BAB II

LANDASAN TEORI

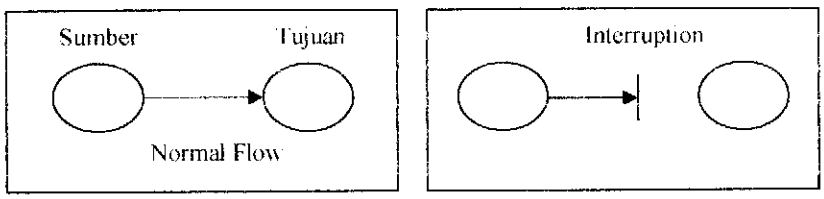
2.1 *Cryptographic System (Cryptosystem)*

Dalam teknologi informasi, telah dan sedang dikembangkan cara-cara untuk menangkal berbagai bentuk serangan. Salah satu cara yang ditempuh mengatasi masalah ini ialah dengan menggunakan kriptografi yang menggunakan transformasi data sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak ketiga. Transformasi ini memberikan solusi pada dua masalah keamanan data, yaitu masalah privasi (*privacy*) dan keautentikan (*authentication*). Privasi mengandung arti bahwa data yang dikirim hanya dapat dimengerti informasinya oleh penerima yang sah. Sedangkan keautentikan mencegah pihak ketiga untuk mengirim data yang salah atau mengubah data yang dikirimkan.

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

1. *Interruption*

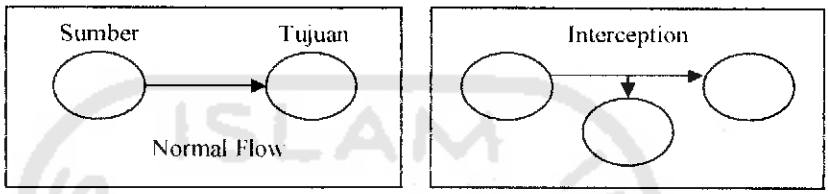
   Merupakan ancaman terhadap *availability* (ketersediaan data dan informasi), yaitu data dan informasi yang berada dalam sistem komputer dirusak atau dibuang, sehingga menjadi tidak ada dan tidak berguna. Contohnya memotong line komunikasi.



Gambar 2.1 Ancaman Terhadap Availability

2. *Interception*

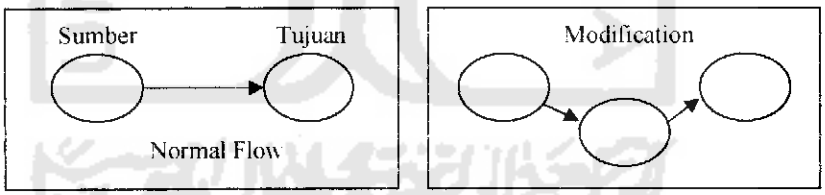
Merupakan ancaman terhadap *secrecy* (akses membaca data dan informasi), yaitu orang yang tidak berhak namun berhasil mendapatkan akses informasi dari dalam sistem komputer. Contohnya menyalin secara tidak sah file atau program.



Gambar 2.2 Ancaman Terhadap Secrecy

3. *Modification*

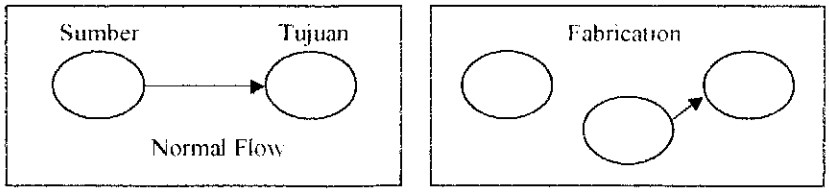
Merupakan ancaman terhadap *integritas* (akses merubah data dan informasi), yaitu orang yang tidak berhak yang tidak hanya berhasil mendapatkan akses informasi dari dalam sistem komputer, tetapi juga dapat melakukan perubahan terhadap informasi. Contohnya merubah program.



Gambar 2.3 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 2.4 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

Adapun tujuan dari sistem kriptografi adalah sebagai berikut :

1. *Confidentiality*

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

2. *Message Integrity*

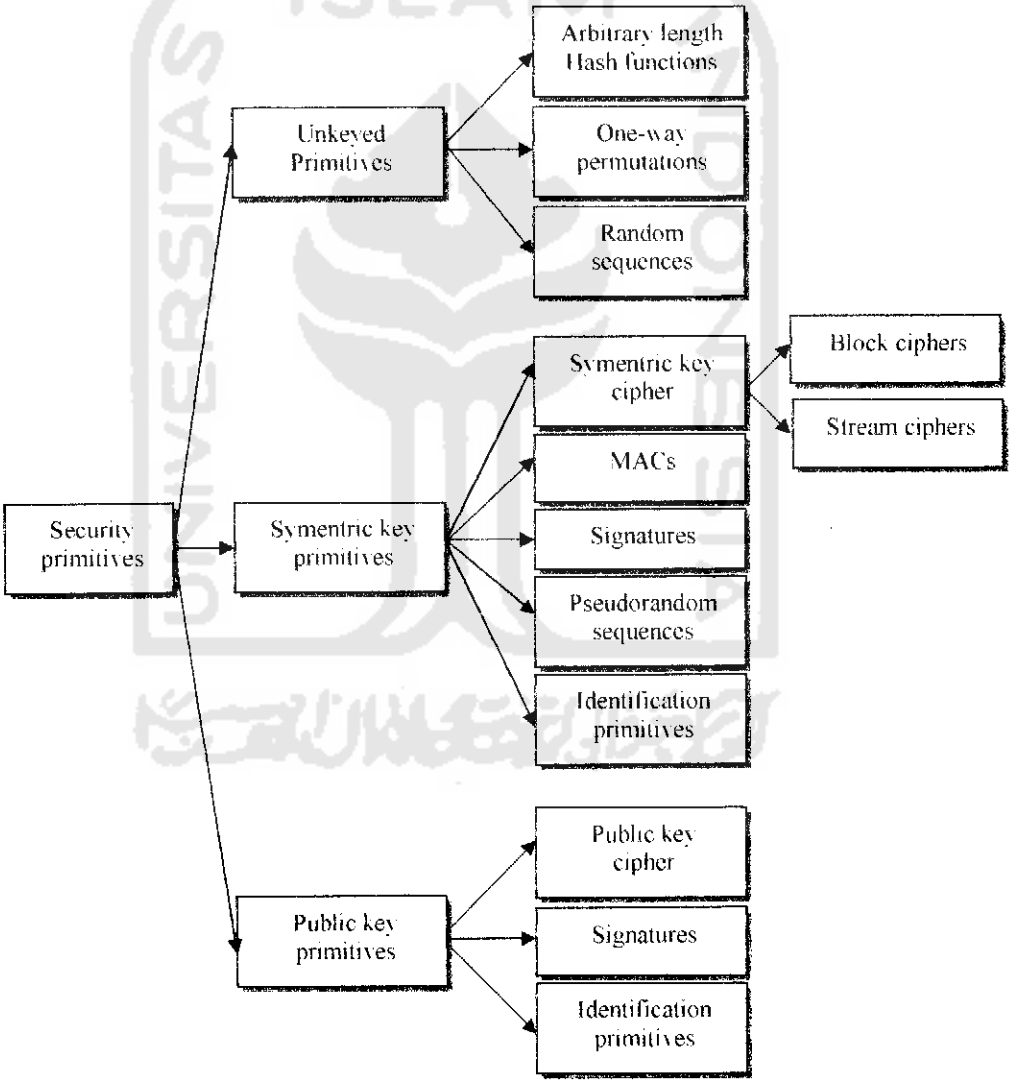
   Memberikan jaminan untuk tiap bagian bahwa pesan tidak akan mengalami perubahan dari saat ia dibuat sampai saat ia dibuka.

