BAB II

LANDASAN TEORI

2.1 Pengertian Kriptografi

Sejalan dengan perkembangan teknologi informasi, maka dikembangkan juga cara-cara untuk menangkal berbagai gangguan dalam pertukaran informasi. salah satu cara yang dianggap aman dalam pertukaran informasi adalah kriptografi yang melakukan transformasi data sehingga data tidak dapat dimengerti oleh orang yang tidak berhak.

Kriptografi berasal dari bahasa Yunani yaitu *kryptos* yang artinya "yang tersembunyi" dan *graphein* yang artinya "tulisan", jadi kriptografi adalah seni dan ilmu untuk menjaga keamanan data. Dan ahlinya disebut sebagai *cryptographer. Cryptanalst* merupakan orang yang melakukan *cryptanalysis,* yaitu seni dan ilmu untuk membuka *ciphertext* menjadi *plaintext* tanpa melalui cara yang seharusnya. Data yang dapat dibaca disebut *plaintext* dan teknik untuk membuat data tersebut menjadi tidak dapat dibaca disebut *enkripsi.* Data hasil dari enkripsi disebut *ciphertext,* dan proses untuk mengembalikan *ciphertext* menjadi *plaintext* disebut *dekripsi.* Cabang matematika yang mencakup kriptografi dan *cryptanalysis* disebut *cryptology* dan pelakunya disebut *cryptologist.*

2.2 Cryptosystem

Sistem kriptogrfi atau *cryptosystem* adalah sebuah algoritma ditambah semua kemungkinan *plaintext*, *ciphertext* dan kunci [SCH96]. Dalam sistem ini,

seperangkat parameter yang menentukan transformasi pen*cipher*an tertentu disebut suatu set kunci. Proses enkripsi dan dekripsi diatur oleh satu atau beberapa kunci kriptografi. Secara umum, kunci-kunci yang digunakan untuk proses enkripsi dan dekripsi tidak perlu identik, tergantung pada sistem yang digunakan.

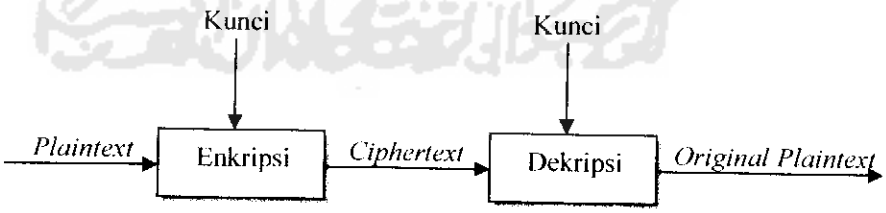
Setiap algoritma kriptografi terdiri algoritma enkripsi (E) dan algoritma dekripsi (D). Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu himpunan yang berisi elemen *plaintext* dan himpunan yang berisi elemen *ciphertext*. Enkripsi dan deskripsi merupakan fungsi tranformasi antara dua himpunan tersebut. Secara umum dapat digambarkan secara matematis sebagai berikut :

$E_k(P) = C \longrightarrow$ (Proses Enkripsi)

$D_k(C) = P \longrightarrow$ (Proses Dekripsi)

$D_k(E(P)) = P$ (Proses Dekripsi)

Dalam proses tersebut, *plaintext* disandikan dengan P dengan suatu kunci K lalu dihasilkan pesan C. Pada proses dekripsi, C diuraikan dengan menggunakan kunci K sehingga menghasilkan M yang sama dengan sebelumnya.



Gambar 2.1 Cryptosystem

Setiap *cryptosystem* yang baik memiliki karakteristik sebagai berikut :

1. Keamanan sistem terletak pada kerahasiaan kunci dan bukan pada kerahasian algoritma yang digunakan.
2. *Cryptosystem* yang baik memiliki ruang kunci (*keyspace)* yang besar.
3. *Cryptosystem* yang baik akan menghasilkan *ciphertext* yang terlihat acak dalam seluruh test statistik yang dilakukan.
4. *Cryptosystem* yang baik mampu menahan seluruh serangan yang telah dikenal sebelumnya.

Namun demikian, perlu diperhatikan bahwa bila suatu *cryptosystem* berhasil memenuhi seluruh karakteristik di atas, belum tentu ia merupakan sistem yang baik. Banyak *cryptosystem* lemah yang terlihat baik pada awalnya. Kadang kala untuk menunjukkan bahwa suatu *cryptosystem* kuat atau baik dapat dilakukan dengan menggunakan pembuktian matematika.[WAH03]

2.3 Sejarah Awal Kriptografi

Algoritma kriptografi pertama kali dikembangkan untuk mengizinkan organisasi tertentu yang ditunjuk untuk mengakses suatu informasi. Pertama kali algoritma kriptografi ini digunakan sebagai petunjuk dalam perang. Menurut sejarah pada tahun 1900 SM *Julius Caesar* dikenal sebagai orang yang pertama kali telah mengembangkan algoritma kriptografi untuk mengirimkan pesan ketentaranya. Dan kurang dari 500 tahun yang lalu. Enkripsi digunakan untuk dokumen religius dan formula, serta komunikasi pemerintah [WAH03].

Algoritma yang digunakan oleh Julius Caesar ini dikenal dengan nama *caesar cipher*. Cara kerja dari algoritma ini adalah menggeser setiap huruf dalam pesan yang akan menjadi algoritma standar sehingga dapat menginformasikan semua keputusan dan kemudian mengirim pesan dalam bentuk yang aman. Standar *caesar cipher* memiliki tabel karakter sandi yang dapat ditentukan sendiri. Ketentuan itu berdasarkan suatu kelipatan tertentu, misalnya tabel karakter sandi memiliki kelipatan tiga dari tabel karakter aslinya.

