PERBANDINGAN TEKNIK KRIPTOGRAFI METODE SAPPHIRE II DAN RC4

Oleh :

Ronal Watrianthos
Dosen Prodi Manajemen Informatika, AMIK Labuhanbatu
Rantauprapat, Medan; ronaw@amik-labuhanbatu.ac.id

Abstract

Sapphire II diciptakan oleh Michael paul Johnson pada tahun 1994 yang merupakan pengembangan dari sapphire stream cipher original. Pada dasarnya metode enkripsi ini mirip dengan RC4 tetapi menggunakan plainteks dan cipherteks feedback, sehingga lebih mudah digunakan secara aman. Perbedaan lain adalah operasinya yang lebih kompleks dan menggunakan indeks yang lebih banyak yait lima buah indeks sedangkan RC4 hanya dua buah indeks. Dengan kompleksnya operasinya dan saling berkaitan operasi sebelum dan sesudahnya (sehingga tidak memungkinkan dieksekusi secara parallel) maka kecepatan enkripsi/dekripsi dari sapphire II tidak akan akan secepat RC4 (kurang lebih hanya sepertiganya).

Kelebihan dari metode ini adalah dapat digunakan sebagi generator bilangan pseudorandom(mestinya setiap metodeenkripsi dapat digunakan untuk hal ini), dapat juga digunakan sebagai hash generation.panjang knci seperti juga pada RC4 dapat mencapai 256 byte (2048 bit). Sapphire II dan RC4 merupakan stream cipher yang melakukan proses secara per byte dengan panjang kunci maksimum 256 byte sehingga ukuran data input akan sama dengan ukuran data output. Dalam hal Avalanche Effects algoritma Sapphire II secara umum beberapa persen di atas dari algoritma ini lebih unggul sedikit dibandingkan dengan RC4.

Hasil uji coba secara umum didapat metode RC4 mempunyai waktu kumputasi yang cepat yang disebabkan oleh algoritma RC4 lebih sederhana dibandingkan dengan Sapphire II.

Keyword : RC, Sapphire,

I. PENDAHULUAN

Dengan semakin pesatnya perkembangan teknologi computer,maka semakin banyak orang yang sanggup mengutak-atik data yang disimpan.untuk mencegah terjadinya pencurian data dari orang yang tidak berhak maka dikembangkanlah berbagai teknik pengamanan data,salah satu teknik yang dapat digunakank untuk menjaga keamanan data adalah dengan menggunakanperangkat lunak kriptografi.

Sapphire II diciptakan oleh Michael paul Johnson pada tahun 1994 yang merupakan pengembangan dari sapphire stream cipher original.pada dasarnya metode enkripsi ini mirip dengan RC4 tetapi menggunakn plainteks dan cipherteks feedback,sehingga lebih mudah digunakan secara aman. Perbedaan lain adalah operasinya yang lebih kompleks dan menggunakan indeks yang lebih banyak yait lima buah indeks sedangkan RC4 hanya dua buah indeks. Dengan kompleksnya operasinya dan saling berkaitan operasi sebelum dan sesudahnya (sehingga tidak memungkinkan dieksekusi secara parallel) maka kecepatan enkripsi/dekripsi dari sapphire II tidak akan akan secepat RC4 (kurang lebih hanya sepertiganya). Kelebihan dari metode ini adalah dapat digunakan sebagi generator bilangan pseudorandom(mestinya setiap metodeenkripsi dapat digunakan untuk hal ini), dapat juga digunakan sebagai hash generation.panjang knci seperti juga pada RC4 dapat mencapai 256 byte (2048 bit).

II. PERMASALAHAN

Berdasarkan ulasan di atas,maka pokok permasalahan dalam penelitian ini adlah bagaimana keunggulan dari teknik pengamanan data yang menggunakan algoritam sapphire II dengan teknik pengamanan data yang menggunakan algoritma RC4 serta bagaimana menampilkan hasil perbandingannya.

III. LANDASAN TEORI

3.1 Kriptografi

Berbagai jenislayanan komunikasi tersedia di internet,diantaranya adalah WEB, E-MAIL, MILLS (MAILING LIST),NEWSGROUPS,dan sebagainya. Dengan semakin maraknya orang memanfaatkan layanan komunikasi di internet tersebut,maka permasalahan pun bermnculan, apalagi ditambah dengan adanya hacker dan cracker.banyak orang kemudian berusaha menyiasati bagaimana cara mendeteksi keaslian dari informasi Yng dikomunikasikannya,atau menyiasati bagaimana cara mendeteksi keaslian dariinformasi yang diterimanya.untuk lebih jelasnya,akan diberikan ilustrasi dari salah satu permasalahan tersebut.seseorang pengirim pesan yang hendak mengirimkan surat elektronik (e-mail) kepada rekannya,menginginkan agar pesannya tidak dibaca oleh orang yang tidak berhak membacanya, padahal bila administrator server e-MAIL sedang iseng,sangat mungkin dia akan membaca e-EMAIL-e-MAIL yang ada server-nya.penerima pun ingin mendapat keyakinan bahwa pengirimnya merupakan orang yang dikenalnya, buku orang yang berpura-pura sebagi temannya.

Plainter dinyatakan dengan M (MESSAGE) atau P (plaintext).pesan dapat berupa aliran bit, file teks,bilmap,aliran suara yang didigitasi,gambar video digital dan sebaghainy. Namun semua pesan tadi biner juga dapat diperlakukan sebagai system bilangan biner. Karena itulah diperlukan matematika untuk menganalisisnya.plainter dapat disimpan mapun dikirim ke jaringan dalam sembarang kasus, M merupakan pesan yang akan dienkrip.

3.1.1 Aspek-Aspek Keamanan

Kriftografi tidak hanya memberikan kerahasiaan dalam telekomunikasi, namun juga memberikan komponen-komponen berikut ini:

1. **Authentication**. Penerimaan pesan dapat memastikan keaslian pengirimnya. Penyearang tidak dapat berpura-pura sebagai orang lain.
2. **Integrity**. Penerima harus dapat memeriksa apakah pesan telah dimodifikasi di tengah jalan atau tidak. Seorang penyusup seharusnya tidak dapat memasukkan tambahan ke dalam pesan, mengurangi atau mengubah pesan selama data berada di jalanan.
3. **Non Repudiation.** Pengirim seharusnya tidakdapat mengelak bahwa dialah pengirim pesan yang sesungguhnya. Tanpa kriptografi, seseorang dapat mengelak bahwa dialah pengirim *e-mail*yang sesungguhnya.
4. **Authority.**informasi yang berada pada system jaringan pada seharusnya hanya dapat dimodifikasi oleh pihak yang berwenang.modifikasi yang tidak diinginkan, dapat berupa penulisan tambahan pesan, pengbahan isi, pengubahan status, pengapusan, pembuatan pesan baruh, pemalsuan,atau menyalin pesan untuk digunakan kemudian oleh penyerang.

   Terdapat persyaratan penting bagi interaksi di dunia nyata. Seseorang yang mempunyai identitas diri, baik berupa KTM, SIM atau passport diharapkan bahwa identitas diri itu memang sah dan benar isinya. Inilah yang diberikan oleh otentikasi,integritas dan non repudiation.

3.1.2 Algoritma Dan Kunci

Algoritma kriptografi selalu terdiri dari dua bagian yaitu fungsi enkripsi dan dekripsi.bila keamanan algoritma tergantung pada kerahasiaan algoritma bekerja, maka algoritma tersebut dikatakan algoritma terbatas (terbatas kemampuanya).algoritma terbatas mempunyai sejarah yang menarik,namun sayangnya tidak cukup baik untuk digunakan pada masa sekarang ini.sejumlah pengguna(yang tidak dalam satu grup) tidak dapat menggunakanya bersama-sama, sehingga tiap kali seorang pengguna meninggalkan grupnya,pemakai lain dalam grup tersebut harus mengganti algoritma,agar algoritma yang mereka gunakan tidak diketahui kelompok lain.dan bilah salah satu tanpa sengaja menampakanalgoritma keluar grupnya,grup tersebut harus mengganti algoritmanya.

Bahkan lebih buruk lagi, algoritma terbatas tidak mengizinkan control kualitas atau standarisasi.setiap grup pemakai harus mempunyai algoritma tersendiri. Mereka tidak dapat menggunakan produk perusahaan lain karena kalua demikian tentu orang lain dapat juga membeli produk tersebut dan kemudian memecahkan algoritmanya sehingga data yang dienkrip dengan algoritma tersebut tidak aman lagi.anggota grup tersebut harus menulis sendiri algoritma dan implementasinya. Sehingga jika tidak ada anggota pun yang ahli kriptografi,mereka tidak akan tahu apakah algoritmanya aman atau tidak.meskipun mempunyai kelemahan yang besar.algoritma terbatas sangat terkenal untuk aplikasi keamanan tingkat rendah.

Kriptografi modern menyelesaikan masalah ini dengan hanya merahasiakan kunci (key) saja tanpa harus menyembunyikan

algoritmanya sendiri.kunci (K) dapat juga disebut sebagai password.keamanan enkripsi hanya tergantung pada kunci,dantidak tergantung apakah algoritmanya ilihat orang lain atau tidak. Rentang kemungkinan nilai kunci ini disebut keyspace.

Bila keseluruhanya keamanan algoritma ini tergantung kunci dan tak satu pun yang didasarkan pada detil algoritma maka algoritma inidapat dipblikasikan dan dianalisis oleh semua orang.produk-produk yang menggunakan algoritma tersebut dapat diproduksi secara massal.tidak masalah bila seseorang mengetahui algoritma tersebut,jika dia tidak mengetahui kunci rahasianya, dia tetap tak dapat membuka pesan.

Contoh system sejenis ini adalah kartu kredit.semua kartu kredit yang beredar di seluruh dunia menggunakan algoritma kriptografi yang sama yaitu DES (data encryption standard) dan RSA. Semua orang boleh mengetahui isi inci algoritma DES dan RSA.namun pengetahuan terhadap algoritma ini tidak membantu proses pembongkaran kodenya. Hanya dengan pengetahuan kuncinya sajalah orang dapat melakukan dekripsi.artinya dengan memberikan kepada setiap kartu kunci, keamanan kartu kredit dapat diandalkan.keuntungan system seperti ini adalah bahwa berbagai produsen kartu kartu kredit yang berbeda dapat menggunakan algoritma keamanan yang sama sehingga dapat menggunakan algoritma keamanan yang samasehingga dapatdigunmakan pada mesinperusahaan yang berbeda.

3.1.3 Crytanalysis

Tugas utama kriptografi adalah untuk menjaga agar baikplaintext maupun kunci ataupun keduanya tetap terjaga kerahasiannya dari penyedap (disebut juga sebagai lawan,penyerang,pencegat, penyelundup pesan,musuh, attacker, dan sebagainya). Pencegat pesan rahasia diasumsikan mempunyai akses yang lengkap ke dalam saluran komunikasi antara pengirim dan penerima.ini sangat mudah terjadi pada jalur internet dan saluran telepon.

CRYTANALYSIS (analisis sandi)adalah ilmu untuk mendapatkan plaintext pesan tanpaharus mengetahui kunci secara wajar. Pemecahan sandi rahasia yang berhasil akan menghasilkan plaintext atau kunci.analisis sandi juga dapat menemukan kelemahan dalam kriptosistem yang pada akhirnya dapat menemukan kunci atau plaintext.kehilangan kunci melalui peralatan non cryptanalytic disebut compromise. Dengan kata lain,analisis sandi merupakan kebalikan dari kriptografi.

Usaha analisis sandi disebut juga dengan attack (serangan).asumsi dasar dalam cryptanalysis pertama kali diungkapkan oleh Dutchman A. KERCKHOFFS pada abad ke-19, yaitu bahwa kerahasiaan harusterletak pada kunci. Kerckhoffs mengasumsikan bahwa analisis sandi mempunyai detil lengkap algoritma kriptografi dan implementasinya. Meskipun dalam dunia nyata,analisis sandi mungkin tidak mempunyai informasi yang sedemikian detil,adalah baik untuk membuat asumsi untuk sejenis itu.