3. *Non-repudiation*

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

4. *Authentication*

   Memberikan dua layanan. *Pertama,* mengidentifikasi keaslian suatu pesan dan memberikan jaminan keautentikannya. *Kedua,* menguji identitas seseorang apabila ia akan memasuki sebuah sistem.



Gambar 2.5 Hirarki Sistem Kriptografi Primitive[HAC96]

2.2 Algoritma Kriptografi

Kriptografi berasal dari bahasa Yunani yaitu *kryptos* yang artinya " yang tersembunyi " dan *graphein* yang artinya " tulisan", jadi kriptografi merupakan teknik menjaga kerahasiaan pesan (*message*), agar pesan tersebut tidak dapat dibaca oleh orang yang tidak berkepentingan. Pengirim pesan harus yakin bahwa pesan yang akan dikirim tidak dapat dibaca oleh orang lain selain penerima pesan. Orang yang mengamankan pesan disebut *cryptographer*. Sedangkan *cryptanalysis* adalah suatu ilmu dan seni membuka (*breaking*) *ciphertext* dan orang yang melakukannya disebut *cryptanalyst*.

Cryptographic system atau *cryptosystem* adalah suatu fasilitas untuk mengkonversikan *plaintext* ke *ciphertext* dan sebaliknya. Proses yang melakukan perubahan dari pesan sesungguhnya (*plaintext*) menjadi sebuah kode rahasia (*ciphertext*) disebut dengan *enkripsi*. Enkripsi dapat diartikan sebagai kode atau *cipher*. Sebuah sistem pengkodean menggunakan suatu *table* atau kamus yang telah didefenisikan untuk menggantikan kata dari informasi atau yang merupakan bagian dari informasi yang dikirim. Sebuah cipher menggunakan suatu algoritma yang dapat mengkodekan semua aliran data (*stream*) bit dari sebuah pesan menjadi *cryptogram* yang tidak dimengerti (*unintelligible*). Sedangkan proses dengan algoritma yang sama untuk mengembalikan informasi teracak (*ciphertext*) menjadi bentuk aslinya (*plaintext*) disebut *dekripsi*.

Dalam sistem ini, seperangkat parameter yang menentukan transformasi pencipheran 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.

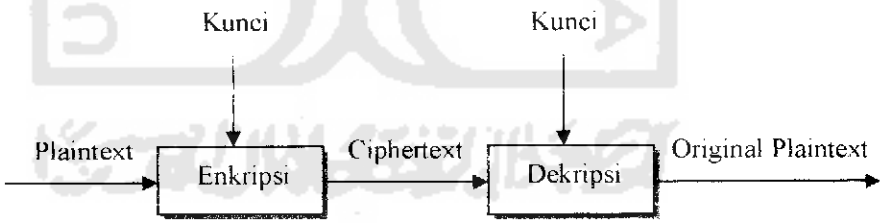
Algoritma kriptografi terdiri dari algoritma enkripsi (E) dan algortima dekripsi (D). Algoritma enkripsi menggunakan kunci enkripsi (E_K), sedangkan algoritma dekripsi menggunakan kunci dekripsi (D_K). Secara umum operasi enkripsi dan dekripsi dapat diterangkan secara matematis sebagai berikut:

$E_K (M) = C \longrightarrow$ (Proses Enkripsi)

$D_K (C) = M \longrightarrow$ (Proses Dekripsi)

$D_K (E (M)) = M$ (Proses Dekripsi)

Pada saat proses enkripsi kita menyandikan pesan M dengan suatu kunci K lalu dihasilkan pesan C. Sedangkan pada proses dekripsi, pesan C tersebut diuraikan dengan menggunakan kunci K sehingga dihasilkan pesan M yang sama seperti pesan sebelumnya.



Gambar 2.6 Cryptosystem Secara Umum

Salah satu contoh yang sederhana dari algoritma kriptografi adalah *caesar cipher*, yang digunakan oleh Julius Caesar untuk berkomunikasi dengan tentaranya. Caesar ini dipertimbangkan menjadi salah seorang yang pertama kali menggunakan enkripsi untuk mengamankan pesanya. Cara kerja dari algoritma ini adalah menggeser setiap huruf dalam pesan yang akan mejadi algoritma standar sehingga dapat menginformasikan semua keputusan dan kemudian megirim 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

Dalam contoh ini huruf A diganti dengan huruf D, huruf B diganti dengan huruf E, dan seterusnya sampai huruf Z diganti dengan huruf C. Dari sini bisa dilihat bahwa pergeseran huruf menggunakan 3 huruf ke kanan. Ketentuan tabel karakter sandi dapat diubah sesuai dengan jumlah kelipatan dari huruf aslinya.

