan *ciphertext* adalah 128 bit.	Panjang *plaintext* dan *ciphertext* adalah 64 bit.
Kunci yang digunakan pada proses enkripsi dan dekripsi berjumlah 4 buah sub kunci.	Kunci yang digunakan pada proses enkripsi dan dekripsi berjumlah 52 buah sub kunci.
Panjang setiap sub kunci adalah 32 bit.	Panjang setiap sub kunci adalah 16 bit.
Kunci yang digunakan pada proses enkripsi dan dekripsi sama.	Kunci yang digunakan pada proses enkripsi dan dekripsi tidak sama. Kunci dekripsi merupakan operasi kebalikan dari kunci enkripsi.
Proses enkripsi dan dekripsi menggunakan operasi perkalian modulo	Proses enkripsi dan dekripsi menggunakan operasi perkalian modulo

$2^{32} - 1$ sehingga tingkat sekuritas lebih tinggi.	$2^{16} + 1$.
Proses enkripsi dan dekripsi jauh lebih cepat daripada IDEA yaitu hanya terdiri dari 2 putaran saja.	Proses enkripsi dan dekripsi lebih panjang yaitu terdiri dari 8 putaran sehingga lebih memakan waktu.
Algoritma pada proses enkripsi berbeda dengan algoritma pada proses dekripsi. Algoritma proses dekripsi merupakan proses kebalikan dari proses enkripsi.	Algoritma pada proses enkripsi sama dengan algoritma pada proses dekripsi.
Proses enkripsi dan dekripsi menggunakan sebuah fungsi nonlinier f.	Tidak menggunakan fungsi nonlinier dalam proses enkripsi dan dekripsi.

Tabel 2.2 Perbandingan

Sedangkan, beberapa kesamaan antara MMB dan IDEA adalah :

1. Panjang kunci yang digunakan adalah 128 bit.
2. Proses pembentukan kunci terlalu sederhana.
3. Menggunakan operasi perkalian modulo.