Lard Knudsen menggolongkan berbagai jenis jenis pemecahan algoritma:

1. **Total breack.**seorang analisis berhasil menemukan kunci, K yang digunakan untukmelindungi data-data,sedemikian sehingga Dk(C)=p.
2. **Global deducation.** Analisis sandi mendapatkan algoritma alternative,A, yang ekivalen dengan Dk(C), tanpa mengetahui K.
3. **Instance (local) deducation.** Analisis sandi mendapatkan plaintext ataupun ciphertext yang disadap.
4. **Information deducation.**analisis sandi memperoleh beberapa informasi mengenai kunci atau plaintext, dan sebagainya.

Untuk mengukur kompleksitas serangan terdapat berbagai jenis cara,

Diantaranya:

a) **Data complexity.** jumlah data yang diperlukan sebagai input attack.semakin sedikit jumlah data yang diperlukan untuk melakukan attack,berarti kualitas algoritma yang digunakan semakin tidak baik.
b) **Processing complexity.** lama waktu yang tersediah untuk melakukan attack. Ini sering disebut faktor kerja.semakin cepat waktu yang dibutuhkan,berarti semakin buruk kualitas algoritma yang digunakan.
c) **Storage requirements.** Jumlah memori yang dibutuhkan untuk melakukan attack.

Kompleksitas dinyatakan sebagai orde besarnya.jika algoritma memiliki kompleksitas 2128, maka sejumlah 2 128 operasi diperlukan untuk memecahkan algoritma.operasi ini mungkin komplek dan banyak menghabiskan waktu.jika anda mempunyai kecepatan komputasi prosesor hingga 1000 juta operasi per detik dan anda menggunakan satu juta prosesor untuk melakukan tugas ini,maka anda akan memerlukan waktu 1016

tahun untuk menemukan kunci.ini berarti satu juta kali umur alam smesta.

Sementara kompleksitas attack konstan (sampai analisis sandi mendapatkan cara yang lebih baik), kekuatan komputasi meningkat pesat. Attack akan meningkat pesan dalam mesin parallel. Tugas dapat dipecah-pecah kedalam milyaran bagian yang kecil-kecil dan tak satu pun dari prosesor-prosesor tersebut memerlukan interaksi dengan lainnya. Kriptosistem yang baik dirancang untuk menghadapi kemajuan peningkatan kecepatan komputasi hingga tahun-tahun kedepan.

3.1.4 Jenis-Jenis Serangan Cryptanalyst

Terdapat beberapa jenis serangan yang mungkin dilakkan oleh pemecah kode (cryptanalyst), dengan asumsi algoritma enkripsinya telah dikenal luas:

1. **Ciphertext onlay attack.** Cryptanalyst hanya memiliki beberapa pesan ciphertext, semuanya dienkripsi dengan algoritma yang sama. Cryptanalyst tidak mengetahui kunci dan plaintext-nya pekerjaan cryptanalyst adalah mendapatkan plaintext atau mencari kuncinya terlebih dahulu.
   Diketahui: CI=EK(PI),C2=EK(P2), C3=EK(P3),…
   dicari: K dan P1,P2,P3,…
2. **known-plaintext attack**. Cryptanalyst dapat mengetahui beberapa plaintext beserta **ciphertext**-nya.misalnya dalam sebuah surat diyakini terdapat tulusan hormat kami’’dan analis sandi telah mendapatkan pula cipher surat tersebut. Maka analis sandi dapat memperkirakan cipher mana yang berkenaan dengan plaintext ‘’hormat kami’’ tersebut. Tugas ciptanalyst selanjuntnya adalah menemukan kunci, sehingga dia dapat menemukan plaintext apabilah telah mendapatkan ciphertext baru lainnya yang dienkripsi dengan algoritma yang sama.
   diketahui: P1,P2,P3,... SERTA C1,C2,C3,…..
   di cari: K atau P lainnya.
3. CHOSEN-PLAINTEXT ATTACK. Cryptanalyst tidak hanya mengetahui sejumlah plaintext dan ciphertext-nya seperti pada kasus 2 di atas, namun bebas memilih sejumlah plaintext tertentu sedemikian sehingga kuncinya dapat di tebak. Tugas analis sandi adalah menebak kunci hal ini dapat terjadi dengan suatu rekayasa agar lawan mengenkrip pesan kita dengan kunci yang akan di bongkar.misalnya kita berpura-pura sebagai atasan petugas mengirim kode rahasia melalui e-mail dan memintanya mengenkrip pesan kita.
   Diketahui: P1,P2,P3,…. Serta C1,C2,C3,… dan cryptanalyst-lah yang memilih P1,P2,P3,
   Dicari: K atau P lainnya.
4. Adaptive-chosen-plaintext attack. Serangan ini merupakan kasus khusus dari serangan jenis ke tiga di atas. Tidak hanya dapat memilih plaintext ysng dienkrip, cryptanalyst juga dapat memodifikasi pilihannya berdasar hasil enkripsi sebelumnya. Dalam Choses-Plaintext Attack, cryptanalyst mungkin hanya dapat memilih satu blok besar plaintext untuk dienkripsi, sedangkan pada serangan ini, dia dapat memilih blok plaintext yang lebih kecil dan kemudian memilih lainnya berdasarkan hasil sebelumnya.
5. Chosen-Ciphertext Attack. Cryptanalyst dapat memilih ciphertext yang berbeda untuk didekripsi dan mempunyai akses terhadap plaintext yang dienkripsi. Sebagai contoh, cryptanalyst mempunyai akses ke kotak elektronik yang dapat melalukan deskripsi secara otomatis. Pekerjaan yang harus dilakukan adalah menemukan kunci K.
   Diberikan: C1,P1 = DK (C1), C2, P2 = DK (C2),…, C1, P1 = DK (Ci)
   Dicari: K
6. Chosen Text. Merupakan gabungan chosen plaintext dan chosen chipertext attack. Attack ini juga dapat digunakan terhadap algoritma kunci publik. Chosen-chipertext attack kadang-kadang efektif menghadapi algoritma simetri dengan baik. Bila chosen-chipertext attack digabung dengan chosen-plaintext attack maka disebut sebagai chosen-text attack.

Bila ditabelkan jenis-jenis serangan diatas akan diperoleh table 2.1 berikut ini**.**

Tabel 2.1 Jenis Serangan Kriptografi

Jenis serangan	Yang diketahui cryptanalyst
1.Ciphertext only attack (hanya tau kode rahasia)	Algoritma enkripsi ciphertextyang akan dibaca
2. Known plaintext (mengetahui plaintext tertentu)	Algoritma enkripsi ciphertext yang akan dibaca sepasang/lebih plaintext-ciphertext yang disusun dengan kunci rahasia tertentu

3.Chosen plaintext (dapat memilih plaintext)	Algoritma enkripsi ciphertext yang akan dibaca plaintext yang dipilih cryptanalyst, bersama dengan ciphertext pasangannya yang dibangkitkan dengan kunci rahasia tertentu
4.Adaptive chosen plaintext attack	Algoritma enkripsi ciphertext yang akan dibaca plaintext dapat dipilih lebih khusus oleh cryptanalyst
5.Chosen ciphertext (dapat memilih ciphertext tertentu yangdiinginkan)	Algoritma enkripsi ciphertext yang akan dibaca ciphertext yang isi pokoknya diketahui,dipilih oleh cryptanalyst,bersama dengan plaintext (terdekripsi) pasangannya yang dibangkitkan dengan kunci tertentu
6.Chosen text	Algoritma enkripsi ciphertext yang akan dibaca plaintext yang dipilih cryptanalyst, bersama dengan ciphertext pasangannya yang dibangkitkan dengan kunci rahasia tertentu ciphertext yang isi pokoknya diketahui, dipilih oleh cryptanalyst, bersama dengan plaintext (Terdekripsi) pasangannya yang dibangkitkan dengan kunci tertentu

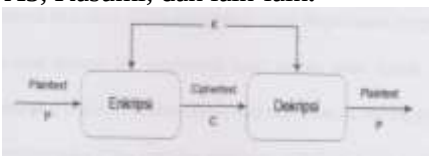
3.2 Jenis-Jenis Algorita Kriptografi

Terdapat dua jenis algorita kriptografi berdasarkan jenis kuncinya, yaitu:

1. Algorita simetri (Konvensional)
2. Algorita Asimetri (Kunci publik)

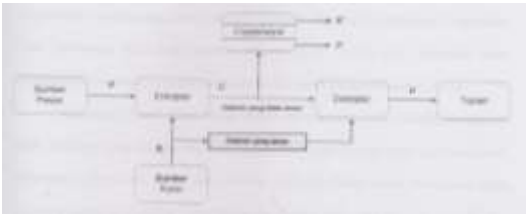
Algoritma simetri

Algoritma simetri disebut juga sebagai algoritma konvensional. Adalah algoritma yang menggunakan kunci enkripsi yang sama dengan kunci deskripsinya. Disebut konvensional karena algoritma yang biasa digunakan orang sejak berabad-abad yang lalu adalah algoritma jenis ini. Algoritma simetrik sering juga disebut sebagai algoritma kunci rahasia, algoritma kunci tunggal, atau algoritma satu kunci, dan mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu sebelum mereka dapat berkomunikasi dengan aman. Keamanan algoritma simetri tergantung pada kunci, membocorkan kunci berarti bahwa orang lain dapat mengenkrip dan mendekrip pesan. Agar komunikasi tetap aman, kunci harus tetap dirahasiakan. Yang termasuk algoritma kunci simetris adalah OTP (One Time Pad), DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5, Kasumi, dan lain-lain**.**



Gambar 2.1 Kriptografi Konvesional

Gambar 2.1 di atas memperlihatkan kriptografi simetri yang biasa disebut juga sebagai kriptografi kunci konvesional.Pesan plaintext P, misalnya SAYA dikodekan (dienkrip) menjadi ciphertext @#$% menggunakan password (kunci K) TES. Untuk mengembalikan cipher @#$% menjadi SAYA dilakukan proses deskripsi dengan kunci yang sama yaitu TES. Karena kunci yang digunakan sama, maka disebut kriptografi kunci simetri. Dan karena jenis ini telah digunakank selama berabad-abad yang lalu, maka dinamakan pula sebagai kriptografi konvesional.



Gambar 2.2 Lingkaran Kriptografi Konvesional

Gambar 2.2 memperlihatkan lingkungan di mana kriptografi sering digunakan Chipper C dikirimkan ke tujuan melalui saluran yang umumnya tidak aman, misalnya melalui internet. Sedangkan kunci K sendiri harus dikirimkan melalui saluran yang aman. Untuk mengirimkan kunci dengan aman, pengirim dan penerima dapat bertemu dan menyepakati kunci tertentu untuk dipakai bersama dalam komunikasi berikutnya. Dalam saluran yang tidak aman ini, seorang penyerang dapat menyadap chipper C dan kemudian melalukan analisis untuk menemukan nilai P. Nilai K dan P yang merupakan perkiraan

yang dihasilkan oleh penyerang disebut sebagai K' dan P'.

Orang sering menggunakan notasi matematika untuk mempermudah penulisan dan analisis, sehingga kriptografi modern selalu berhubungan dengan matematika. Dengan pesan asal P dank ode rahasia C yang diperoleh dari enkripsi dengan kunci K, maka dapat dituliskan : C = EK (P)

Notasi ini menyatakan bahwa C dihasilkan oleh fungsi enkripsi E yang dioperasikan terhadap masukan P dengan kunci K. Operasi ini dilakukan di pengirim. Pada penerima, dilakukan operasi sebaliknya P = D_k (C). pemecahkode (cryptanalyst) sering kali hanya memiliki C dan harus menemukan nilai P nya.