Huruf asli : 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

Huruf sandi: 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

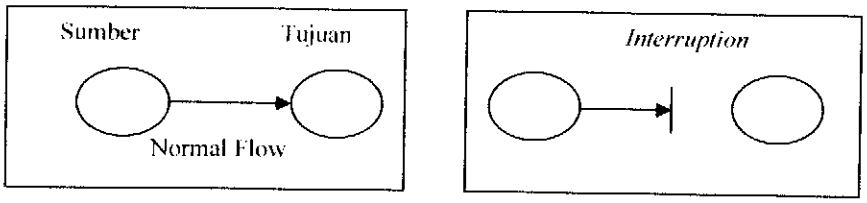
Jadi *caesar cipher* adalah *cipher* pergeseran, karena alfabet *ciphertext* diambil dari alfabet *plaintext*, dengan menggeser masing-masing huruf dengan jumlah tertentu.

2.4 Aspek Ancaman Terhadap Keamanan

Ada beberapa aspek yang berhubungan dengan ancaman terhadap keamanan, antara lain [WAH03] :

1. *Interruption*

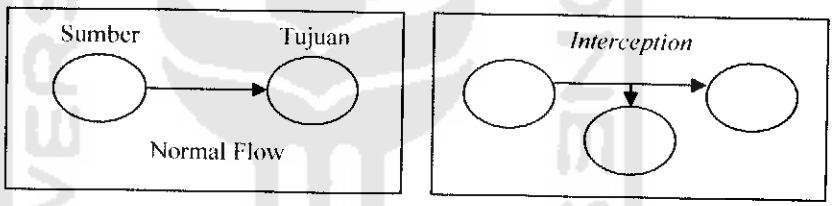
   Merupakan ancaman terhadap *availability* (ketersediaan data dan informasi), yaitu data dan informasi yang berada dalam sistem komputer dirusak atau dibuang, sehingga menjadi tidak ada dan tidak berguna. Contohnya memotong line komunikasi.



Gambar 2.2 Ancaman terhadap availability

2. *Interception*

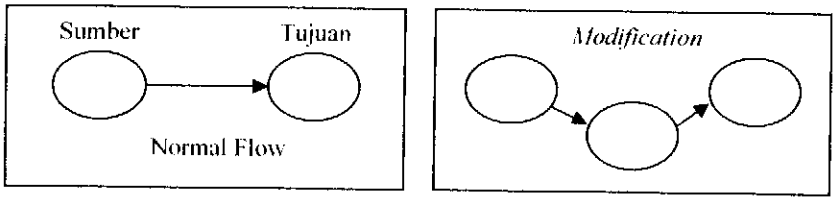
Merupakan ancaman terhadap *secrecy* (akses membaca data dan informasi), yaitu orang yang tidak berhak namun berhasil mendapatkan akses informasi dari dalam sistem komputer. Contohnya menyalin secara tidak sah *file* atau program.



Gambar 2.3 Ancaman terhadap *secrecy*

3. *Modification*

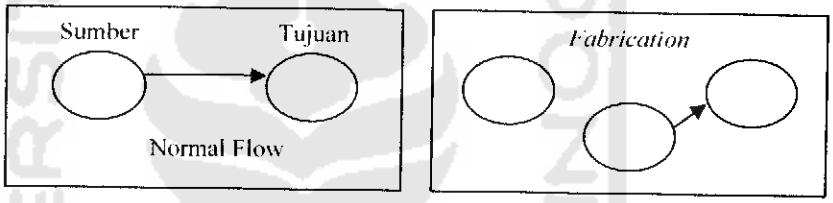
Merupakan ancaman terhadap *integritas* (akses merubah data dan informasi), yaitu orang yang tidak berhak yang tidak hanya berhasil mendapatkan akses informasi dari dalam sistem komputer, tetapi juga dapat melakukan perubahan terhadap informasi. Contohnya merubah program.



Gambar 2.4 Ancaman terhadap *integrity* (*Modification*)

4. *Fabrication*

Yaitu orang yang tidak berhak yang meniru atau memalsukan suatu obyek ke dalam sistem. Contohnya dengan menambahkan *record* ke dalam *file*.



Gambar 2.5 Ancaman terhadap *integrity* (*Fabrication*)

Secara garis besar ancaman terhadap keamanan suatu sistem komputer dapat dilihat pada tabel 2.1 berikut :

Tabel 2.1 Ancaman terhadap keamanan sistem komputer

Sistem Komputer	***Availability***	***Secrecy***	***Integrity***
Hardware	Peralatan dicuri atau dirusak	-	-
Software	Program dibuang atau dihapus	*Software* dikopi	Program diubah
Data	*File* dibuang atau dihapus	Analisis data untuk keperluan ilegal	*File* diubah atau *file* baru disisipkan
Line Komunikasi	Kabel diputus atau dirusak	Informasi disadap	Informasi diubah

2.5 Aspek-aspek Keamanan

Kriptografi tidak hanya memberikan kerahasiaan dalam telekomunikasi, namun juga memberikan komponen-komponen berikut : [KUR04]

1. *Authentication.* Penerima pesan dapat memastikan keaslian pengirimnya. Penyerang tidak dapat perpura-pura sebagai orang lain.
2. *Integrity.* Penerima harus dapat memeriksa apakah pesan telah dimodifikasi ditengah dijalan atau tidak. Seorang penyusup seharusnya tidak dapat memasukan tambahan ke dalam pesan, mengurangi atau mengubah pesan selama data berada di perjalanan.
3. *Nonrepudiation.* Pengirim seharusnya tidak dapat mengelak bahwa dialah pengirim pesan yang sesungguhnya. Tanpa kriptografi, seseorang dapat mengelak bahwa dia yang mengirim email yang sesungguhnya.
4. *Authority.* Informasi yang berada pada sistem jaringan seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang.