Misal pesan plaintextnya adalah : " t o p s e c r e t "
Maka ciphertextnya adalah : " w r s v h f u h w "

Melalui algoritma ini, jika musuh melakukan sabotase terhadap pesan, itu akan menjadi sia-sia karena hanya kelompok caesar yang dapat membacanya.

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

Setiap *cryptosystem* yang baik harus memiliki karakteristik sebagai berikut :

1 Keamanan sistem terletak pada kerahasiaan kunci dan bukan pada kerahasiaan 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 tes statistik yang dilakukan terhadapnya.

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.[AND03]

Beberapa algoritma enkripsi memiliki kelemahan pada kunci yang digunakan. Untuk itu, kunci yang lemah tidak boleh digunakan. Selain itu, panjangnya kunci, yang biasanya dalam ukuran bit, juga menentukan kekuatan dari enkripsi. Kunci yang lebih panjang biasanya lebih aman dari pada kunci yang pendek. Jadi enkripsi dengan menggunakan kunci 128-bit lebih sukar dipecahkan dengan algoritma enkripsi yang sama tetapi memiliki kunci 56-bit. Semakin panjang sebuah kunci, semakin besar keyspace yang harus dijalani untuk mencari kunci dengan cara *brute force attack* atau coba-coba karena keyspace yang harus dilihat merupakan pangkat bilangan dari 2. Jadi kunci 128-bit

2^{128}, sedangkan kunci 56-bit memiliki keyspace 2^{56}. Artinya semakin lama kunci baru bisa ditemukan.[AND03]

2.3 Fungsi Algoritma Kriptografi

Sampai sekarang, berbagai algoritma kriptografi telah diusulkan dan masing-masing mempunyai karakteristik yang berbeda-beda. Di antara kaakteristik-karakteristik itu paling mendasar yang akan digunakan pada sistem jaringan, jaringan komputer maupun internet. Komponen-komponen yang sangat penting adalah *secrecy*, *integrity*, dan *authenticity*.

1. *Secrecy*

   Komponen yang digunakan untuk menjaga pesan yang biasanya digunakan oleh seseorang yang mengirim pesan. Komponen ini hanya mengizinkan seseorang yang tahu akan kunci pada pesan yang telah dienkripsi dengan algoritma kriptografi.

2. *Integrity*

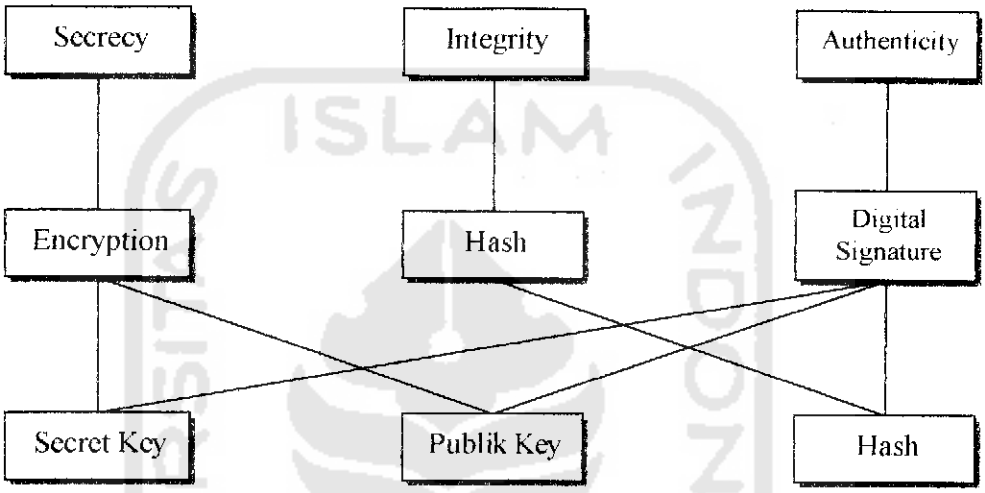
   Komponen yang digunakan untuk memeriksa apakah sebuah pesan telah diubah pada saat pengiriman, biasanya menggunakan algoritma *hash*

3. *Authenticty*

   Komponen untuk membuktikan asli atau tidaknya dokumen atau pesan yang dipakai sekelompok orang dalam bertransaksi.

2.4 Tipe dan Karakteristik Algoritma Kriptografi

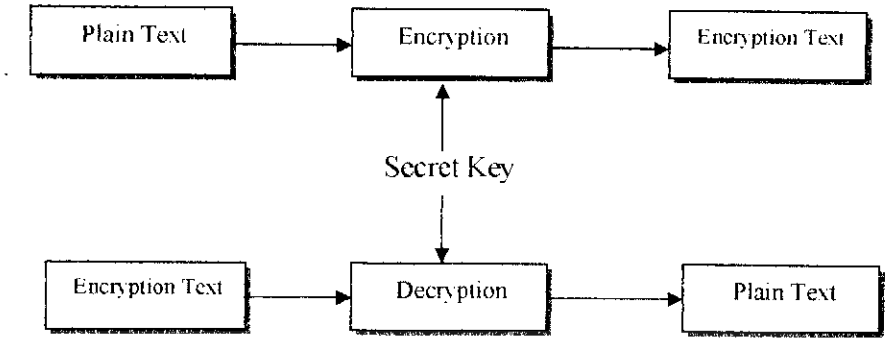
Karakteristik kunci yang menggunakan algoritma kriptografi dapat digolongkan sebagai berikut: algoritma kriptografi kunci rahasia (simetris), algoritma kriptografi kunci publik (asimetris) dan algoritma hash.[AND03]



Gambar 2.7 Tipe dan Karakteristik Algoritma Kriptografi

2.4.1 Algoritma Kriptografi Kunci Rahasia

Dalam algoritma kriptografi kunci rahasia, kunci digunakan untuk enkripsi data dan tidak diberi kuasa kepada publik melainkan hanya kepada orang tertentu yang tahu dan dapat membaca data yang dienkripsi. Karakterisitik algoritma kriptografi kunci rahasia adalah bahwa kunci enkripsi sama dengan kunci dekripsi. Algoritma kriptografi kunci rahasia juga disebut algoritma kriptografi kunci simetris. Untuk menggunakan algoritma kriptografi kunci rahasia dalam lingkungan komunikasi, kedua belah pihak yang berkomunikasi satu dengan yang lainnya harus membagi kunci enkripsi sebelumnya.[WAH03].