Algoritma simetri dapt dibagi dalm dua kategori.jenis pertama beroperasi pada plaintext yang berupasatu bit tunggal pada satuu waktu, yaitu disebut stream algorithms (algoritma aliran atau stream ciphers).jenis kedua beroperasi pada plaintext dalam bit-bit ini disebut blok.dan algoritmanya disebut sebagai algoritma blok atau kode rahasia blok. Untuk algoritma computer modern, ukuran blok dasarnya adalah 64 bit atau 128 bit, cukup besar untuk menghindari analisis pemecehan kode dan cukup kecil agar dapat bekerja dengan cepat.sebelum pemakaian computer,algoritma biasanya beroperasi padaplaintext,satu karakter per satu operasi.maka dapat dikatakan bahwa seperti algoritma aliran yang beroperasi pada aliran karakter.

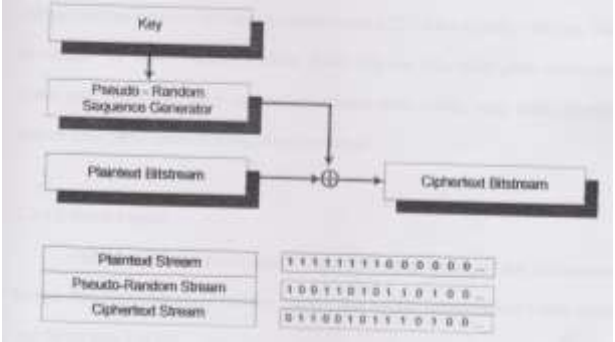
3.2.1.1 stream cipher

Stream ciphers dapat dibuat sangat cepat, jauh lebih cepat dibandingkan dengan algoritma block cipher secara umum unit plaintext yang 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,tranformasi dari unit plaintext yang lebih kecil ini berbeda antara satu dengan lainnya, tergantung pada kapan unit tersebut ditemukan selama proses ekripsi.

Satu stream cipher menghasilkan apa yang disebut suatu keystram (satu barisan bit yang digunakan sebagai kunci). Proses ekripsi dicapai dengan menggabungkan keystream dengan plaintext biasanya dengan operasi bitwise **xor.**pembentukan keystream dapat dibuat independen terhadap plaintext dan ciphertext, menghasilkan apa yang disebut dengan synchronous stream cipher, atau dapat dibuat tergantung pada data dan ekripsinya, dalam hal mana stream cipher disebut sebagai self-synchronizing. Kebanyakan bentk stream cipher adalah synchronous stream ciphers**.** 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 (random). Keystream memiliki panjang sam dengan pesan plaintext dan string acakrandom)digabungkan dengan menggunakanbitwise **xor** dengan plaintext untuk menghasilkan ciphertext. Karena keystream seluruhnya adalh random, 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 secrey), 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 severe. Sedangkan keamanan sempurna,one-time pad secara umum adalah tidak praktis.

Stream ciphers dikembangkan sebagai satu perkiraan 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 standar secara de facto. Metode stream cipher yang umum digunakan adalah **rc4.** Satu halmenarik bahwa metodeoperasi tertentu dari block cipher dapat mentransformasikan secara efektif hasil operasi tersebut kedalam satu keystream generator dan dalam hal ini, block cipher apa saja digunakan sebagai satu stream cipher, seperti dalam **des, cfb** atau **ofb.** Akan tetapi, stream ciphers dengan desain khsus biasanya jauh lebih cepat.



Gambar 2.3 Stream Cipher

Gambar 2.3 terlihat stream cipher. Key merupakan kunci induk yang untuk membangkitkan aliran kunci acak semu (yang dibangkitkan dari pseudo-random sequence generator). Kunci acak semu ini di XOR-kan dengan plaintext untuk menghasilkan ciphertext.

Stream cipher rawan terhadap attack pembalikan bit. Misalkan terdapat transfer antar rekening di sebuah bank sejumlah 10 USD untuk sebuah transaksi. Plaintext QT-TRNSFER USD $000010 FRM ACCNT 12345-67 to ciphertext aMz0raplixMipun7uxorilm42zuweem0qapii7wept anxil

00101101 Fiplow bit

00101100 Ciphertext amz0raplixmpipun7txorilm42zuweem0qap117wep tanxil

Plaintext QT-TRNSFER USD $ 100010,00 FRM ACCNT 12345-67 TO

Ditengah jalan, bit ‘’I” dari kerakter “U” diganti menjadi “O” yang mengakibatkan kerakter “U” akan berubah menjadi “T” dan setelah didekripsi ulang di tujuan,”10 dolar” menjadi 100010 dolar”. Di sini kita tidak perlu mengetahui kunci yang digunakan untuk melakukan enkripsi sama sekali, yang harus diketahui adalah letak pesan-pesan tertentu yang kita minati.

3.2.1.3 Block Cipher

Algoritma block cipher adalah algoritma yang masukan dan keluarannya berupa satu block,dan setiap blocknya terdiri dari banyak bit (maksimal 1 blok terdiri dari 64 bit atau 128 bit). Bentuk algoritmaenkripsi kunci simetri block cipher mentrasformasikan satu blok data tertentu dari plaintext (unencrypted text) ke dalam satu blok data ciphertext (encrypted text)dengan panjang yang sama.transformasikan ini berlangsung melalui penggunaan kunci rahasia yang disediakan oleh user. Dekripsi dilakukan dengan menggunakan transformasi sebaliknya terhadap blok ciphertext dengan kunci yang sama. Panjang blok tertentu disebut ukuran blok (block size),dan untuk sejumlah block cipher,ukuran blok adalah 64-bit dalam beberapa tahun ke depan, ukuran blok akan meningkat menjadi 128-bit sesuai dengan perkembangan kemampuan mikroprosesor. 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.

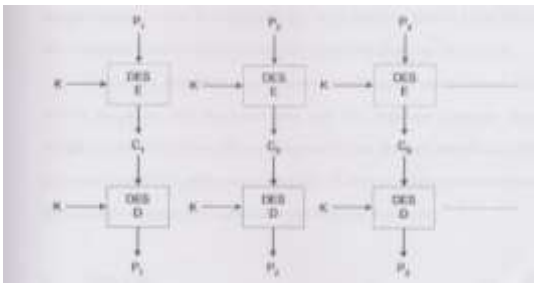
Semua algoritma kriptografi konvensional dapat digunakan pada metode operasi ini. Mode-mode digunakan dengan tujuan untuk megatasi keamanan cara penyandian dan juga untuk mempermudah penyadian. Mode-mode lain masih sangat beraneka ragam jenisnya. Namun keempat mode yang disebutkan di bawah ini merupakan dasar darimode lainnya dan sangat sering digunakan.

1. Mode ecb (electronic codebook)
2. Mode cbc (cipher block chaining)
3. Mode cfb (cipher feed back)
4. Mode ofb (output feed back)

Pada setiap mode, pesan plaintext (P) yang panjang di pecah menjadi satuan unit data disebut blok.misalkan saja panjang setiap blok 64 bit. Jika terdapat blok yang panjangnya kurang dari 64 bit, maka blok tersebut terlebih dahulu harus ditambah dengan bit padding (tambahan) agar jumlah totalnya mencapai 64 bit. Padding dapat dilakukan dengan penambahan bit “I” yang diikuti bit “0” hingga mencapai panjang yang diinginkan.

1. MODE ECD

   Pada mode ini, setiap plaintext dienkripsi (dengan algoritma DES,ASE atau algoritma blok cipher lainnya) menjadi satu blok cipher tanpa mempengaruhi blok pesan yang lain. Satu blok terdiri dari 64 bit atau 128 bit bagian dari pesan.untuk lebih jelasnya perhatikan gambar berikut ini:



GAMBAR 2.4 MODE Enkripsi dan dekripsi ECB

Pada gambar 2.4 bagian atas.algoritma DES (atau blok cipher lainnya) mengenkripsi pesan dengan mode ECB. Sementara pada bagian bawah,DES (atau blok cipher lainnya) digunakan untuk melakukan dekripsi.mode ini merupakan cara lain, sehingga jika penerima mendapatkan satu blok rusak,dia hanya

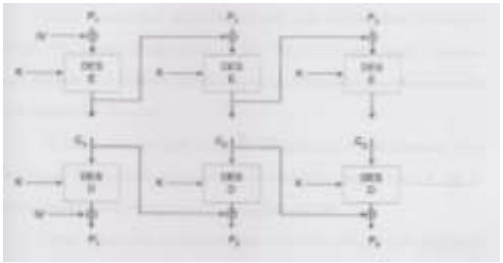
perlu meminta blok yang rusak tersebut,sehingga pengiriman dapat cepat. Dekripsi juga tidak perlu menunggu seluruh pesan sampai terlebih dahulu,sehingga dekripsi dapat dilakukan pada saat bersamaan dengan penerimaan.

Sifat dasa yang penting dari ECB adalah bahwa blok plaintext yang sama akan dikodekan menjadi cipher yang sama. Jika hal ini sering berlangsung,pihak lawan dapat membuat perkuraan apa kira-kira isi plaintext –nya karena seseorang cenderung membuat pesan yang mempunyai keteraturan. Misalnya dalam suatu surat, tanggal biasanya terletak di sebelah atas kiri, nama penulis di sebelah kanan bawah dan sebagainya. Semakin struktur, semakin rawan pula mode enkripsi jenis ini.

Bahaya lain yang dapat munculmisalnya seseorang mengetahui bahwa blok tertentu dari aliran data merupakan data gaji. Jika seseorang karyawan dapat mengakses blok ini misalnya dengan pencegatan di jalan, dia dapat menukarkan blok gaji atasan dengan blok gajinya tanpa harus tahu isi plaintext yang sebenarnya karena dia yakin bahwa gajinya tentu kalah besar disbanding gaji atasan.

2. MODE CBC

Pada mode ini,plaintext yang sama akan dienkripsi ke dalam cipher yang berbeda, karena blok cipher tidak hanya bergantung pada blok plaintext yang berhubungan, melainkan bergantung pada cipher yang sebelumnya.untuk lebih jelasnya perhatikan gambar 2.5 berikut ini.



GAMBAR 2.5 MODE ENKRIPSI DAN DEKRIPSI CBC

Pada mode ini, masukan (untuk dienkripsi) merupakan hasil XOR antara plaintext sekarang dengan cipher sebelumnya. Kunci K digunakan pada setiap blok. Pola plaintext yang berulang akan tertutup operasi XOR sehingga lawan lebih sulit menganalisis kemungkinan plaintext-nya.

Ketika dekripsi, setiap blok cipher dilakukan algoritma dekripsi. Hasilnya di-XOR-kan dengan blok cipher sebelumnya untuk menghasilkan blok plaintext. Untuk membuktikan bahwa hal ini bekerja, dapat dilihat bentuk formal dari kenyataan di atas:

Cn=EK[Cn-1 pn]----------
----------(enkripsi)

Kemudian pada proses dekripsi
Dk[CN=DK[EK[CN-1 PN]]
DK[CN-1 PN
CN-1 DK[CN]=CN-1 CN-1 PN=PN

Untuk menghasilkan blok cipher pertama pada enkripsi,suatu intilization vector (IV) digunakan untuk menggantikan cipher yang sebelumnya. Sebaliknya pada dekripsi,IV di-XOR-kan dengan keluaran algoritma dekripsi untuk keamanan yang maksimum,IVseharusnya diproktesi sebagaimana halnya dengan kunci K. hal ini dapat dilakukan dengan pengiriman IVmenggunakan ECB.

Untuk menghindari kerugian IV pada CBC serta kerugian ECD yang dapat menimbulkan pola cipher berulang, dapat digunakan pola ECB dengan blok yang besar misalnya 64 byte dan bukannya 64bit.