2.6 Macam-macam Serangan *Cryptanalyst*

Terdapat beberapa macam serangan yang munkkin dilakukan oleh *cryptanalyst*, dengan asumsi algoritma enkripsi telah dikenal luas : [KUR04]

1. *Ciphertext only attack. Cryptanalyst* hanya memiliki *ciphertext* yang akan dibaca dan mengetahui algoritma enkripsi yang digunakan.
2. *Known paintext attack. Cryptanalyst* dapat mengetahui beberapa *plaintext* beserta *ciphertext*nya. Dengan menganalisis *ciphertext* yang didapat dan membandingkan dengan *plaintext*nya, maka *cryptanalyst* dapat

menentukan *cipher* mana yang berkenaan dengan *plaintext* tersebut. Tugas *cryptanalyst* selanjutnya adalah menemukan kunci, sehingga dapat menemukan *plaintext* lainnya.

3. *Chosen plaintext attack. Cryptanalyst* tidak hanya mengetahui sejumlah *plaintext* dan *ciphertext*nya, namun bebas memilih *plaintext*nya supaya dienkrip dengan algoritma kunci yang sama. Analis sandi memilih sejumlah *plaintext* tertentu sehingga kuncinya dapat ditebak. Tugas analis sandi adalah menebak kuncinya.
4. *Adaptive chosen plaintext attack.* Serangan ini merupakan kasus khusus dari serangan jenis ketiga. Tidak hanya memilih *plaintext* yang dienkrip, *cryptanalyst* juga dapat memodifikasi pilihannya berdasarkan hasil enkrip sebelumnya. Dalam chosen *plaintext attack*, *cryptanalyst* mungkin hanya dapat memilih satu *block* besar *plaintext* untuk dienkrip, sedangkan pada serangan ini, dia dapat memilih *block plaintext* yang lebih kecil dan kemudian memilih lainnya berdasarkan hasil sebelumnya.
5. *Chosen ciphertext attack. Cryptanalyst* dapat memilih *ciphertext* yang berbeda untuk dienkrip dan mempunyai akses terhadap *plaintext* yang dienkrip. *Cryptanalyst* telah mengetahui pokok dari ini *plaintext*. *Cryptanalyst* hanya berusaha mencari kuncinya.
6. *Chosen text.* Merupakan gabungan *chosen plaintext* dan *chosen ciphertext*.

Sebuah algoritma yang kriptografi yang berkualitas apabila telah dikenal luas, dianalisis banyak orang dan telah teruji dapat bertahan dari usaha penjebolan.

2.7 Keamanan Algoritma

Suatu algoritma dikatakan aman, bila tidak ada cara ditemukan *plaintext*nya. Karena selalu terdapat kemungkinan ditemukannya cara baru untuk menembus algoritma kriptografi, maka algoritma kriptografi yang dikatakan "cukup" atau "mungkin" aman, bila memiliki keadaan sebagai berikut : [KUR04]

1. Bila harga untuk menjebol algoritma lebih besar daripada nilai informasi yang dibuka, maka algoritma itu cukup aman.
2. Bila waktu yang digunakan untuk membobol algoritma tersebut lebih lama daripada lamanya waktu yang diperlukan oleh informasi tersebut harus tetap aman, maka algoritma tersebut mungkin aman.
3. Bila jumlah data yang dienkrip dengan kunci dan algoritma yang sama lebih sedikit dari jumlah data yang diperlukan untuk menembus algoritma tersebut, maka algoritma itu aman.

2.8 Algoritma kriptografi

Kriptografi mempunyai sejarah yang amat panjang, mulai dari jaman romawi sampai saat ini. Sejak orang mengenal kriptografi, perkembangannya semakin pesat. Perkembangan algoritma kriptografi dapat kita bagi menjadi dua, yaitu : [KUR04]

1. Algoritma kriptografi klasik
2. Algoritma kriptografi modern

2.8.1 Algoritma Kriptografi Klasik

Pada algoritma klasik, diterapkan teknik enkripsi konvensional (simetris). Algoritma ini merupakan algoritma kriptografi yang biasa digunakan orang sejak berabad-abad yang lalu. Dua teknik dasar yang biasa digunakan, yaitu : [KUR04]

1. Teknik Subsitusi
2. Teknik Transposisi

2.8.1.1 Teknik Subsitusi

Subsitusi adalah penggantian setiap karakter *plaintext* dengan karakter lain. Beberapa istilah yang mungkin perlu diingat adalah : [KUR04]

a. *Monoalfabet* : Setiap karakter *ciphertext* mengganti satu macam karakter *plaintext* tertentu.

b. *Polyalfabet* : Setiap karakter *ciphertext* dapat mengganti lebih dari satu 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.

Cipher subsitusi paling tua yang dikenal adalah subsitusi yang dilakukan Julius Caesar. Beberapa teknik subsitusi yang pernah dilakukan, antara lain : [KUR04]

a. Subsitusi deret campuran kata kunci yaitu subsitusi yang kata kuncinya didapat dari mengumpulkan karakter yang sama dari sebuah *plaintext* dan pada *ciphertext*nya ditambahkan semua sisa karakter dalam abjad.

b. Subsitusi *monomer-dinome-trinome*. *Monome* berarti setiap satu karakter *plaintext* akan disubsitusi oleh satu karakter *ciphertext*, *dinome* disubsitusi dua karakter *ciphertext*, *tridome* disubsitusi tiga karakter *ciphertext*. Jadi sistem ini adalah campuran dari ketiga sistem dasar.

c. Subsitusi multilateral variant. Subsitusi ini masih termasuk jenis *monoalfabet* yang dalam mensubsitusi memanfaatkan huruf abjad a,b,c,…,z yang disusun dalam matrik 5 X 5.

d. Subsitusi digrafik. Pada sistem ini, setiap huruf *plaintext* akan disubsitusi oleh dua huruf *ciphertext*. Pola huruf *cipher text* diambil dari sebuah matrik 26 X 26 yang berasal dari 26 abjad yang memiliki pola khusus.

e. Subsitusi persegi panjang. Sistem digrafik terlalu memerlukan matrik yang besar. Untuk memperkecil matrik dengan keamanan yang setara dapat digunakan sistem empat persegi.