Gambar 2.8 Algoritma Kriptografi Kunci Rahasia

Ada beberapa model enkripsi yang termasuk dalam golongan ini, diantaranya adalah: *Simple Substitution Chiper, DES, Triple DES, Rivest Code 4 (RC4), IDEA, Blowfish, Twofish ,Vigenere cipher, enigma cipher*, dan lainnya.

2.4.2 Algoritma Kriptografi Kunci Publik

Algoritma publik-key juga disebut algoritma asimetris yang dirancang sehingga key yang digunakan untuk enkripsi berbeda dengan key yang digunakan untuk dekripsi. Selanjutnya key dekripsi tidak dapat dihitung dengan dari key enkripsi. Algoritma tersebut disebut public-key karena key enkripsi dapat dibuat secara public. Orang asing dapat menggunakan key enkripsi tersebut untuk mengenkripsi sebuah pesan, tetapi hanya orang tertentu dengan key dekripsi sepadan dapat mendekripsi pesan tersebut. Dalam sistem ini key enkripsi sering disebut public key sedangkan key dekripsi sering disebut private key.

Cara kerja enkripsi ini secara singkat dapat diterangkan sebagai berikut. Setiap orang yang menggunakan enkripsi ini hanya mempunyai dua buah kunci, satu disebut kunci pribadi (private key) dan yang lain disebut kunci publik yang

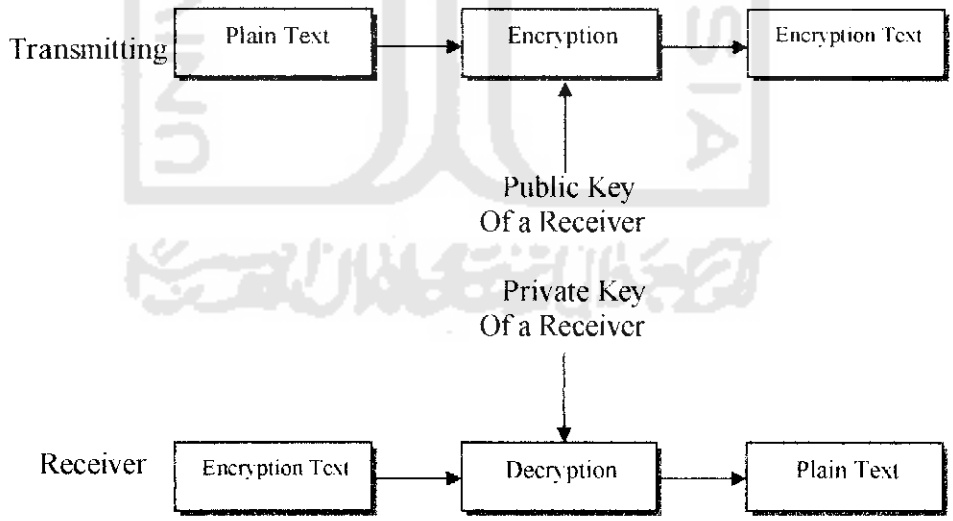
disebarkan ke orang lain. Contoh jika si A hendak mengirim pesan kepada si B, si A perlu mengenkripsi pesan itu dengan kunci publik milik B. Pesan si A yang telah di enkripsi dengan menggunakan kunci publik si A hanya bisa dibuka dengan menggunakan kunci pribadi si B, pesan ini tidak bisa dibuka dengan kunci publik itu sendiri. Jadi, si B wajib untuk menjamin keamanan kunci pribadinya.

Pada metode kriptografi asimeteris digunakan kunci umum untuk enkripsi dan kunci pribadi untuk dekripsi. Bila kunci umum dinotasikan dengan 'PK' dan kunci pribadi dinotasikan dengan 'SK' maka proses enkripsi-dekripsi metode kriptografi asimeteris dapat dinotasikan dengan :

E_{PK} (M) = C → Proses Enkripsi

D_{SK} (C) = M → Proses Dekripsi

D_{SK} (E_{PK} (M)) = M



Gambar 2.9 Algoritma Kriptografi Kunci Publik

Ada beberapa algoritma yang terkenal dari cara enkripsi kunci publik key, misalnya *Sistem Diffie Hellman, RSA,* dan *PGP.*

2.4.3 Algoritma Hash

Fungsi *hash* mengurangi data dari ukuran yang berubah-ubah menjadi ukuran yang khusus. Fungsi *hash* dibutuhkan dalam bagian konfigurasi sistem untuk memudahkan pengecekan terhadap kelebihan data. Seluruh data dapat diperiksa untuk melihat apakah data yang berkapasitas besar dapat diulang, sebab hal ini akan mendatangkan kerugian besar dalam kecepatan dan waktu.

Fungsi *hash* digunakan dalam kriptografi, yaitu dalam hal membagi atribut yang mirip, terutama dalam hal tanda tangan digital. Biasanya sering dikombinasikan dengan fungsi kriptografi yang punya integritas, seperti dalam hal kombinasi antara algoritma tanda tangan digital dan fungsi *hash* atau fungsi *hash* dengan algoritma kriptografi kunci rahasia.

Beberapa contoh pada algoritma hash yang terkenal antara lain : MD5, SHA-1, dan RIPEMD-160.

2.5 Stream Cipher

Kriptosistem kunci simetris adalah tipe yang sangat penting dalam kriptosistem modern saat ini. Sistem kunci simetris adalah dimana enkripsi dan dekripsi menggunakan kunci yang sama. Pada umumnya kriptosistem kunci simetris lebih cepat daripada kriptosistem kunci publik. Ada dua tipe pada sistem kunci simetris yaitu stream cipher dan block cipher.