3. MODE CFB

Secara matematis, CFB dapat dinyatakan sebagai berikut:

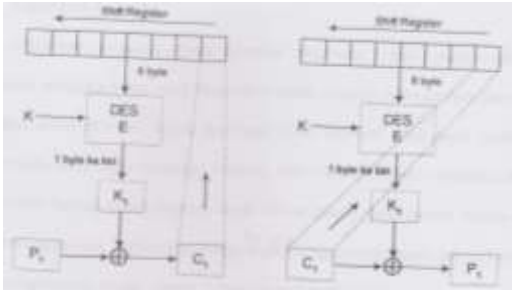
Ci=pi EK(Ci-I)
Pi=Ci EK(Ci-1)

Motode ini digunakan untuk mengenkripsi aliran cipher.dengan cara ini tidak diperlukan lagi padding karena jumlah bit data tidak harus merupakan kelipatan blok minuman. Mode ini adalah bahwa panjang cipher akan tepat dengan panjang plaintext.

Pada mode ini,input di proses 8 bit setiap kali enkripsi dilakukan. Ciphertext sebelumnya digunakan sebagai bagian input dari algoritma enkripsi untuk menghasilkan keluaran acak. Keluaran ini diambil 8 bit paling kirinya untuk di XOR-kan dengan plaintext sepanjang 8 bit untuk menghasilkan ciphertext berikutnya.

Input enkripsi terdiri dari input yang lama digeser ke kiri sejauh 8 bit. Kekosongan sebanyak 8 bit ini akan diisi oleh ciphertext sebelumnya. Input enkripsi mula-mula adalah initialization vector (IV) misalkan saja panjang (IV) ADALAH 64 bit berada pada register geser. Keluaran enkripsi misalkan saja juga 64 bit.

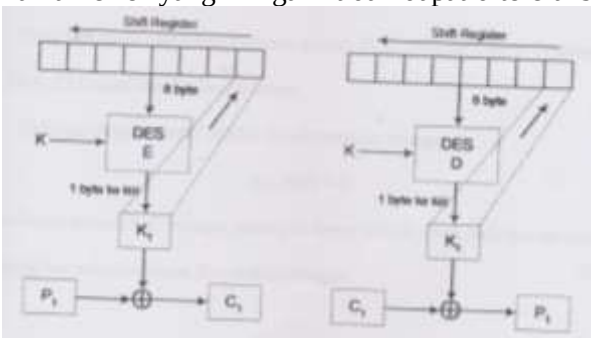
Salah satu kerugian mode CFB adalah perambatan kesalahan. Jika salah satu blok cipher mengalami kesalahan ketika di saluran, maka blok-blok berikutnya akan terpengaruh. IV juga harus unik untuk setiap pesan, sebab bila IV sama untuk pesan yang berbeda, maka keluaran Ki akan juga sama untuk plaintext yang berbeda akibatnya seperti pada system OTP,dimana bila kunci yang sama digunakan untuk mengenkripsi pesan yang berbeda maka kunci akan mudah ditemukan. Bila IV nya terpaksa harus sama,maka kunci enkripsi K yang harus berbeda untuk setiap pesan agar diperoleh Ki yang berbeda.



Gambar 2.6 Operasi Enkripsi dan Deskripsi CFB

4. MODE OFB

Mode OFB tidak mengalirkan error. Satu bit error pada ciphertext hanya mempengaruhi satu bit plaintext pada prosesdekripsi. Ini berguna untuk sistem analog yang didigitasi seperti voice atau video, di mana error satu bit dapat ditoleransi, namun error yang mrngalir tidak dapat ditoleransi.



Gambar 2.7 Operasi Enkripsi Dan Dekripsi Ofb

3.2.2 Algoritma Asimetri

Algoritma asimetrik (juga disebut algoritma kunci public) didesain sedemikian sehingga kunci yang digunakan untuk enkripsi berbeda dari kunci yang digunakan untuk dekripsi. Lebih jauh lagi, kunci dekripsi tidak dapat (sedikitnya dalam waktu yang dapat diterima) di hitung dari kunci enkripsi. Algoritma disebut kunci public karena kunci enkripsi dapat dibuat publik yang berarti semua orang boleh mengetahuinya.sembarang orang dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi pesan, namun hanya orang yang tertentu (calon penerima pesan dan sekaligus pemilik kunci dekripsi yang merupakan pasangan kunci public) yang dapat melakukan dekripsi terhadap pesan tersebut. Dalam system ini, kunci enkripsi sering disebut kunci public, sementara kunci dekripsi sering disebut kunci privat. Kunci privat kadang-kadang disebut kunci rahasia. Istilah kunci rahasia pada algoritma simetri digunakan untuk menyatakan kunci enkripsi dan sekaligus kunci dekripsi, sementara pada algoritma asimetri digunakan untuk menyatak kunci privat, karena kunci public tidak dirahasiakan.

Yang termasuk algoritma asimetriadalah ECC (ELLIPTIC CURVE CRYPTOSYSTEM), LUC,RSA,EL Gamal dan Diffie-hellman.

Enkripsi dengan kunci public ke dinyatakan sebagai:

Eke(M)=C

Dengan kunci privat (Kd)sebagai pasangan kunci publik (Ke),dekripsi dengan kunci privat yang bersesuaian dapat dinyatakan dengan:

DKd (C)=M

Di sini ke merupakan pasangan Kd.artinya tidak ada Kd lain yang dapat digunakan untuk melakukan dekripsi kode C yang merupakan hasil enkripsi dengan kunci Ke.sebaliknya, pesan dapat dienkrip dengan kunci privat dan didekrip dengan

kunci publik. Metode ini digunakan pada tanda tangan digital. Meskipun aga kmembingungkan,operasi ini dapat dinyatakan sebagai:

Ekd (M)=C
DKE (C)=M

Artinya kunci privat dank unci publik dapat digunakan secara berlawanan dengan tujuan yang berbeda.sifat ini hanya berlaku untuk algoritma kunci public tertentu sejenis RSA sifat ini tidak berlaku pada algoritma Diffi-hellman.

2.3 landasan matematika kriptografi

2.3.1 ADD

ADD merupakan operasi penjumlahan yang dilambangkan dengan tanda (+) jika terjadi carry maka akan dilakukan padding pada bit selanjutnya.contoh operasi penjumlahan sebagai berikut:

0110 1000 0000 1001
0001 1000 0110 0001
+
1000 0000 0110 1010

2.3.2 SUBSTRACT

SUBSTRACT merupakan operasi pengurangan yang dilambangkan dengan tanda (-). Jika pengurangan yang akan dikurangkan terlalu kecil dibandingkan bilangan pengurangan maka akan dilakukan carry dari bit berikutnya contoh operasi pengurangan sebagai berikut:

1110 1000 1001 1001
0001 1000 **0110 0001**
-
1101 0000 0011 1000

2.3.3 XOR

XOR adalah operasi exclusive-OR yang dilambangkan dengan "+" standar dalam operasi dalam bit: 0+ 0=0,0+1=1,1+0=1,1+1=0.

Karena di XOR dengan nilai yang sama dua kali maka akan mengembalikan nilai asli, sehingga XOR cenderung dipakai dalam proses enkripsi dan dekripsi yang memiliki algoritma yang sama.

P +K=C
C + K=p

2.3.4 SHIFT

Shift bit merupakan operasi terhadap bit dengan membuang suatu barisan bit sebanyak yang diinginkan. Bit yang telah dibuang akan hilang bit sisanya akan digeser dan bit yang dibuang tersebut diganti dengan bit "0" operasi shift bit terbagi atas:

a. Operasi shift kiri (left shift). Membuang barisan bit dari kiri sebanyak nilai yang diberikan secara per bit, kemudian bit sisanya di geser kekiri dan tambahkan bit "0" di bagian kanan sebanyak nilai yang diberikan. Lambing operasi ini adalah<<. Berikut ini adalah contoh operasi left shift.
   01111111<<1:11111110
   01111111<<2:11111100
b. Operasi shift kanan (right shift). Membuang barisan bit dari kanan sebanyak nilai yang diberikan secara per bit, kemudian bit sisanya di geser ke kanan dan tambahkan bit "0" di bagian kiri sebanyak nilai yang diberikan.lambang operasi ini adalah >>.
   Berikut ini adalah contoh operasi right shift.
   01111111>>1:0011111
   0111111>>2:00011111

2.3.5 ROTATE

Rotate merupakan operasi terhadap bit dengan memutar suatu barisan bit sebanyak yang diinginkan.bit yang telah tergeser tidak akan hilang karena bit tersebut akan dipindahkan ke sisi barisan bit yang berlawanan dengan arah puutaran bit.rotasi bit terbagi atas:

1. Operasi rotasi kiri (rotate left), memutar barisan bit kekiri sebanyak nilai yang diberikan secara per bit, kemudian bit kosong yang telah bergeser di sebelah kananya akan digantikan dengan bit yang telah tergeser di sebelah kirinya.Operasi rotate left dilambangkan dengan "<,,"berikut ini adalah contoh operasi rotate left:
   0111111<<,1;1111110
   0111111<<,2:1111101
2. Operasi rotasi kanan 9rotate right), memutar barisan bit ke kanan sebanyak nilai yang diberikan secara per bit, kemudian bit yang kosong yang telah telah tergeser di sebelah kirinya akan digantikan dengan bit yang telah tergeser di sebelah kananya. Operasi rotate rigt dilambangkan dengan >>>".berikut ini adalah contoh operasi rotate right:
   01111111>>>1;10111111
   011111111>>>2:11011111

2.3.6 BITWISE AND

Bitwise and merupakan operasi bit yang membandingkan bit input pertama dengan bit input kedua hasil dari perbandingan ini dapat dilihat pada table 2.2berikut ini.

TABEL 2.2 bitwise AND

Input 1	Input 2	output
0	0	0
0	1	0
1	0	0

1	1	1

Contoh 10101111 AND 1000001`= 10000001

2.4 Algoritma Stream Cipher Rc4

RC4 adalah jenis kriptografi symmetric key, secret key,dan stream cipher yang didesain oleh ron rivest.RC merupakan singkatan dari "RON'sCODE".dikenal secara umum sebagai block cipher RC2 dan RC5,dan blok cipher RC6 yang di desain oleh Ron Rivest bersama orang lain.RC4 didesain sekitar tahun 1990-an.

RC4 adalah ini sial rahasia perdagangan.tetapi pada September 1994 implementasi RC4 dipublikasikan tanpa nama pada cyherpunks mailing list, berita tersebut dengan menyebar pada usenet pad asci.crypt nwsgroup dan pada beberapa sile di internet. Karena algoritmanya telah diketahui maka RC4 tidak lagi merupakan rahasia perdagangan. Namun dari RC4 adalah merupakan sebuah merek dengan RC4 sering disebut sebagai "ARCFOUR"untuk menghindari kemungkinan dalam masalah merek dagang. RC4 merupakan bagian dariprotokol standar enkripsi yang umum digunakan termasuk dalam SSL (security socket layer) yang dipakai untuk mengamankan jaringan web brouser.

RC4 diinilisasi dari sebuah kunci rahasia.kemudian di generatesebuah "keystream"yang disederhanakan dengan XOR dengan plaintext untuk menghasilkan ciphertext.proses dekripsi sama dengan proses enkripsi. Salah satu alasan untuk kepopuleran RC4 adalah kesederhanaannya. Algoritma RC4 dapat diingat dan mudah diiplementasikan.algoritma RC4 menggunakan 256 byte dari memori,S[0]hingga S[255], dan menggunakan variable integer i,J, dan k.