f. Subsitusi kode *playfair*. Kode rahasia multi huruf yang paling terkenal adalah *playfair*. *Playfair* menggunakan 676 digraf. Selama waktu yang lama, kode ini dianggap tak dapat dipecahkan. *Playfair* dijadikan sistem standar oleh tentara Inggris dalam PD I dan masih digunakan secara luas oleh tentara Amerika dan sekutu selama PD II. Sistem ini menggunakan matrik 5 X 5.

g. Subsitusi *Polialfabet* periodik. Dalam sistem *polialfabet*, setiap *ciphertext* dapat memiliki banyak kemungkinan *plaintext*. Dan sistem periodik itu sendiri dikarenakan adanya kunci yang berulang. Jenis *polialfabet* klasik yang terkenal adalah *Vigenere*.

h. Enigma. Merupakan mesin kriptografi yang digunakan oleh tentara NAZI Hitler pada masa PD II. Mesin ini menggunakan rotor. Enigma menggunakan tiga rotor untuk melakukan subsitusi. Tiga rotor berarti tiga kali subsitusi.

2.8.1.2 Teknik Transposisi (Permutasi)

Beberapa model kriptografi yang menggunakan teknik transposisi, antara lain : [KUR04]

1. Algoritma transposisi kolom dengan kunci numerik. Teknik ini menggunakan permutasi karakter. Kunci dapat diperoleh dari kata yang mudah dibaca dan kemudian dikodekan menjadi bilangan.
2. Masukan *plaintext* pola *zig-zag*, keluaran *ciphertext* berupa baris.
3. Masukan pola segitiga, keluaran berupa kolom, dibaca dari atas kebawah.
4. Masukan berpola spiral, dari luar kedalam, keluaran berupa kolom dibaca dari atas ke bawah.
5. Dimasukan secara diagonal dari kiri bawah ke kanan atas, keluaran baris.
6. Masukan spiral dari dalam ke luar, keluaran diagonal bergantian.

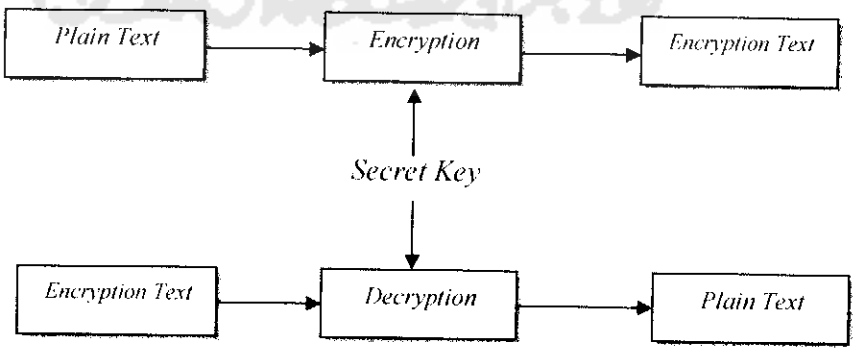
Kombinasi subsitusi dan transposisi yang komplek menjadi dasar pembentukan algoritma-algoritma kriptografi modern. Salah satu algoritma klasik yang menggunakan kedua teknik ini adalah VIC yang tidak memerlukan komputer dalam penggunaannya.

2.8.2 Algoritma Kriptografi Modern

Perkembangan dan kemajuan teknologi menuntut adanya algoritma-algoritma kriptografi yang lebih baik dan aman. Untuk itu telah muncul berbagai macam algoritma baru yang terus dianalis, dicoba dan disempurnakan untuk mencari sebuah algoritma yang dianggap dapat memenuhi standar keamanan yang diharapkan. Pada algoritma modern saat ini lebih berfukos pada kunci yang digunakan, jadi tidak hanya tergantung pada kesulitan algoritmanya. Macam-macam algoritma menurut kuncinya adalah algoritma simetris dan algoritma asimetris.

2.8.2.1 Algoritma Simetris

Algoritma simetris disebut juga sebagai algoritma konvensional, yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan deskripsinya. Keamanan algoritma simetris tergantung pada kuncinya. Algoritma simetris sering juga disebut algoritma kunci rahasia, algoritma kunci tunggal atau algoritma satu kunci. Dua kategori yang termasuk pada algoritma simetris ini adalah algoritma *block cipher* dan *stream cipher*.



Gambar 2.6 Algoritma kriptografi simetris

2.8.2.1.1 *Block Cipher*

Algoritma *block cipher* adalah algoritma yamg masukan dan keluarannya berupa satu *block*, dan setiap *block*nya terdiri dari banyak bit. Beberapa mode operasi enkripsi *block cipher* :

1. Data Enkripsi Standard (DES)

Algoritma enkripsi yang paling banyak digunakan di dunia adalah DES yang telah diadopsi oleh NIST (*Nasional Institude of Standard and Tecnology*) sebagai standard pengolahan informasi Federal AS. Data dienkrip dalam *block-block* 64 bit menggunakan kunci 56 bit.[KUR04]

Algoritma DES berasal dari algoritma Lucifer buatan IBM. Algoritma ini ditawarkan kepada NIST dan menjadi DES tahun 1977. Akan tetapi terdapat dua masalah besar pada algoritma ini. Pertama, kunci yang hanya 56 bit, sehingga sangat rawan terhadap serangan *brute force*. Kedua, desain struktur internal DES dimana bagian subsitusinya (S-box) masih dirahasiakan.

2. AES (*Advanced Encrytion Standard*)

Sekitar tahun1990-an ketika semakin banyak komputer yang dapat menembus kunci DES yang disebabkan terlalu pendeknya panjang kunci.