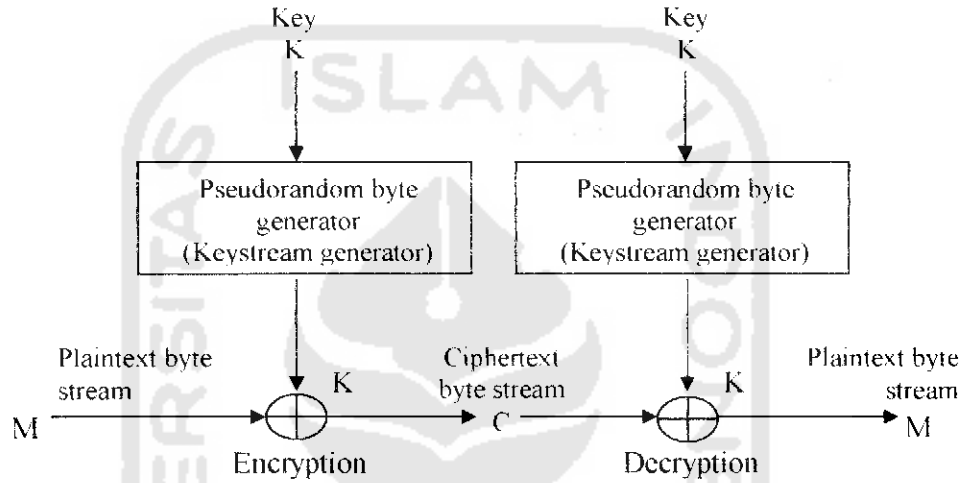
Stream cipher adalah suatu cipher dimana input data adalah enkripsi satu bit atau satu byte dalam satu waktu. Stream cipher disebut juga *state cipher* karena enkripsi dari sebuah bit bergantung pada keadaan *stream*. Stream cipher dapat bekerja seperti block cipher dengan ukuran blok yang sangat kecil. Sedangkan block cipher memproses sekaligus sejumlah tertentu data (biasanya 64 bit atau 128 bit blok), contoh dari block cipher adalah Blowfish, DES, GOST, IDEA, Twofish, RC5, RC6.

Stream cipher terdiri dari *Pseudorandom Number Generator* (PRNG) dan gerbang Eklusive OR (XOR). PRNG diinisialisasi dengan sebuah kunci, dan keluaran berupa urutan atau rangkaian bit yang disebut dengan *keystream*. Proses enkripsi yaitu dengan mengXORkan bit plaintext dengan bit keystream sedangkan proses dekripsi yaitu dengan mengXORkan bit ciphertext dengan bit keystream. Ada dua tipe stream cipher yaitu *self synchronizing stream cipher*, menggunakan beberapa bit ciphertext sebelumnya untuk menghitung keystream, sedangkan yang kedua adalah *synchronous stream cipher* dimana keystream dihasilkan secara bebas dari plaintext dan ciphertext.

Gambar 2.10 adalah diagram dari struktur stream cipher. Pada struktur ini sebuah kunci sebagai input ke pseudorandom bit generator yang menghasilkan aliran (*stream*) 8 bit yang acak. Pseudorandom stream tidak dapat diprediksi tanpa mengetahui dari kunci masukan. Output dari generator disebut dengan *keystream* yang dikombinasikan dengan satu byte dalam satu waktu dengan plaintext dengan operator Eklusive OR (XOR). Tabel kebenaran pada XOR adalah sebagai berikut:

Tabel 2.2 Tabel Kebenaran XOR

P	Q	P ⊕ Q
1	1	0
1	0	1
0	1	1
0	0	0



Gambar 2.10 Stream Cipher Diagram [Will00]

Misalnya jika byte yang dibangkitkan dengan generator adalah 01101100 dan byte plaintext adalah 11001100 maka hasil ciphertext adalah :

1 1 0 0 1 1 0 0 ⟶ Plaintext

0 1 1 0 1 1 0 0 ⟶ Key Stream

⊕

1 0 1 0 0 0 0 0 ⟶ Ciphertext

Untuk proses dekripsi menggunakan rangkaian pseudorandom yang sama:

1 0 1 0 0 0 0 0 ⟶ Ciphertext

0 1 1 0 1 1 0 0 ⟶ Key Stream

⊕

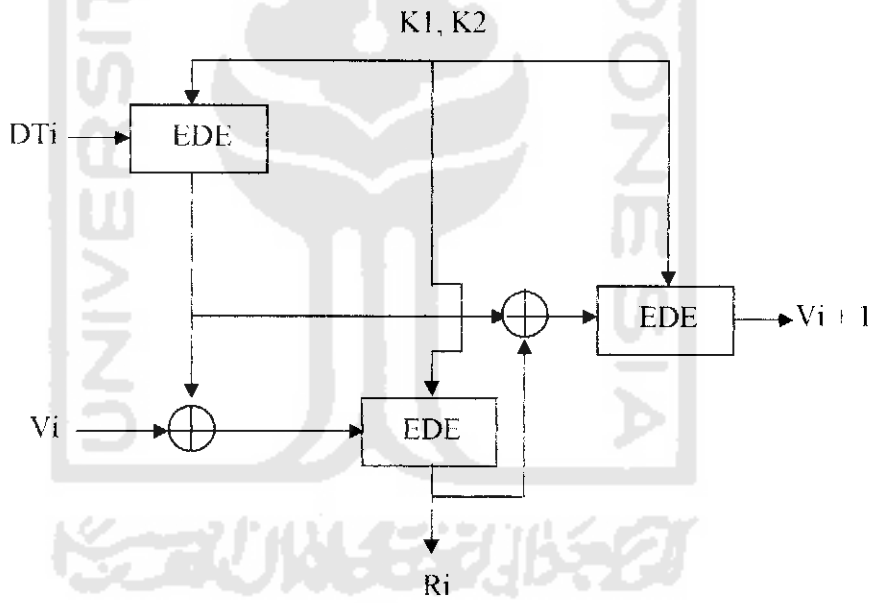
1 1 0 0 1 1 0 0 ⟶ Plaintext

Yang termasuk dalam stream cipher adalah RC4, SEAL, Wake, ISAAC, CHAMELON, SOBER, A5, Oryx dan HELIX.