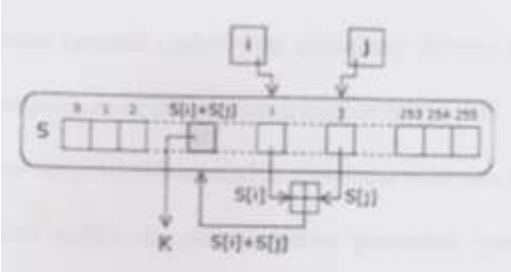
RC4 adalah salah satu cipher yang tercepat yang dipergunakan secara luas untuk pekerjaan yang serius.cryptanalisis dari RC berada pada tahap yang tidak tentu. Secara teoritis RC4 dapat dipecahkan jika stream ciphertext yang dihasilkan berjumlah giga byte.tetapi ini tidak merupakan masalah utama dalam penerapanya pada tahun 2001 sebuah penemuan baruh yang mengejutkan terjadi: semua kemungkinan RC4,statistic dari byte pertama dari output keystream tidak dalam keadaan acak. Hal ini berefek ketika digunakan untuk memecahkan enkripsi WEP (WIRED EQUIVALENT PRIVACE) yang dipakai pada 802.11 jaringan tanpa kabel. WEP menerapkan RC4 dengan banyak kunci yang mirip,sehingga memungkinkan jaringan ini terbuuka untuk diserang.implementasi RC4 saat ini sering mengabaikan 256 byte pertama atau lebih dari stream untuk mengatasi masalah tersebut.

Seperti halnya dengan cipher stream,RC4 mudah dipecahkan jika kunci yang sama dipakai dua kali. Masalah ini biasanya di atasi dengan melakukan hashing kunci dengan vector inisialisasi unik (unique initialization vector) setiap kali kunci in ini dipakai dan vector inisialisasi ini dikirim bersamaan dengan pesan.

RC4 merupakan stream cipher yang sangat cepat dana man dari RSA data security, inc.RC4 digunkan dalam lingkungan dengan sumber daya yang kecil dengan resiko yang tinggi.RC4 tidak dipatenkan meskipun untuk tujuan komersial sekalipun. RC4 menggunakan panjang kunci variable dari 1-256 byte (mempunyai kemampuan antara 1q-2048 bit)untuk menginilisiasi 256-byte state table.algoritma RC4 dibagi menjadi dua tahap,yaitu membentuk kunci dan ciphering (enkripsi/dekripsi).pembentukan kunci merupakan tahap pertama dan tersulit.

Selama pembentukan kunci,kunci enkrepsi digunakan untuk menghasilkan sebuah variable enkrip menggunakan 2 array (state arry&key arry)dan sejumlah operasi penjumlahan.berikut ini penjelasan algoritma taha-tahap dalam proses ciphering dari RC4:

1. pembentuukan kunci (key scheduling)
   for i = 0 …. 255
   next i
   j = 0
   for i = 0 …. 255
   j = (j + S[i] + key [I mod key_length]) mod 256
   swap (S[i],S[j])
   next i
2. **proses enkripsi**
   for I = 0 to len (plaintext – I)
   i = (i + 1) mod 256
   j = (j + S[i],S[j]) mod 256
   swap (S[i] , S [j])
   k = S [(S[i] + S [j]) mod 256]
   ciphertext (i) = k XOR plaintext (i)
   next i
3. **proses dekripsi**
   for i = 0 to len (ciphertext – 1)
   i =(I + 1) mod 256
   j = (j + S [i]) mod 256
   swap (S [i], S [j])
   k = S [(S[i] + S [j]) mod 256]
   plaintext (i) = k XOR ciphertext (i)
   next i



Gambar 2.8 Lookup Strage RC4

2.5 Algoritma Stream Cipher Sapphire Ii

Stream cipher sapphire sangat mirip dengan versi sapphire sebelumnya sebagaimana merupakan hasil kerja awal dari Michel paul Johnson pada bulan November 1993. Stream cipher ini juga mirip pada beberapa bagian dengan RC4 yang di-post ke sci.crypt. keduanya beroperasi dengan memutasi vector permutasi. RC4 tidak menyertakan prinsip feedback pada ciphertext atau plaintext. Ini membuatnya lebih lamah terhadap serangan yang dikenal sebagai known plain text attack, tetapi dalam beberapa hal RC4 lebih cepat dibandingkan dengan sapphire.

Stream cipher sapphire digunakan dalam produk shareware seperti Quicrypt, dimana tersedia pada ftp:// ftp.csn.net/mpj/qcrypt10.zip dan pada Colorado Catacombs BBS (303-772-1062). Terdapat dua versi dari Quicrypt:versi ekspor (dimana knci sesi dibatasi hingga 32 bit tetapi dengan penambahan kunci user) dan versi komersial yaitu versi north American (menggunakan kunci sesi 128 bit). Suatu variant dari stream Cipher Sapphire juga digunakan dalam program shreware atbash, dimana tidak mempunyai versi ekspor yang lemah.

Stream CIPHER SAPPHIRE II merupakan modifikasi dari stream cipher sapphire yang dirancang lebih tahan terhadap adaptive chose plaintext attacks (dengan pengorganisasi kembali cipher yang diizinkan).stream Cipher II digunakan dalam program utility enkripsi yang disebut ATBASH2.

Stream cipher sapphire II berdasarkan atas suatu state machine.state tersebut terdiri atas lima nilai indeks dan sebuah vector permutasi dipindahkan pada posisi baru (dimana mungkin sama seperti lokasi lama)untuk setiap output byte.output byte merupakan fungsi nonlinear dari semua lima nilai indeks dari delapan dari byte dalam vector permutasi, hingga menhasilkan suatu usaha yang sulit untuk memecahkan variable state pada output sebelumnya. Pada proses inisialisasi, vector permutasi (disebut dengan cards array dalam source code yang diterbitkan oleh Michael paul Johnson)diacak berdasarkan pada kunci user.proses pengacakan ini dilakukan dengan suatu cara yang dirancang untuk memperkecil bias dalam hasil byte pada array. Keintungan terbatas dalam metode ini adalah tidak melakukan eliminasi bias, tetapi memperlambat proses untuk membuat brute force attack lebih lama dan lebih mahal.pengrangan bias (relative seperti pada RC4)lebih mudah,tetapi keuntungan ini mungkin mempunyai nilai kriptografik yang kecil.variabel indeks diset pada elemen vector permutasi pada lokasi 1,3,5,7, dan suatu nilai kunci dependent (rsum) dipindahkan kekiri melalui proses pengacakan dari vector permutasi (cards array).

Stream cipher SAPPHIRE II dirancang mempunyai beberapa property seperti berikut:

1. digunakan untuk mengenerasi nilai cek kriptografi dan untuk mempriteksi pesan.
2. Menerima variable length key.
3. Cukup kuat untuk mengatur sekurang-kurang kunci 64 bit untuk keseimbangan sekuritas.
4. Cukup kecil untuk dibangun ke dalam aplikasi yang lain dengan beberapa kunci aktif.
5. Key setup cukup cepat untuk mendukung operasi perubahan kunci tetapi lambat saat dilakukan brute force attack pada kunci.
6. Cukup cepat tetapi tidak secara signifikan berdampak langsung pada operasi pembacaan dan penulisan file pada platform yang paling baru.
7. Portable pada computer yang umum dan efisien pada Bahasa C,C++, dan pascal,
8. Bersifat byte oriented.
9. Menyertakan ciphertext dan plaintext feedback (untuk menyembunyikan data lebih optimal dan nilai dalam pembentukan nilai cek kriptografik).
10. Unjuk kerja yang dapat diterima sebagai generator bilangan acak murni tanpa menyediakan suatu data stream untuk proses enkripsi atau dekripsi.
11. Mengizinkan suatu pembatasan kunci untuk dipakai kembali tanpa adanya degredasi keamanan yang serius.

2.5.1 Key Setup

Key setup (diilustrasikan dengan fungsi initialize () pada bagian source code) terdiri atas tiga bagian:

1. Menginisialisasi variable indeks
2. Menset vector permutasi untuk suatu known state (suatu urutan perhitungan)
3. Dimulai dengan bagian akhir dari vector,mempertukarkan setiap elemen dari

vector permutasi dengan suatu elemen yang diindeks pada suatu lokasi dari 0 hingga indeks saat ini (dipilih oleh fungsi keyrand())

Maksimum berdasarkan atas kunci user,state saat vector permutasi,dan indeks yang sedang dijumlahkan disebut dengan rsum.sebagai catatan panjang kunci yang digunakan dalam keyrand(),seperti suatu kunci "abcd"tidak akan menghasilkan permutasi yang dengan kunci seperti "abcd".

2.5.2 enkripsi sapphire II

Setiap proses enkripsi melibatkan proses updating nilai indeks,berkisar pada empat byte dalam vector permutasi,kemudian memilih sebuah byte output,dan penambahan byte output dengan bitwise modulo-2 (exclusive-or)pada byte plaintext untk menghasilkan byte ciphertext.nilai indeks ditambahkan dengan aturan yang berbeda.indeks disebut dengan rotor hanya menambahkan bertambah satu (modulo 256) setiap waktu. Peningkatan oleh nilai pada vector permutasi dilakukan oleh rotor.peningkatan avalance oleh nilai dalam vector permutasi oleh byte yang lain dalam vector permutasi dilakukan oleh byte ciphertext terakhir.byte plaintext terakhir dan byte ciphertext terakhir juga disimpan sebagai variable indeks.agar lebih jelas maka perhatikan fungsi ecrypt()pada bagian fragmen source code.

2.5.3 DEKRIPSI SAPPHIRE II

Proses dekripsi mempunyai prinsip yang sama dengan proses dekripsi,kecuali pada proses swapping dari pengisian nilai pada plaintext terakhir dan ciphertext terakhir dan pengembalian nilainya.lihat pada bagian fungsi decrypt () pada bagian fragmen source code pada bagian lampiran.

2.6 persamaan dan perbedaan sapphire II dengan RC4

sapphire II dengan RC4 mempunyai kesamaan dalam hal jumlah fungsi yang maksimum yang dapat dipakai yaitu 256 karakter atau 256 byte (2048 bit).dalam hal algoritmanya secara geris besar kedua algoritma ini mempunyai kesamaan yaitu pertama kali kunci yang diberikan digunakan untuk mengacak suatu variable yaitu pada sapphire II digunakan untuk mengacak variable CARDS sedangkan pada RC4 untuk mengacak sbox.kedua algoritma ini menggunakan XOR

BAB III
PEMBAHASAN DAN PERANCANGAM

3.1 Pembahasan

Pada bagian pembahasan ini penulis akan menjelaskan secara umum bagaimana cara kerja dari algoritma kriptografi Sapphire II dan RC4 dengan memberikan contoh kasus untuk proses enkripsi dan dekripsi Pada bagian pembahasan ini penulis akan menjelaskan secara umum model enkripsi dan dekripsi dari algoritma stream cipher Sapphire II dan RC4. Gambar 3.1 berikut ini merupakan bentuk model dari algoritma stream cipher Sapphire II dan RC4.

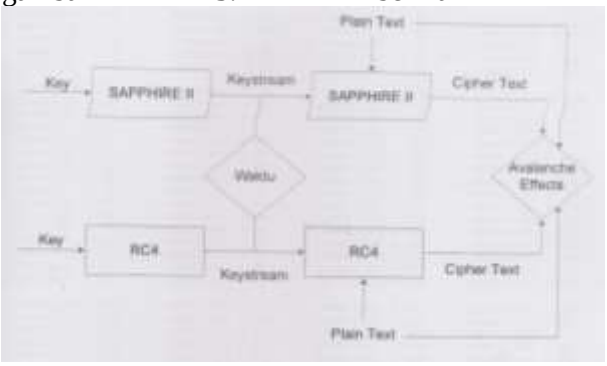
Dari skema umum proses enkripsi dan dekripsi Sapphire II dan RC4 terlihat bahwa kedua algoritma stream cipher ini merupakan jenis keiftografi kunci privat dimana kubci yang sama digunakan kembali baik untuk proses enkripsi dan proses dekripsi. Secara umum kedua stream cipher merupakan jenis stream cipher yang dapat memproses stream dalam ukuran bit ataupun secara per byte.