Dalam kriptografi modern, panjang kunci dalam ukuran jumlah bit yang digunakan, merupakan salah satu faktor yang sangat penting. Hal ini dikarenakan penggunaan komputer yang sangat intensif dalam dunia kriptografi. Untuk itu NIST mengadakan sebuah kontes untuk mencari sebuah algoritma standard baru untuk menggantikan DES. Pada bulan oktober 2000, *Rijndael* dipilih menjadi pemenang kontes AES. Algoritma *Rijndael* dipilih

bukan karena yang paling aman, melainkan karena keseimbangan antara keamanan dan fleksibilitas dalam berbagai platform *software* dan *hardware*. Algoritma menyingkirkan saingan terdekatnya yaitu Algoritma RC6 buatan RSA.[KUR04]

3. ***Blowfish* (*Review* tulisan Anton)**

Blowfish merupakan *block cipher* 64-bit dengan panjang kunci variabel. Algoritma ini terdiri dari dua bagian: *key expansion* dan enkripsi data. *Key expansion* merubah kunci yang dapat mencapai 448 bit menjadi beberapa array subkunci (*subkey*) dengan total 4168 *byte*.[SCH01]

Enkripsi data terdiri dari iterasi fungsi sederhana sebanyak 16 kali. Setiap putaran terdiri dari permutasi kunci-*dependent* dan substitusi kunci dan data *dependent*. Semua operasi adalah penambahan dan XOR pada variable 32-bit. Tambahan operasi lainnya hanyalah empat penelusuran tabel (*table lookup*) array berindeks untuk setiap putaran. *Blowfish* menggunakan subkunci yang besar. Kunci ini harus dihitung sebelum enkripsi atau dekripsi data. [SCH01]

2.8.2.1.2 *Stream Cipher*

Stream cipher (*Cipher* aliran) adalah *cipher* yang berasal dari hasil XOR antara bit *plaintext* dengan setiap bit kuncinya. *Stream cipher* sangat rawan terhadap *attack* pembalikan bit. Beberapa model algoritma *stream cipher* antara lain :

1. ***One Time Pad*** **(OTP)**

Dalam OTP terdapat teknik enkripsi yang sempurna. Ditemukan oleh Mayor J Maugborne dan G Verman tahun 1917. Setiap kunci hanya digunakan untuk sekali pesan. Teknik ini dikatakan sempurna di karena kunci yang acak dan hanya digunakan sekali. Untuk membangkitkan kunci OTP diperlukan pembangkit bilangan acak yang tidaklah mudah.

2. ***Rivest Code 4*** **(RC 4) (** ***Review*** **tulisan Dian Wahyudi)**

RC4 merupakan salah satu algoritma kunci simetris yang berbentuk *stream cipher*, yaitu memproses unit atau *input* data pada satu saat. Unit atau data pada umumnya sebuah *byte* atau kadang-kadang bit. Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah *input* data tertentu sebelum diproses, atau menambahkan *byte* tambahan untuk mengenkripsi. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan RSA.[WAH04]

RC4 merupakan enkripsi *stream simetrik proprietary* yang dibuat oleh *RSA Data Security Inc* (RSADSI). Penyebarannya diawali dari sebuah source *code* yang diyakini sebagai RC4 dan dipublikasikan secara '*anonymously*' pada tahun 1994. Algoritma yang dipublikasikan ini sangat identik dengan implementasi RC4 pada produk resmi. RC4 digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. Sampai saat ini diketahui tidak ada yang dapat memecahkan/membongkarnya. RC4 tidak

dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas (*trade secret*).[WAH04]

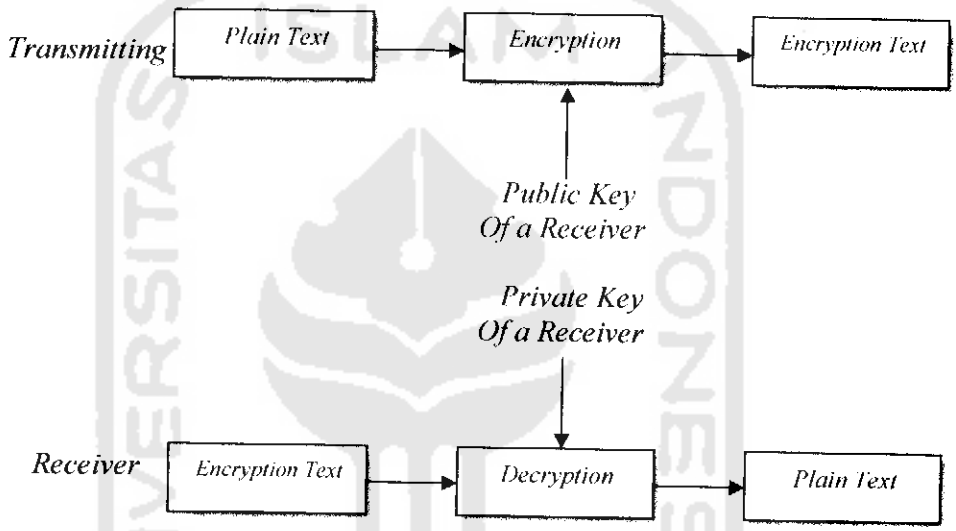
Algoritma RC4 bekerja pada dua tahap, menyetem susunan (*key setup*) dan pengkodean (*ciphering*). Kunci susunan merupakan hal yang lebih awal dan merupakan tahap yang paling sulit dari algoritma ini. Selama menyetem susunan suatu N-bit (N menjadi panjangnya kunci), kunci enkripsi digunakan untuk menghasilkan suatu variabel enkripsi yang menggunakan dua arrays, state dan kunci, dan jumlah N dari operasi pencampuran. Operasi pencampuran terdiri dari menukar *bytes*, modulo operasi, dan rumusan lain. Suatu modulo operasi adalah proses sisa dari suatu hasil divisi. Sebagai contoh, 11/4 adalah 2 sisa 3; oleh karena itu 11 mod 4 sama dengan 3. [WAH04]

Sekali variabel enkripsi dihasilkan dari *key setup*, langkah selanjutnya adalah masuk ke fase *ciphering* di mana dalam proses ini hasilnya akan diXORkan dengan *plaintext*. Sekali penerima mendapat pesan yang dienkripsi, langkah selanjutnya adalah mendekripsinya dengan XOR pesan yang dienkripsi dengan menggunakan variabel yang sama.[WAH03]

2.8.2.2 Algoritma Asimetris

Algoritma asimetrik atau biasa disebut algoritma kunci publik dirancang sedemikian mungkin sehingga kunci yang digunakan untuk mengenkripsi dan mendeskripsi berbeda. Selanjutnya kunci dekripsi tidak dapat dihitung dengan dari kunci enkripsi. Algoritma tersebut disebut public- kunci karena kunci enkripsi

dapat dibuat secara *public*. Orang asing dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi sebuah pesan, tetapi hanya orang tertentu dengan kunci dekripsi sepadan dapat mendekripsi pesan tersebut. Dalam sistem ini kunci enkripsi sering disebut *public* kunci sedangkan key dekripsi sering disebut *private key*.