2.6 *Pseudorandom Number Generators (PRNG)*

Suatu *Pseudorandom Number Generator* (PRNG) adalah sebuah algoritma yang menghasilkan urutan angka-angka, dimana unsur-unsurnya tidak terikat satu sama lainnya. Output dari PRNG tidak sepenuhnya random, hanya mendekati beberapa sifat dari angka-angka yang random. Angka-angka pseudorandom merupakan bagian dari sistem perhitungan modern., dari kriptografi dengan menggunakan metode Monte Carlo untuk simulasi sistem-sistem fisik. Metode Monte Carlo adalah metode untuk pemecahan berbagai macam permasalahan komputasional dengan menggunakan angka-angka yang acak atau yang lebih dikenal dengan *pseudorandom number*. Analisis matematika diperlukan untuk memastikan bahwa angka-angka yang dihasilkan cukup acak. Kelas algoritma yang umum antara lain :*Linear Congruential Generators, Lagged Fibonacci Generators, Linear Feedback Shift Register, dan Generalised Feedback Shift Register.* Contoh Algoritma yang baru adalah *Blum-Blum Shub, Fortuna dan Mersenne Twister.*

PRNG bekerja pada komputer deterministic (berbeda dengan komputer quantum) yang menggunakan algoritma deterministic, dimana outputnya mempunyai sifat rangkaian acak yang tidak dapat dilihat. Jika generator menggunakan jumlah memori yang telah ditetapkan kemudian memberikan nomor yang cukup dari iterasi, generator akan kembali ke status internal dua kali, setelah itu akan diulang selamanya. Suatu generator yang tidak berkala dapat didesain, tetapi kebutuhan memorinya akan bertambah. PRNG dapat dimulai dari nilai awal yang berubah-ubah atau status yang telah ditentukan dan akan menghasilkan urutan yang sama dari nilai awal tersebut.[WII.00]



Gambar 2.11 Pseudorandom Number Generator [ANSI X9]

Keterangan:

DTi : Tanggal / waktu awal pada tahap pembangkitan ke-i.

Vi : Nilai seed pada tahap pembangkitan ke-i.

Ri : Angka-angka pseudorandom yang dihasilkan oleh tahap pembangkitan ke-i.

K1, K2 : Kunci-kunci DES yang digunakan pada setiap tingkatan. Maka diperoleh persamaan.[WILL00]:

$$Ri = EDE\ K1, K2\ [Vi \oplus EDE\ K1, K2\ [DTi]\ ]$$

$$Vi+1 = EDE\ K1, K2\ [Ri \oplus EDE\ K1, K2\ [DTi]\ ]$$

Input : Dua masukkan pseudorandom yang menggerakan generator. Yang pertama adalah 64 – bit representasi dari tanggal dan waktu sekarang dimana diperbaharui setiap pembangkitan nomor. Dan yang kedua adalah 64-bit nilai seed yang diinisialisasikan ke beberapa nilai arbitrary dan diubah selama proses pembangkitan.

Key : Generator menggunakan tiga triple DES modul enkripsi. Semuanya menggunakan sepasang 56-bit kunci yang sama dan digunakan hanya untuk pembangkitan nomor pseudorandom.

Output : Output terdiri dari 64-bit nomor pseudorandom dan 64-bit nilai seed.

Dimana EDE mengacu pada urutan enkripsi dekripsi yang menggunakan dua kunci triple DES. Beberapa faktor berperan untuk menguatkan kriptografi. Teknik ini melibatkan semua 2 bit kunci dan 3 EDE enkripsi untuk total dari 9 DES enkripsi. Skema ini digerakkan oleh dua pseudorandom masukkan, nilai tanggal dan nilai waktu, dan nilai seed yang dihasilkan oleh generator yang mana berbeda dari nilai pseudorandom yang dihasilkan oleh generator. Akan sangat sulit untuk menyimpulkan Vi + 1 dari Ri karena penambahan operasi EDE digunakan untuk menghasilkan Vi + 1.

2.7 Konsep Algoritma Rivest Code 4 (RC4)

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.

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*).

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.

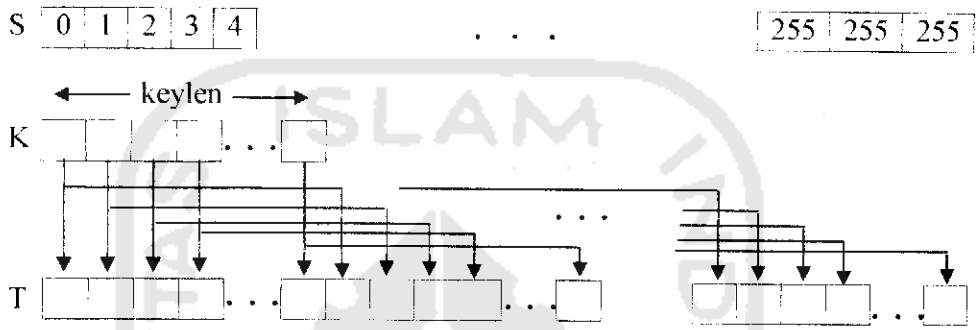
Sekali variabel enkripsi dihasilkan dari *key setup*, langkah selanjutnya adalah masuk ke fase *ciphering* di mana dalam proses ini hasilnya akan diXORkan dengan *plaintext*. Sekali penerima mendapat pesan yang dienkripsi, langkah selanjutnya adalah mendekripsinya dengan XOR pesan yang dienkripsi dengan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

2.7.1 Inisialisasi S-Box

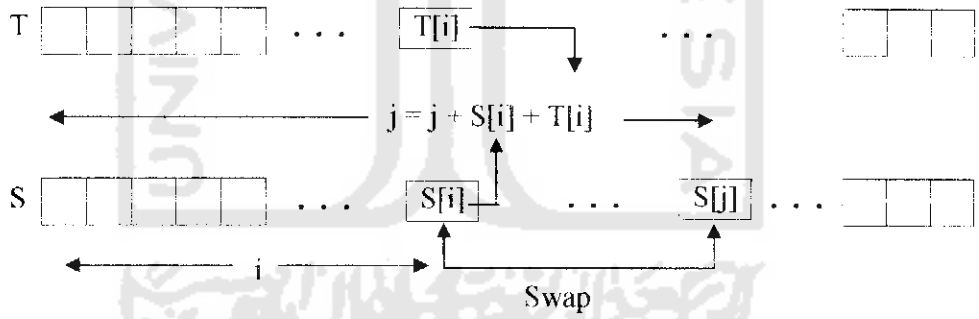
RC4 menggunakan variabel yang panjang kuncinya dari 1 sampai 256 byte (8 – 2048 bit) yang digunakan untuk menginisialisasikan tabel sepanjang 256 byte yang disebut dengan *S-Box* atau disebut juga S, dengan elemen S[0], S[1], ...S[255] yang berisi permutasi dari 0 sampai 255 dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Pertama isi secara berurutan $S_0 = 0$, $S_1 = 1$,...,$S_{255} = 255$. Kemudian isi array 256 byte lainnya dengan kunci yang diulangi sampai seluruh array K_0, K_1,...,K_{255} terisi seluruhnya. Set indeks j dengan nol, berikut langkahnya.[WILL00] :

```
/ * Inisialisasi  S dan K* /
for i = 0 to 255
S[i] = i
T[i] = K [i mod keylen]
```

```
/ * Inisialisasi Permutasi S */
j = 0
for i = 0 to 255
j = (j + S[i] + T[i]) mod 256
swap (S[i], S[j])
```