Gambar 3.1 Skema Umum Proses Enkripsi dan Dekripsi Sapphire II dan RC4

Secara khusus rangkaian proses algoritma Sapphire II dan RC4 sendiri terdiri atas dua tahapan yaitu tahapan pertama adalah pembentukan keystream yang merupakan Pembentukan sub kunci sedangkan tahap keduanya merupakan taahaap enkripsi dan dekripsi. Pada algoritma sapphire II *keystream* yang dihasilkan akan mengacak array CARDS sedangkan pada RC4 digunakan untuk mengacak nilai pada array Sbox. Seluruh rangkaian tahap tersebut adalah sama dimana pada tahap enkripsi dan dekripsi hanya keystream yang dihasilkan dilakukan proses *xor* dengan *plaintext* dan ciphertext.

Program yang dirancang digunakan untuk membandingkan dua buah algoritma stream Cipher yaitu RC4 dan Sapphire II. Perbandingan yang dilakukan mencakup perbandingan waktu (kecepatan proses algoritmna) serta perbandingan nilai *avalanche effect.* Khusus untuk perbadingan kecepatan algoritma akan dilakukan dengan membandingkan lama proses yang mencakup pembentukan kunci hingga proses enkripsi / dekripsi sedangkan perbandingan nilai *avalanche effects* dilakukan dengan cara membandingkan bit-bit *file* input dengan bit-bit *fileoutput* dari tiap algoritma sesudah diproses.

Agar lebih jelas bagaiman proses perbandingan pada program ini maka dapat dilihat pada diagram perbandingan algoritma seperti terlihat pada gambar 3.2 beriku ini



Gambar 3.2 Diagram perbandingan kedua algoritma

3.1.1 Perhitungan Algoritma Sapphire II

Proses pertamaa dimulai dengan melakukan enkripsi dengan algoritma sapphire II. *String*sampel dalam hal ini adalah Test.txt yang terdiri atas 7 byte dan merupakan plain text yang berisi string "NEMESIS". Untuk algoritma Sapphire II ini pertama sekali dibnetuk 256 buah array yang bernama cards dengan nilai sebagai berikut. Proses ini pada algoritma Sapphire II diseut sebagai proses inisialisasi .

For i = 0 . . 255

Cards[i] = i

Cards[0] = 0 Cards[17] = 17
Cards[34] = 34 Cards[51] = 51
Cards[1] = 1 Cards[18] = 18
Cards[35] = 35 Cards[52] = 52
Cards[2] = 2 Cards[19] = 19
Cards[36] = 36 Cards[53] = 53
Cards[3] = 3 Cards[20] = 20
Cards[37] = 37 Cards[54] = 54
Cards[4] = 4 Cards[21] = 21
Cards[38] = 38 Cards[55] = 55
Cards[5] = 5 Cards[22] = 22
Cards[39] = 39 Cards[56] = 56
Cards[6] = 6 Cards[23] = 23
Cards[40] = 40 Cards[57] = 57
Cards[7] = 7 Cards[24] = 24
Cards[41] = 41 Cards[58] = 58
Cards[8] = 8 Cards[25] = 25
Cards[42] = 42 Cards[59] = 59
Cards[9] = 9 Cards[26] = 26
Cards[43] = 43 Cards[60] = 60
Cards[10] = 10 Cards[27] = 27
Cards[44] = 44 Cards[61] = 61
Cards[11] = 11 Cards[28] = 28
Cards[45] = 45 Cards[62] = 62
Cards[12] = 12 Cards[29] = 29
Cards[46] = 46 Cards[63] = 63
Cards[13] = 13 Cards[30] = 30
Cards[47] = 47 Cards[64] = 64
Cards[14] = 14 Cards[31] = 31
Cards[48] = 48 Cards[65] = 65
Cards[15] = 15 Cards[32] = 32
Cards[49] = 49 Cards[66] = 66
Cards[16] = 16 Cards[33] = 33
Cards[50] = 50 Cards[67] = 67
Cards[68] = 68 Cards[84] = 84
Cards[100] = 100 Cards[116]= 116
Cards[69] = 69 Cards[85] = 85
Cards[101] = 101 Cards[117]= 117
Cards[70] = 70 Cards[86] = 86Cards[102] = 102 Cards[118]= 118
Cards[71] = 71 Cards[87] = 87
Cards[103] = 103 Cards[119]= 119
Cards[72] = 72 Cards[88] = 88
Cards[104] = 104 Cards[120]= 120
Cards[73] = 73 Cards[89] = 89
Cards[105] = 105 Cards[121]= 121
Cards[74] = 74 Cards[90] = 90
Cards[106] = 106 Cards[122]= 122
Cards[75] = 75 Cards[91] = 91
Cards[107] = 107 Cards[123]= 123
Cards[76] = 76 Cards[92] = 92
Cards[108] = 108 Cards[124]= 124
Cards[77] = 77 Cards[93] = 93
Cards[109] = 109 Cards[125]= 125
Cards[78] = 78 Cards[94] = 94
Cards[110] = 110 Cards[126]= 126
Cards[79] = 79 Cards[95] = 95
Cards[111] = 111 Cards[127]= 127
Cards[80] = 80 Cards[96] = 96
Cards[112] = 112 Cards[128]= 128
Cards[81] = 81 Cards[97] = 97
Cards[113] = 113 Cards[129]= 129
Cards[82] = 82 Cards[98] = 98
Cards[114] = 114 Cards[130]= 130
Cards[83] = 83 Cards[99] = 99
Cards[115] = 115 Cards[131]= 131
Cards[132] = 132 Cards[148] = 148
Cards[164]= 164 Cards[180]= 180
Cards[133] = 133 Cards[149] = 149
Cards[165]= 165 Cards[181]= 181
Cards[134] = 134 Cards[150] = 150Cards[166]= 166 Cards[182]= 182
Cards[135] = 135 Cards[151] = 151
Cards[167]= 167 Cards[183]= 183
Cards[136] = 136 Cards[152] = 152
Cards[168]= 168 Cards[184]= 184
Cards[137] = 137 Cards[153] = 153
Cards[169]= 169 Cards[185]= 185
Cards[138] = 138 Cards[154] = 154
Cards[170]= 170 Cards[186]= 186

Cards[139] = 139 Cards[155] = 155 Cards[171]= 171 Cards[187]= 187
Cards[140] = 140 Cards[156] = 156 Cards[172]= 172 Cards[188]= 188
Cards[141] = 141 Cards[157] = 157 Cards[173]= 173 Cards[189]= 189
Cards[142] = 142 Cards[158] = 158 Cards[174]= 174 Cards[190]= 190
Cards[143] = 143 Cards[159] = 159 Cards[175]= 175 Cards[191]= 191
Cards[144] = 144 Cards[160] = 160 Cards[176]= 176 Cards[192]= 192
Cards[145] = 145 Cards[161] = 161 Cards[177]= 177 Cards[193]= 193
Cards[146] = 146 Cards[162] = 162 Cards[178]= 178 Cards[194]= 194
Cards[147] = 147 Cards[163] = 163 Cards[179]= 179 Cards[195]= 195
Cards[196] = 196 Cards[212] = 212Cards[228]= 228Cards[244]= 244
Cards[197] = 197 Cards[213] = 213Cards[229]= 229Cards[245]= 245
Cards[198] = 198 Cards[214] = 214Cards[230]= 230Cards[246]= 246
Cards[199] = 199 Cards[215] = 215 Cards[231]= 231Cards[247]= 247
Cards[200] = 200 Cards[216] = 216 Cards[232]= 232Cards[248]= 248
Cards[201] = 201 Cards[217] = 217 Cards[233]= 233 Cards[249]= 249
Cards[202] = 202 Cards[218] = 218 Cards[234]= 234Cards[250]= 250
Cards[203] = 203 Cards[219] = 219 Cards[235]= 235Cards[251]= 251
Cards[204] = 204 Cards[220] = 220 Cards[236]= 236Cards[252]= 252
Cards[205] = 205 Cards[221] = 221 Cards[237]= 237Cards[253]= 253
Cards[206] = 206 Cards[222] = 222 Cards[238]= 238Cards[254]= 254
Cards[207] = 207 Cards[223] = 223 Cards[239]= 239Cards[255]= 255
Cards[208] = 208 Cards[224] = 224 Cards[240]= 240
Cards[209] = 209 Cards[225] = 225 Cards[241]= 241
Cards[210] = 210 Cards[226] = 226 Cards[242]= 242
Cards[211] = 211 Cards[227] = 227 Cards[243]= 243

Langkah berikut adalah mengacak CARDS berdasarkan key yang diinput.
Dengan adanya 256 buah Sbox berarti Sapphire II dapat mendukung hingga 256 karaktek untuk key-nya . dalam pengujian ini digunakan key dengan panjang 3 karakter . aapun kuncu (key) yang dipakain adalah string “ABC”

```
Key = “ABC”
Key_lenght = 3
Key[0] = 65; key[1] = 66; key[2] = 67 // dalam bentuk decimal
```

Langkah selanjutnya adalah melakukan swapping pada Cards dengan algoritma sebagai berikut :

```
Toswap = 0
Toswap = 1
rsum = 0

For I = 255 To 0 step -1
    Toswap = KeyRand(i, Key, KeySize, arsum, KeyPos)
    Swaptemp = cards (i)
    Cards(i) = Cards (toswap)
    Cards (Toswap) = swaptemp
Next I

Rotor = Cards (1)
retchet = Cards (3)
avalanche = Card (5)
last_plain = Cards (7)
last_Chiper = Cards (rsum)

toswap = 0
swaptemp = 0
rsum = 0
KeyPos = 0
```

Dengan algoritma KeyRand adalah sebagai berikut :

```
rsum = arsum
keyPos = KeyPos
v=0
mask = 1
while (mask , Limit)
    mask = ShiftLeftOne(mask) + 1
    rsum = (Cards(rsum) + Asc(Mid$(Key, KeyPos, 1))) And &HFF
    KeyPos = KeyPos + 1
    If (Keypos >= KeySize) Then
      KeyPos = 1
      Rsum = (rsum + KeySize) And &HFF
    End If
    u = mask and rsum
    v = v + 1
    If (v > 11)Then u = u Mod Limit
    KeyRand
```

Wend

Perhitungannya adalah sebagai berikut.