Gambar 2.7. Algoritma kriptografi asimetris

Beberapa algoritma asimetrik antara lain :

1. **RSA**

   RSA tidak pernah dibuktikan aman tidaknya, hanya karena sulitnya pemfaktoran bilangan yang sangat besar, maka RSA dianggap aman. Dalam pembangkitan kedua kunci, digunakan dua bilangan prima acak yang sangat besar.

2. ***Diffie-Hellman*** **(DH)**

Diffie Helman dianggap merupakan algoritma asimetrik yang pertama kali ditemukan pada tahun 1976, meskipun NSA telah mengaku menemukan algoritma asimetrik jauh-jauh hari sebelumnya. Algoritma ini memperoleh keamanannya dari sulitnya menghitung logaritma diskrit pada bilangan yang amat besar. Akan tetapi algoritma ini hanya dapat digunakan untuk pertukaran kunci (simetris) dan tidak dapat digunakan untuk enkripsi/dekripsi maupun untuk tandat tangan digital.

2.8.2.3 Fungsi *Hash*

Fungsi satu arah (*one way function*) sering disebut fungsi hash, *message digest*, *fingerprint*, fungsi kompresi, dan *message authentication code* (MAC). Fungsi ini biasanya diperlukan jika kita menginginkan mengambil *sidik jari* suatu pesan. Sebagaimana sidik jari manusia yang menunjukkan identitas si pemilik sidik jari, fungsi ini juga diharapkan mempunyai kemampuan serupa dengan sidik jari manusia, dimana sidik jari pesan diharap menunjuk ke satu pesan dan tidak dapat menunjuk ke pesan lainnya. Dinamakan sebagai fungsi kompresi karena biasanya, masukan fungsi satu arah ini lebih besar dari pada keluarannya. Namun kompresi hasil pesan ini tidak dapat dikembalikan ke asalnya sehingga disebut fungsi satu arah. Dinamakan sebagai *message digest* karena seolah-olah merupakan inti pesan. Padahal tidak demikian, sebab inti sari pesan mestinya merupakan ringkasan pesan yang masih dapat dipahami maknanya, sedangkan

disini justru sebaliknya, malahan dengan diketahuinya sidik jari ini, justru orang diharapkan tidak tahu pesan aslinya. [KUR04]

Salah satu funsi *hash* yang paling banyak digunakan dalam keamanan jaringan komputer dan internet adalah MD (*Message Digest*) versi 5. MD5 merupakan kelanjutan MD4 yang dirancang dengan tujuan :

1. **keamanan**. Secara perhitungan matematis tidak dimungkinkan untuk mendapat dua pesan yang memiliki *hash* yang sama. Tidak ada serangan yang lebih efisien dibanding *brute force*.
2. **Keamanan Langsung.** Kemanan MD5 tidak didasarkan pada asumsi, seperti kesulitan pemfaktoran.
3. **Kecepatan.** MD5 sesuai untuk diimplementasikan dengan perangkat lunak yang berkecepatan tinggi, karena berdasar pada sekumpulan manipulasi operan 32-bit.
4. **Kesederhanaan dan Kompak.** MD5 sederhana tanpa struktur data atau program yang komplek. [KUR04]

2.9 Algoritma Kriptografi RC6

Algoritma RC6 merupakan salah satu kandidat *Advanced Encryption Standard* (AES) yang diajukan oleh *RSA Laboratoriest* kepada NIST. Dirancang oleh Ronald L Rivest, M.J.B. Robshaw, R. Sidney dan Y.L. Yin, algoritma ini merupakan pengembangan dari algoritma sebelumnya yaitu RC5 dan telah memenuhi semua kriteria yang diajukan oleh NIST.

2.9.1 Gambaran RC6

Algoritma RC6 adalah versi yang dilengkapi dengan beberapa parameter, sehingga dituliskan sebagai RC6-w/r/b, dimana parameter w merupakan ukuran kata dalam satuan bit, r adalah bilangan bulat bukan negatif yang menunjukan banyaknya iterasi selam proses enkripsi, dan b menunjukan ukuran kunci enkripsi dalam *byte*. Ketika algoritma ini masuk sebagai kandidat AES, maka ditetapkan nilai parameter w = 32, r = 20 dan b bervariasi antara 16, 24, dan 32 *byte*.[ABD02]

RC6-w/r/b memecah *block* 128 bit menjadi 4 buah *block* 32 bit, dan mengikuti enam aturan operasi dasar sebagai berikut :

A + B	Operasi penjumlahan bilangan integer.
A – B	Operasi pengurangan bilangan integer.
A B	Operasi *exclusive*-OR (XOR)
A x B	Operasi perkalian bilangan integer.
A<<<B	A dirotasikan ke kiri sebanyak variabel kedua (B)
A>>>B	A dirotasikan ke kanani sebanyak variabel kedua (B)