Gambar 2.12 Inisialisasi Keadaan S dan T

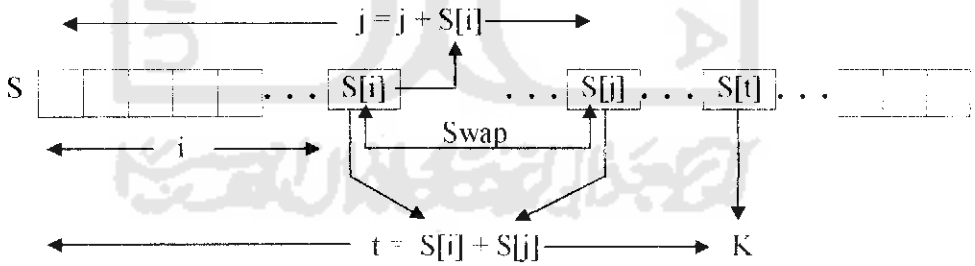


Gambar 2.13 Inisialisasi Permutasi S

2.7.2 Stream Generator

Satu dari S vector diinisialisasi, dimana kunci masukkan tidak lagi digunakan. Stream generator melibatkan mulai dengan S[0] sampai S[255] dan untuk masing-masing S[i], kemudian menukarkan S[i] dengan byte yang lain pada S menurut pola dari konfigurasi yang sekarang dari S. Setelah S[255] dicapai, proses terus berlanjut dimulai lagi pada S[0].[WIL00]

```
/ * Stream Generator */
i = 0
j = 0
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i], S[j])
t = (S[i] + S[j]) mod 256
K = S[t]
```



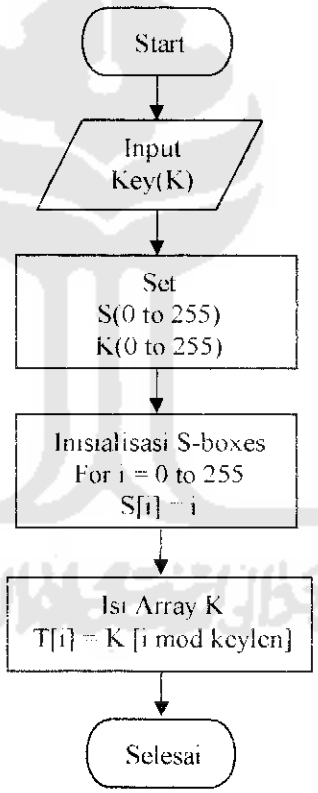
Gambar 2.14 Stream Generator

2.7.3 Proses Enkripsi

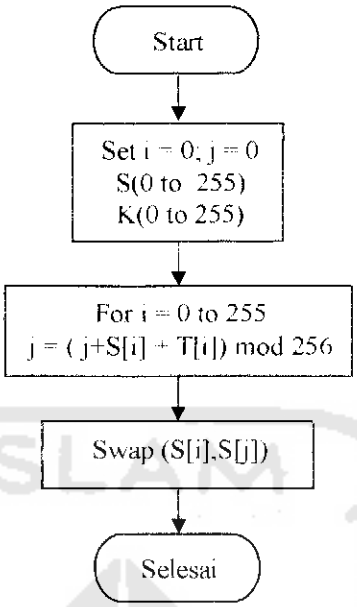
Untuk proses enkripsi, setelah proses inisialisasi S-boxes dan keystream generator maka diperoleh *keystream*, keystream inilah yang kemudian di XORkan dengan plaintext untuk menghasilkan ciphertext.

2.7.4 Proses Dekripsi

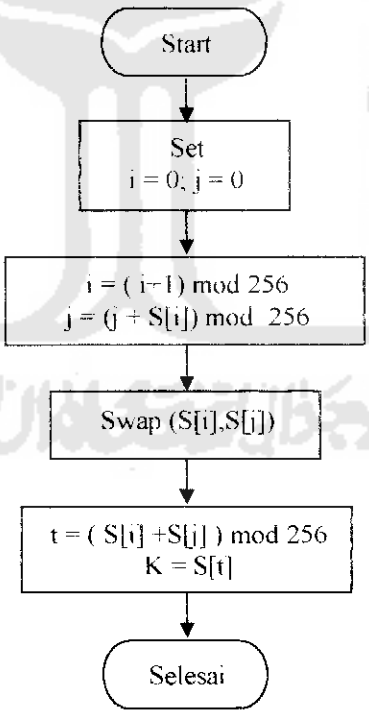
Proses dekripsi yaitu dengan mengXORkan ciphertext dengan keystream untuk memperoleh file semula (plaintext).



Gambar 2.15 Flowchart Inisialisasi S-Boxes dan K



Gambar 2.16 Inisialisasi Permutasi S



Gambar 2.17 Flowchart Stream Generator

Untuk mengetahui lebih jelas proses inisialisasi S-boxes, keystream generator dan proses enkripsi dan dekripsi akan diberi contoh dengan menggunakan kunci 8 bit.dengan tujuan penyederhanaan:

Misalkan kita ciptakan 8 bagian bit S-boxes yang terdiri dari jumlah angka 0 sampai dengan 7.

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
0 1 2 3 4 5 6 7

Ciptakan 8 kunci bit yang terdiri dari pengulangan kunci utama untuk mengisi seluruh bagian array. (Misal angka 1 dan 5)

K_i = K_0 K_1 K_2 K_3 K_4 K_5 K_6 K_7
1 5 1 5 1 5 1 5

Untuk operasi pencampuran berikut ini, kita akan menggunakan variabel i dan j dengan inisialisasi 0. Operasi pencampuran adalah iterasi dari rumus ($j + S_i + K_i$) mod 8 yang diikuti dengan penukaran S_i dan S_j.