Toswap = 0
Toswap = 1
rsum = 0

toswap = KeyRand (255, “ABC”, 3, 0, 1)

Pemanggilan fungsi KeyRand

V = 0
Mask = 1

While (mask < Limit) -> 1 < 255
mask = mask << 1 + 1
mask = 3

rsum = ((Cards(rsum) + (Key)) And &HFF `FF = 255
rsum = 65

KeyPos = KeyPos + 1
Keypos = 2
u = mask and resume
u = 3 and 65
u=1
v = v + 1
v = 1

Jika (v > 11) maka u = u Limit

u = 1 Mod 255
u = 1
While (mask < Limit) -> 3 < 255
Mask = mask << 1 + 1
Mask = 7

rsum = ((Cards(rsum) + (Key)) And &HFF
rsum = 131

KeyPos = KeyPos + 1
KeyPos = 3jika KeyPos >= KeySize ->3 >= 3
KeyPos = 1
rsum = rsum (rsum + KeySize) And &HFF
rsum = 134
u = mask and rsum
u = 7 And 134
v = v + 1
v = 2
Jika (v > 11) Maka u = u Mod Limit
u = 6 Mod 255
u = 6
.
.
While (mask < Limit) -> 1 < 2
Mask = mask << 1+ 1
Mask = 3

rsum = (rsum + Key Size) And &HFF
rsum = 117

KeyPos = KeyPos + 1
KeyPos = 3 And 117
v = v + 1
v = 1

Jika (v > 11) Maka u = u Mod Limit
u = 1 Mod 2
u = 1
KeyRand = u
KeyRand = 1
Toswap = 1
Swaptemp = Cards (2)
Cards(i) = cards (toswap)
Cards(2) = Cards (1)
Cards(2) = 7
Cards(Toswap) = swaptemp
Cards(1) = 107
Toswap = KeyRand (1)
Pemanggilan fungsi KeyRand
Retry_limiter = 0

Mask =1
KeyRand = u
KeyRand = 0
Toswap = 0
Swaptemp = Cards(1)
Cards(i) = cards (toswap)
Cards(1) = Cards (0)
Cards(1) = 118
Cards(toswap) = swaptemp
Cards(0) = 107
Toswap = KeyRand (0)
Pemanggilan fungsi KeyRand
Retry_limiter = 0
Mask =1
KeyRand = u
KeyRand = 0

Dan Seterusnya. . .
Toswap = 0
Swaptemp = Cards(0)

Cards(i) = cards (toswap)
Cards(0) = Cards (0)
Cards(0) = 107
Cards(toswap) = swaptemp

Cards(0) = 107

Rotor = Cards (1)
Rotor = 118
ratchet = Cards(3)
ratchet = 1
ratchet = Cards(5)
avalanche = 97

last_plain = Cards (7)
last_plain = 98
last_Cipher = Cards (rsum)
last_Ciper = 95
toswap = 0
swaptemp = 0
rsum = 0
KeyPos = 0

Setelah dilakukan looping dan swapping pada Cards akan didapat hasil variable Cards sebagai berikut:

Cards[0] = 107 Cards[16] = 28 Cards[33] = 4
Cards[1] = 118 Cards[17] = 59 Cards[34] = 47
Cards[2] = 7 Cards[18] = 27 Cards[35] = 10
Cards[3] = 1 Cards[19] = 21 Cards[36] = 20
Cards[4] = 62 Cards[20] = 76 Cards[36] = 143
Cards[5] = 97 Cards[21] = 170 Cards[37] = 22
Cards[6] = 46 Cards[22] = 117 Cards[38] = 125
Cards[7] = 98 Cards[23] = 81 Cards[39] = 9
Cards[8] = 43 Cards[24] = 49 Cards[40] = 24
Cards[9] = 230 Cards[25] = 3 Cards[41] = 108
Cards[10] = 84 Cards[26] = 218 Cards[42] = 53
Cards[11] = 12 Cards[27] = 229 Cards[43] = 52
Cards[12] = 2 Cards[28] = 25 Cards[44] = 77
Cards[13] = 31 Cards[29] = 29 Cards[45] = 30
Cards[14] = 106 Cards[30] = 58 Cards[46] = 96
Cards[15] = 82 Cards[31] = 17 Cards[47] = 215
Cards[48] = 34 Cards[64] = 166 Cards[80] = 121
Cards[49] = 26 Cards[65] = 45 Cards[81] = 19
Cards[50] = 115 Cards[66] = 67 Cards[82] = 126
Cards[51] = 0 Cards[67] = 33 Cards[83] = 73
Cards[52] = 56 Cards[68] = 124 Cards[84] = 93
Cards[53] = 119 Cards[69] = 228 Cards[85] = 71
Cards[54] = 112 Cards[70] = 111 Cards[86] = 226
Cards[55] = 110 Cards[71] = 23 Cards[87] = 60
Cards[56] = 41 Cards[72] = 38 Cards[88] = 65
Cards[57] = 48 Cards[73] = 72 Cards[89] = 85
Cards[58] = 13 Cards[74] = 11 Cards[90] = 32
Cards[59] = 224 Cards[75] = 66 Cards[91] = 83
Cards[60] = 39 Cards[76] = 79 Cards[92] = 105
Cards[61] = 51 Cards[77] = 70 Cards[93] = 80
Cards[62] = 36 Cards[78] = 103 Cards[94] = 15
Cards[63] = 167 Cards[79] = 74 Cards[95] = 44
Cards[96] = 120 Cards[112] = 78 Cards[128] = 88
Cards[97] = 16 Cards[113] = 114 Cards[129] = 185
Cards[98] = 8 Cards[114] = 128 Cards[130] = 90
Cards[99] = 116 Cards[115] = 165 Cards[131] = 251
Cards[100] = 187 Cards[116] = 87 Cards[132] = 192
Cards[101] = 18 Cards[117] = 95 Cards[133] = 141
Cards[102] = 227 Cards[118] = 61 Cards[134] = 138
Cards[103] = 232 Cards[119] = 6 Cards[135] = 139
Cards[104] = 69 Cards[120] = 219 Cards[136] = 140
Cards[105] = 64 Cards[121] = 5 Cards[137] = 89
Cards[106] = 57 Cards[122] = 92 Cards[138] = 42

Cards[107] = 159 Cards[123] = 109 Cards[139] = 135
Cards[108] = 225 Cards[124] = 68 Cards[140] = 164
Cards[109] = 54 Cards[125] = 130 Cards[141] = 137
Cards[110] = 113 Cards[126] = 86 Cards[142] = 134
Cards[111] = 123 Cards[127] = 122 Cards[143] = 127
Cards[144] = 196 Cards[160] = 180 Cards[176] = 172
Cards[145] = 169 Cards[161] = 133 Cards[177] = 173
Cards[146] = 234 Cards[162] = 198 Cards[178] = 182
Cards[147] = 147 Cards[163] = 35 Cards[179] = 175
Cards[148] = 236 Cards[164] = 136 Cards[180] = 184
Cards[149] = 37 Cards[165] = 189 Cards[181] = 177
Cards[150] = 238 Cards[166] = 142 Cards[182] = 178
Cards[151] = 231 Cards[167] = 63 Cards[183] = 179
Cards[152] = 240 Cards[168] = 200 Cards[184] = 188
Cards[153] = 233 Cards[169] = 161 Cards[185] = 181
Cards[154] = 50 Cards[170] = 14 Cards[186] = 214
Cards[155] = 235 Cards[171] = 163 Cards[187] = 183
Cards[156] = 40 Cards[172] = 176 Cards[188] = 160
Cards[157] = 237 Cards[173] = 217 Cards[189] = 129
Cards[158] = 202 Cards[174] = 178 Cards[190] = 190
Cards[159] = 75 Cards[175] = 223 Cards[191] = 191
Cards[192] = 132 Cards[208] = 252 Cards[224] = 100
Cards[193] = 193 Cards[209] = 249 Cards[225] = 221
Cards[194] = 194 Cards[210] = 254 Cards[226] = 222
Cards[195] = 195 Cards[211] = 131 Cards[227] = 99
Cards[196] = 144 Cards[212] = 152 Cards[228] = 104
Cards[197] = 197 Cards[213] = 253 Cards[229] = 101
Cards[198] = 242 Cards[214] = 94 Cards[230] = 146
Cards[199] = 55 Cards[215] = 255 Cards[231] = 239
Cards[200] = 244 Cards[216] = 168 Cards[232] = 148
Cards[201] = 145 Cards[217] = 213 Cards[233] = 149
Cards[202] = 246 Cards[218] = 162 Cards[234] = 150
Cards[203] = 243 Cards[219] = 91 Cards[235] = 151
Cards[204] = 248 Cards[220] = 216 Cards[236] = 220
Cards[205] = 245 Cards[221] = 241 Cards[237] = 153
Cards[206] = 250 Cards[222] = 102 Cards[238] = 154
Cards[207] = 247 Cards[223] = 171 Cards[239] = 155
Cards[240] = 156 Cards[248] = 208
Cards[241] = 157 Cards[249] = 205
Cards[242] = 158 Cards[250] = 210
Cards[243] = 199 Cards[251] = 207
Cards[244] = 204 Cards[252] = 212
Cards[245] = 201 Cards[253] = 209
Cards[246] = 206 Cards[254] = 174
Cards[247] = 203 Cards[255] = 211

Setelah itu barulah proses enkripsi dilakukan pada tiap karakter hingga panjang pesan atau *string* terakhir.

Panjang string yang akan dienkripsi : LEN(“NEMSIS”) = 7

Jika dinyatakan dalam bentuk desimal maka :

N = 78 // message[1] = 78
E = 69 // message[2] = 69
M = 77 // message[3] = 77
E = 69 // message[4] = 69
S = 83 // message[5] = 83
I = 73 // message[6] = 73
S = 83 // message[7] = 83

Proses enkripsi Sapphire II

Ratchet = ((ratchet) + Cards (Rotor)) And &HFF
Ratchet = ((ratchet) + Cards (118)) And &HFF
Ratchet = 62

Rotor = Rotor + 1
Rotor = 119 + 1
Rotor = 119

Swaptemp = Cards (last_cipher)
Swaptemp = Cards (95)
Swaptemp = 44

Cards(last_Cipher) = Cards(ratchet)
Cards(95) = Cards(62)
Cards(95) = 36

Cards(ratchet) = Cards (last_plain)
Cards(98) = Cards (119)
Cards (98) = 6

Cards(Rotor) = swaptemp
Cards(119) = 44

Avalanche = (avalanche + Cards(swaptemp)) And &HFF
Avalanche = (174 + Cards(44)) And &HFFavalanche = 174

i = 1 //enkripsi karakter string pertama
Last_Cipher = 78 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 78 Xor Cards(Cards(62) + Cards(119) And &HFF)Xor
Cards(Cards((Cards222) + Cards(95) + Cards(174) And
&HFF))
LAST_Cipher = 222

i = 2 //enkripsi karakter string kedua
Last_Cipher = 69 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 69 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards33) + Cards(78) + Cards(174) And
&HFF))
LAST_Cipher = 33

i = 3 //enkripsi karakter string ketiga
Last_Cipher = 77 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 77 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards176) + Cards(78) + Cards(174) And
&HFF))
LAST_Cipher = 176

i = 4 //enkripsi karakter string keempat
Last_Cipher = 69 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 69 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards241) + Cards(77) + Cards(174) And
&HFF))
LAST_Cipher = 241

i = 5 //enkripsi karakter string kelima
Last_Cipher = 83 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 83 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards0) + Cards(69) + Cards(174) And
&HFF))
LAST_Cipher = 0

i = 6 //enkripsi karakter string keenam
Last_Cipher = 73 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 73 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards18) + Cards(83) + Cards(174) And
&HFF))
LAST_Cipher = 18

i = 7 //enkripsi karakter string ketujuh
Last_Cipher = 83 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 83 Xor Cards(Cards(62 + Cards(119 And &HFF)Xor
Cards(Cards((Cards101) + Cards(73) + Cards(174) And
&HFF))
LAST_Cipher = 101

Hasil dari proses enkripsi dapat diperlihatkan pada tabel dibwah ini :

Tabel 3.1 Tabel Hasil enkripsi string " NEMESIS" dengan Sapphire II

Perhitungan Avalanche effects nyaadalah sebagai berikut :
Biner pada string "NEMESIS" adalah :

01001110 01000101 01001101 01000101 01010011 01001001 01010011
Biner pada string hasil enkripsi d4engan Sapphire II :
11011110 00100001 10110000 11110001 00000000 00010010 01100101

Dari hasil diatas terlihat bahwa prbedaan bit pada posisi yang sama sebanyak 29 bit jadi avalanche effect-nya adalah sebagai berikut

$$\frac{Jumlah\ bit\ beda}{jumlah\ bit\ yang\ dibandingkan}\ x\ 100\% + \frac{29}{56}\ x\ 100\% = 51{,}78\%$$

Untuk proses dekripsi pada sapphire II urutan pengacakan nilai pada CARDS sama seperti dengan proses enkripsi termasuk juga algoritma deskripsinya miripdengan algoritma enkripsinya. Proses dekripsi dilakukan pada tiap karakter hingga panjang pesan atau string terakhir.