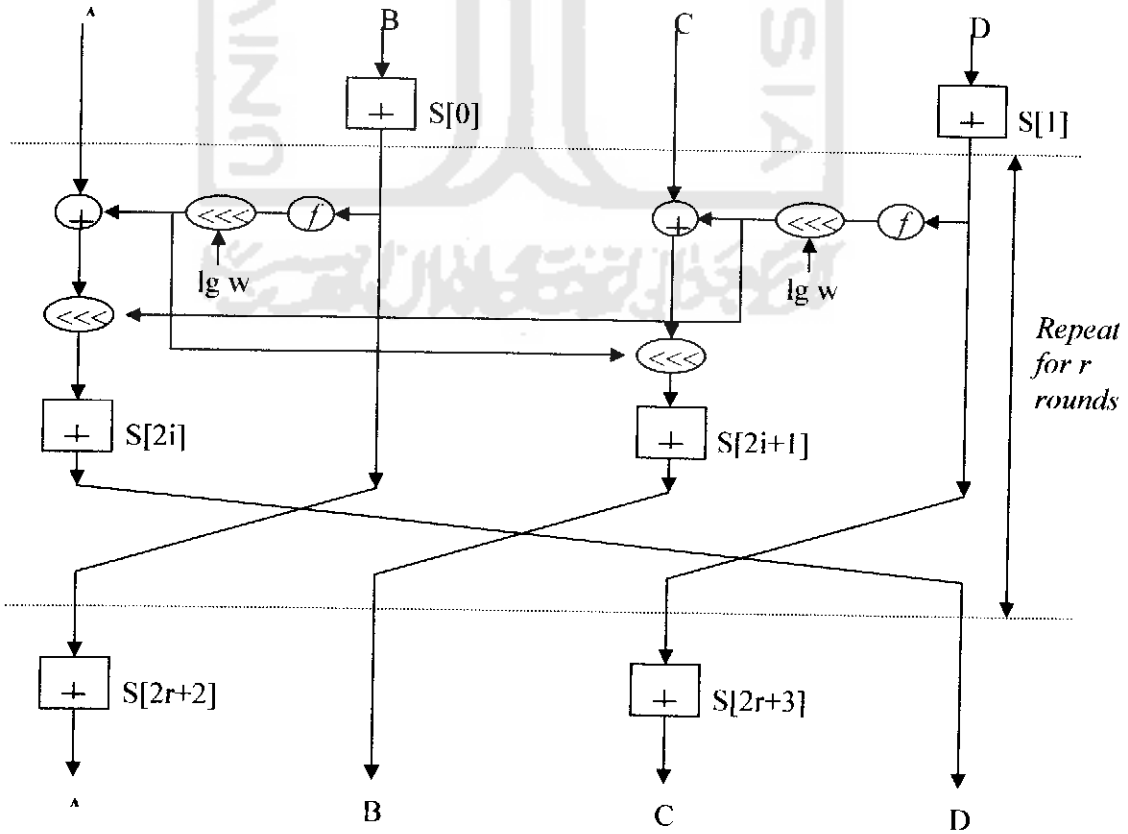
2.9.2 Algoritma Enkripsi RC6

Karena RC6 memecah *block* 128 bit menjadi 4 buah *block* 32 bit, maka algoritma ini bekerja dengan 4 buah register 32-bit A, B, C, D. *Byte* yang pertama dari *plaintext* atau *ciphertext* ditempatkan pada *byte* A, sedangkan *byte* yang terakhirnya ditempatkan pada *byte* D. Dalam prosesnya akan didapatkan (A, B, C,

D) = (B, C, D, A) yang diartikan bahwa nilai yang terletak pada sisi kanan berasal dari register disisi kiri.[ABD02] Berikut ini adalah algoritma enkripsi RC6 :

```
B = B + S[ 0 ]
D = D + S[ 1 ]
for i = 1 to 20 do
  {
    t = ( B x ( 2B + 1 ) ) <<< 5
    u = ( D x ( 2D + 1 ) ) <<< 5
    A = ( ( A ⊕ t ) <<< u ) + S[ 2i ]
    C = ( ( C ⊕ u ) <<< t ) + S[ 2i + 1 ]
    (A, B, C, D) = (B, C, D, A)
  }
A = A + S[ 42 ]
C = C + S[ 43 ]
```

Diagram *block* berikut akan lebih menjelaskan proses enkripsi yang terjadi pada algoritma RC6 :



Gambar 2.8. *Round*/iterasi algoritma RC6

Algoritma RC6 menggunakan 44 buah sub kunci yang dibangkitkan dari kunci dan dinamakan dengan S[0] hingga S[43]. Masing-masing sub kunci panjangnya 32 bit. Proses enkripsi pada algoritma RC6 dimulai dan diakhiri dengan proses *whitening* yang bertujuan untuk menyamarkan iterasi yang pertama dan yang terakhir dari proses enkripsi dan dekripsi. Pada proses *whitening* awal, nilai B akan dijumlahkan dengan S[0], dan nilai D dijumlahkan dengan S[i]. Pada masing-masing iterasi pada RC6 menggunakan 2 buah sub kunci. Sub kunci pada iterasi yang pertama menggunakan S[2] dan S[3], sedangkan iterasi-iterasi berikutnya menggunakan sub-sub kunci lanjutannya. Setelah iterasi ke-20 selesai, dilakukan proses *whitening* akhir dimana nilai A dijumlahkan dengan S[42], dan nilai C dijumlahkan dengan S[43].[ABD02]

Setiap iterasi pada algoritma RC6 mengikuti aturan sebagai berikut, nilai B dimasukan ke dalam fungsi f, yang didefinisikan sebagai f(x) =x(2x+1) , kemudian diputar kekiri sejauh lg-w atau 5 bit. Hasil yang didapat pada proses ini dimisalkan sebagai u. Nilai u kemudian di XOR dengan C dan hasilnya menjadi nilai C.

Nilai t juga digunakan sebagai acuan bagi C untuk memutar nilainya kekiri. Begitu pula dengan nilai u, juga digunakan sebagai acuan bagi nilai A untuk melakukan proses pemutaran kekiri.