Iterasi ke -1

Untuk i = 0

j = (j + S[0] + K[0]) mod 8
= (0 + 0 + 1) mod 8
= 1

Swap (S[0],S[1])

Si = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
1 0 2 3 4 5 6 7

Iterasi ke-2

Untuk i = 1

j = (j + S[1] + K[1]) mod 8
= (1 + 0 + 5) mod 8
= 6

Swap (S[1],S[6])

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
1 6 2 3 4 5 0 7

Iterasi ke-3

Untuk i = 2

j = (j + S[2] + K[2]) mod 8

= (6 + 2 + 1) mod 8

= 9 mod 8

= 1

Swap (S[2], S[1])

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
1 2 6 3 4 5 0 7

Iterasi Ke-4

Untuk i = 3

j = (j + S[3] + K[3]) mod 8

= (1 + 3 + 5) mod 8

= 9 mod 8

= 1

Swap (S[3],S[1])

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
1 3 6 2 4 5 0 7

Iterasi Ke-5

Untuk i = 4

j = (j + S[4] + K[4]) mod 8

= (1 + 4 + 1) mod 8

= 6 mod 8

= 6

Swap (S[4],S[6])

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
1 3 6 2 0 5 4 7

Iterasi Ke-6

Untuk i = 5

j = (j + S[5] + K[5]) mod 8

= (6 + 5 + 5) mod 8

= 16 mod 8

= 0

Swap (S[5],S[0])

S_i =	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	5	3	6	2	0	1	4	7

Iterasi Ke-7

Untuk i = 6

j = (j + S[6] + K[6]) mod 8

= (0 + 4 + 1) mod 8

= 5 mod 8

= 5

Swap (S[6],S[5])

S_i =	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	5	3	6	2	0	4	1	7

Iterasi Ke-8

Untuk i = 7

j = (j + S[7] + K[7]) mod 8

= (5 + 7 + 5) mod 8

= 17 mod 8

= 1

Swap (S[7],S[1])

S_i =	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
	5	7	6	2	0	4	1	3

Bangkitkan sebuah bit yang acak untuk enkripsi. Beri nilai i dan j dengan 0 kemudian beri nilai j = (j + S_i) mod 8. Kemudian tukarkan S_i dengan S_j sehingga hasilnya menjadi: (S_i + S_j) mod 8. Hasil dari rumus ini kemudian disimpan dalam S_t.

i = 0

j = 0

i = (i + 1) mod 8

= (0 + 1) mod 8

= 1

j = (j + S[i]) mod 8

= (0 + S[1]) mod 8

= (0 + 7) mod 8

= 7

Swap (S[1],S[7])

S_i = S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
5 3 6 2 0 4 1 7

t = (S[1] + S [7]) mod 8

= (3 + 7) mod 8

= 10 mod 8

= 2

St = 2 → keystream

Hasil ini yang kemudian diXORkan dengan plaintext untuk menghasilkan ciphertext, sedang untuk proses dekripsi adalah dengan mengXORkan ciphertext dengan keystream.

BAB III

ANALISIS KEBUTUHAN PERANGKAT LUNAK

3.1 Metode Analisis

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh kedalam bagian komponennya untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan.

Tahap analisis merupakan tahap yang paling penting, karena kesalahan di dalam tahap ini akan menyebabkan juga kesalahan ditahap selanjutnya. Oleh sebab itu diperlukan suatu metode yang dapat digunakan sebagai pedoman dalam pengembangan sistem.

Metode yang dipakai untuk menganalisis kebutuhan guna mendukung rancang bangun implementasi kriptografi algoritma RC4 adalah dengan metode analisis terstruktur (*Structured Approach*), yang menggambarkan secara menyeluruh target dan kebutuhan sitem yang diperlukan, schingga hasil dari analisis akan menghasilkan analisis dari kebutuhan sistem yang terstruktur dan dapat didefinisikan dengan baik dan jelas.

3.2 Analisis Kebutuhan

Analisis kebutuhan didapat setelah kita mengetahui apa saja masukan atau input serta output dari program yang ada, serta kebutuhan perangkat lunak ataupun perangkat keras dari sistem tersebut.

3.2.1 Input/Masukan

Adapun analisis input yang dibutuhkan untuk mengimplementasikan enkripsi ini adalah :

a. Untuk enkripsi file yaitu inputnya berupa dokumen elektronik (file yang telah tersimpan dalam komputer).
b. Untuk enkripsi file text inputnya berupa file dengan format *.txt yang tersimpan dalam komputer atau dengan menginputkan karakter huruf-huruf dalam ASCII.

3.2.2 Output/Keluaran

Output atau keluaran yang diharapkan dari enkripsi ini adalah:

a. Untuk enkripsi file maka outputnya berupa file berekstensi *.enc
b. Untuk file text yang dienkripsi maka outputnya berupa karakter ASCII yang tidak dapat terbaca.
c. File yang dihasilkan dalam bentuk file enkripsi maupun dekripsi.

3.2.3 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk mendukung berjalannya sistem ini adalah :

1. Visual Basic 6.0 merupakan bahasa pemrograman yang digunakan untuk pengembangan dan implementasi kriptografi algoritma RC4

2. Windows 9x adalah sistem operasi yang digunakan dalam mengimplementasikan perangkat lunak.

3.2.4 Kebutuhan Perangkat Keras

Perangkat keras yang digunakan untuk Studi dan Aplikasi Enkrispi / Dekripsi dengan Menggunakan Algoritma Rivest Code 4 ini dijalankan dengan komputer yang minimal harus memenuhi spesifikasi sebagai berikut :

1. Komputer dengan prosesor pentium 133 MHz, sekelasnya atau yang lebih tinggi.
2. RAM 32 MB atau lebih.
3. Ruang Hardisk yang dibutuhkan adalah 2 GB.
4. Monitor VGA atau SVGA.
5. Mouse dan Keyboard.

3.2.5 Kebutuhan Antar Muka

Antar muka (*interface*) yang ditawarkan untuk memudahkan user dalam menggunakan perangkat lunak yang digunakan pada sistem ini yaitu menu utama yang terdiri dari :

1. Encrypt File yaitu menu untuk mengenkripsi file.
2. Encrypt Text yaitu menu untuk mengenkripsi file text.
3. Algortihm RC4 yaitu penjelasan singkat tentang algoritma RC4
4. About Program berisi tentang informasi program.