Kemudian sub kunci S[2i] pada iterasi dijumlahkan dengan A, dan sub kunci S[2i+1] dijumlahkan dengan C. keempat bagian dari *block* kemudian akan

dipertukarkan dengan mengikuti aturan, bahwa nilai A ditempakan pada D, nilai B ditempatkan pada A, nilai C ditempatkan pada B, dan nilai (asli) D ditempatkan pad C. demikian iterasi tersebut akan terus berlangsung hingga 20 kali.[ABD02]

2.9.3 Algoritma Dekripsi RC6

Proses dekripsi *ciphertext* pada algoritma RC6 merupakan pembalikan dari proses enkripsi. Pada proses *whitening*, bila proses enkripsi menggunakan operasi penjumlahan, maka pada proses dekripsi menggunakan operasi pengurangan. Sub kunci yang digunakan pada proses *whitening* setelah iterasi terakhir diterapkan sebelum iterasi pertama, begitu juga sebaliknya sub kunci yang diterapkan pada proses *whitening* sebelum iterasi pertama digunakan pada *whitening* setelah iterasi terkahir. Akibatnya, untuk melakukan dekripsi, hal yang harus dilakukan semata-mata hanyalah menerapkan algoritma yang sama dengan enkripsi, dengan tiap iterasi menggunakan sub kunci yang sama dengan yang digunakan pada saat enkripsi, hanya saja urutan sub kunci yang digunakan terbalik.[ABD02] Berikut ini adalah algoritma deskripsi RC6 :

```
C = C - S[ 43 ]
A = A - S[ 42 ]
for i = 20 downto 1 do
  {
    (A, B, C, D) = (D, A, B, C)
    u = ( D x ( 2D + 1 ) ) <<< 5
    t = ( B x ( 2B + 1 ) ) <<< 5
    C = ( ( C - S[ 2i + 1 ] ) >>> t ) ⊕ u
    A = ( ( A - S[ 2i ] ) >>> u ) ⊕ t
  }
```

```
D = D - S[ 1 ]
B = B - S[ 0 ]
```

2.9.4 Pembangkitan Kunci

Pengguna memasukkan sebuah kunci yang besarnya b *byte*, dimana 0≤b≤255. *byte* kunci ini kemudian ditempatkan dalam array c w-bit words L[0]...L[c-1]. *Byte* pertama kunci akan ditempatkan sebagai pada L[0], *byte* kedua pada L[1], dan seterusnya. (Catatan, bila b=0 maka c=1 dan L[0]=0). Masing-masing nilai kata w-bit akan dibangkitkan pada penambahan kunci *round* 2r+4 dan akan ditempatkan pada array S[0,...,2r+3].[ABD02]

Konstanta P32 = B7E15163 dan Q32 = 9E3779B9 (dalam satuan heksadesimal) adalah "konstanta ajaib"yang digunakan dalam penjadwalan kunci pada RC6. Nilai P32 diperolch dari perluasan bilangan biner e-2, dimana e adalah sebuah fungsi logaritma. Sedangkan nilai Q32 diperolch dari perluasan bilangan biner ∅-1, dimana ∅ dapat dikatakan sebagai "*golden ratio*" (rasio emas). Algoritma untuk pembangkitan kunci RC6 adalah sebagai berikut :

```
S[ 0 ] = 0xB7E15163
for i = 1 to 43 do S[i] = S[i-1] + 0x9E3779B9
A = B = i = j = 0
for k = 1 to 132 do
  { A = S[ i ] = ( S[ i ] + A + B ) <<< 3
    B = L[ j ] = ( L[ j ] + A + B ) <<< ( A + B )
    i = ( i + 1 ) mod 44
    j = ( j + 1 ) mod c        }
```

2.9.5 Kajian RC6 Secara Teoritis

1. Struktur *cipher*

   RC6 menggunakan mekanisme pembagian *block* dari 128 bit menjadi empat buah *block* 32 bit. Adanya pergeseran lima bit ke kiri dengan menggunakan fungsi $f(x) = x(2x+1)$ dapat memberikan tingkat keamanan yang tinggi.

2. Banyak iterasi

   Dalam penentuan banyaknya iterasi didasarkan pada azas keseimbangan. Diman jika jumlah round sedikit akan menyebabkan *cipher* menjadi mudah untuk dipecahkan. Sedangkan jjika terlalu banyak akan menyebabkan kecepatan proses enkripsi/deskripsi akan semakin berkurang.

3. Perubahan besar arsip

   Hasil dari enkripsi atau *ciphertext* mempunyai ukuran yang lebih besar. Hal ini terjadi karena adanya proses *padding*.

4. *Avallanche effect*

   Avallanche effect merupakan salah satu cara untuk menentukan apakah algoritma kriptografi baik atau tidak. Perubahan yang kecil pada *plaintext* maupun key akan menyebabkan perubahan yang signifikan terhadap *ciphertext* yang dihasilkan. Atau dengan kata lain, perubahan satu bit pada palin*text* maupun key akan menghasilkan perubahan banyak bit pada *ciphertext*. Suatu *avalanche effect* dikatakan baik jika perubahan bit yang dihasilkan berkisar antara 45-60% (sekitar separuhnya, 50 % adalah hasil yang sangat baik). [SCH96] Hal ini dikarenakan perubahan tersebut berarti

membuat perbedaan yang cukup sulit untuk kriptanalis melakukan serangan.

5. Kunci lemah dan setengah lemah

   Pada algoritma RC6 sampai saat ini masih belum ditemukan adanya kunci lemah. Semua kunci mempunyai kekuatan yang sama dan dapat digunakan untuk enkripsi ganda (kunci lemah mempunyai sifat yaitu jika sebuah *plaintext* dienkripsi ganda menggunkan kunci lemah akan menghasilkan *plaintext* itu sendiri). Kunci setengah lemah adalah sepasang kunci yang mempunyai sifat yaitu jika sebuah *plaintext* dienkripsi dengan suatu kunci dan didekripsi kembali dengan kunci pasangannya akan menghasilkan *plaintext* itu kembali.