Panjang string yang akan didekripsi : LEN (■!± NULL ↕ *e*") = 7
Jika dinyatakan dalam bentuk desimal maka :

= 222 // cipher[1] = 222
= 33 // cipher[2] = 33
= 176 // cipher[3] = 176
= 241 // cipher[4] = 241
= 0 // cipher[5] = 0
= 18 // cipher[6] = 18
= 101 // cipher[7] = 101

i = 1 //enkripsi karakter string pertama
Last_Cipher = 222 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 222 Xor Cards(Cards(62) + Cards(119) And &HFF)Xor
Cards(Cards((Cards78) + Cards(95) + Cards(174) And
&HFF))
LAST_Cipher = 78

i = 2 //enkripsi karakter string kedua
Last_Cipher = 33 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 33 Xor Cards(Cards(62) + Cards(119) And &HFF)Xor
Cards(Cards((Cards69) + Cards(222) + Cards(174) And
&HFF))
LAST_Cipher = 69

i = 3 //enkripsi karakter string ketiga
Last_Cipher = 176 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +

Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 176 Xor Cards(Cards(62) + Cards(119) And &HFF)Xor
Cards(Cards((Cards77) + Cards(33) + Cards(174) And &HFF))
LAST_Cipher = 77

i = 4 //enkripsi karakter string keempat
Last_Cipher = 241 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 241 Xor Cards(Cards(62) + Cards(119 And &HFF)Xor
Cards(Cards((Cards69) + Cards(176) + Cards(174) And &HFF))
LAST_Cipher = 69

i = 5 //enkripsi karakter string kelima
Last_Cipher = 0 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 0 Xor Cards(Cards(62) + Cards(119 And &HFF)Xor
Cards(Cards((Cards83) + Cards(241) + Cards(174) And &HFF))
LAST_Cipher = 83

i = 6 //enkripsi karakter string keenam
Last_Cipher = 18 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 18 Xor Cards(Cards(62 + Cards(119) And &HFF)Xor
Cards(Cards((Cards73) + Cards(0) + Cards(174) And &HFF))
LAST_Cipher = 73

i = 7 //enkripsi karakter string ketujuh
Last_Cipher = 101 Xor Cards(Cards(ratchet) + Cards(Rotor) And &HFF)
Xor
Cards(Cards((Cards(last_plain) +
Cards(last_cipher) + Cards(avalanche)) And &HFF))

Last_Cipher = 101 Xor Cards(Cards(62) + Cards(119 And &HFF)Xor
Cards(Cards((Cards83) + Cards(18) + Cards(174) And &HFF))
Last_Cipher = 83

3.1.2 Perhitumgam Algoritma RC4

Proses pertama dimulai dengan melakukan proses enkripsi dengan algoritma RC4. *String* sampel dalam hal ini adalah Test.txt yang terdiri atas 7 byte dan merupakan plain text yang berisi "NEMESIS". Untuk algoritma RC4 ini pertama sekali dibentuk 256 buah Sbox yang akan dipakai sebagai kunci dalam RC4

Langkah berikut ini adalah membentuk *key*. Dengan adanya 256 buah Sbox berarti RC4 dapat mendukung hingga 256 karakter umtuk key-nya. Dalam pengujian ini digunakan key dengan panjang 3 karakter. Adapun kunci (key) yang dipakai adalah string "ABC".

Key = "ABC"
Key_length = 3
Key[0] = 65; key[1] = 66; key[2] = 67; // dalam bentuk desimal
j = 0

S[6]	= 226	S[22]	=	161
S[7]	= 96	S[23]	=	56
S[8]	=220	S[24]	=	145
S[9]	= 192	S[25]	=	236
S[10]	= 12	S[26]	=	179
S[11]	= 90	S[27]	=	10
S[12]	= 57	S[28]	=	13
S[13]	= 234	S[29]	=	99
S[14]	= 111	S[30]	=	194
S[15]	= 119	S[31]	=	35
S[64]	= 248	S[80]	=	159
S[65]	= 164	S[81]	=	252
S[66]	= 126	S[82]	=	238
S[67]	= 115	S[83]	=	130
S[68]	= 45	S[84]	=	65
S[69]	= 88	S[85]	=	151
S[70]	= 9	S[86]	=	228

S[71] = 74 S[87] = 203 S[166] = 75 S[182] = 25
S[72] = 87 S[88] = 98 S[167] = 26 S[183] = 247
S[73] = 202 S[89] = 121 S[168] = 219 S[184] = 190
S[74] = 185 S[90] = 235 S[169] = 204 S[185] = 152
S[75] = 41 S[91] = 16 S[170] = 59 S[186] = 114
S[76] = 133 S[92] = 173 S[171] = 58 S[187] = 94
S[77] = 134 S[93] = 110 S[140] = 86 S[156] = 214
S[78] = 85 S[94] = 224 S[141] = 129 S[157] = 40
S[79] = 36 S[95] = 102 S[142] = 37 S[158] = 66
S[128] = 225 S[144] = 253 S[143] = 196 S[159] = 200
S[129] = 89 S[145] = 205 S[192] = 86 S[208] = 39
S[130] = 183 S[146] = 24 S[193] = 129 S[209] = 79
S[131] = 242 S[147] = 103 S[194] = 37 S[210] = 72
S[132] = 113 S[148] = 171 S[195] = 196 S[211] = 211
S[133] = 8 S[149] = 136 S[196] = 86 S[212] = 202
S[134] = 162 S[150] = 245 S[197] = 129 S[213] = 60
S[135] = 0 S[151] = 218 S[198] = 148 S[214] = 240
S[136] = 167 S[152] = 92 S[199] = 244 S[215] = 165
S[137] = 251 S[153] = 166 S[200] = 6 S[216] = 51
S[138] = 182 S[154] = 30 S[201] = 117 S[217] = 193
S[139] = 195 S[155] = 123 S[202] = 100 S[218] = 212
S[36] = 23 S[52] = 5 S[203] = 80 S[219] = 3
S[37] = 239 S[53] = 209 S[204] = 223 S[220] = 118
S[38] = 31 S[54] = 138 S[205] = 82 S[221] = 178
S[39] = 70 S[55] = 28 S[206] = 255 S[222] = 180
S[40] = 181 S[56] = 230 S[207] = 139 S[223] = 124
S[41] = 232 S[57] = 215 S[172] = 187 S[188] = 227
S[42] = 140 S[58] = 64 S[173] = 175 S[189] = 73
S[43] = 63 S[59] = 71 S[174] = 188 S[190] = 153
S[44] = 68 S[60] =128 S[175] = 52 S[191] = 176
S[45] = 18 S[61] = 2 S[224] = 86 S[240] = 154
S[46] = 76 S[62] = 17 S[225] = 129 S[241] = 14
S[47] = 131 S[63] = 34 S[226] = 37 S[242] = 149
S[96] = 141 S[112] = 120 S[227] = 196 S[243] = 157
S[97] = 170 S[113] = 168 S[228] = 86 S[244] = 43
S[98] = 93 S[114] =54 S[229] = 129 S[245] = 206
S[99] =163 S[115] = 67 S[230] = 69 S[246] = 47
S[100] = 198 S[116] = 250 S[231] = 186 S[247] = 81
S[101] = 108 S[117] = 97 S[232] = 169 S[248] = 147
S[102] = 146 S[118] = 213 S[233] = 101 S[249] = 27
S[103] = 78 S[119] = 32 S[234] = 91 S[250] = 210
S[104] = 229 S[120] = 158 S[235] = 150 S[251] = 109
S[105] = 143 S[121] =127 S[236] = 142 S[252] = 11
S[106] = 135 S[122] = 19 S[237] = 233 S[253] = 29
S[107] = 184 S[123] = 125 S[238] = 243 S[254] = 231
S[108] = 172 S[124] = 106 S[239] = 216 S[255] = 144
S[109] = 48 S[125] = 249
S[110] = 197 S[126] = 62
S[111] = 61 S[127] = 254
S[160] = 191 S[176] = 17
S[161] = 222 S[177] = 53
S[162] = 104 S[178] = 22
S[163] = 156 S[179] = 189
S[164] = 4 S[180] = 199
S[165] = 15 S[181] = 95

Setelah itu barulah proses enkripsi dilakukan pada tiap karakter hingga panjang pesan atau string terakhir.

Panjang string yang akan dienkripsi : LEN (“NEMESIS) = 7
Jinyatakan dalam bentuk decimal maka:

N = 78 // message[1] = 78

E = 69	//message[2]	=	69
M = 77	//message[3]	=	77
E = 69	//message[4]		=69
S = 83	//message[5]	=	83
I = 73	//message[6]	=	73
S = 83	//message[7]	=	83

Hasil dari proses enkripsi dapat diperlihatkan pada table di bawah ini:
Tabel 3.3 Tabel Hasil Enkripsi String "NEMESIS" dengan RC4

Cipher(i)	**Biner**	**Heksadecimal**	**Desimal**	**Karakter**
1	**10000000**	**80**	**128**	**C**
2	**01100001**	**61**	**97**	**A**
3	**00010011**	**13**	**19**	**!!**
4	**11010110**	**D6**	**214**	
5	**00011100**	**1C**	**28**	**(cursor right)**
6	**11100111**	**E7**	**231**	
7	**00010101**	**15**	**21**	

Setelah itu proses dekripsi dilakukan pada *string* tersebut dengan terlebih dahulu sama seperti halnya dengan proses enkripsi dibentuk 256 buah SBox melalui proses *swapping* dengan key "ABC" (key length = 3).
Kemudian barulah proses dekripsi dilakukan pada tiap karakter hingga panjang *clipher* terakhir.
Panjang clipher yang akan didekripsikan : 7

3.2 Perancang Perangkat Lunak

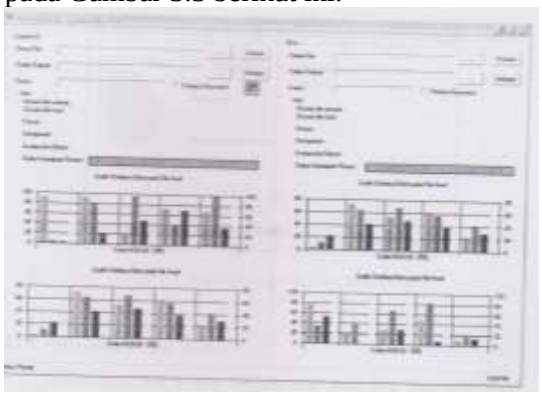
Pada bagian perancang akan dijelaskan hal yang menyangkut perancang program. Pada bagian ini akan dijelaskan rancangan tampilan (rancangan *form*). Perancang *form* program dilakukan dalam lingkungan Visual Basic.

Secara umum program dapat melakukan dua tipe operasi yaitu yang pertama melakukan proses enkripsi dan dekripsi langsung pada *file* dan bagian kedua menampilkan hasil distribusi byte dalam bentuk grafik. Khusus untuk proses pada *file* tidak akan ditunjukkan langkah enkripsi atau dekrifsi.

Pada operasi *file* dalam satu proses hanya dapat memproses satu *file* saja dan operasi ini akan bersifat menimpa *file* yang sama jika nama *folder* yang diberikan terdapat nama *file* yang sama.

3.2.1 Perancangan Form

Program ini hanya dibuat dalam dua tampilan *form* yaitu *form* utama dan *form splash screen*. Bentuk tampilan *form* utama ditunjukkan pada Gambar 3.3 berikut ini.



Gambar 3.3 Rancangan Bentuk Form Utama

Pada *form* utama ini perancang menggunakan komponen visual seperti MSChart, *text box, command button, common dialog, frame, progress bar,* dan *label* serta *check box.*
Sedangkan pada *form splash screen* ini hanya menggunakan objek seperti *label, command button, picture box,* dan *line*. Tampilan pada *form splash screen* ini dapat dilihat pada gambar 3.4 berikut ini.

Gambar 3.4 Rancangan Bentuk Form Splash Screen

3.2.2 Perancangan Class Module

Perancangan Class Module bertujuan agar objek *class* yaitu berisi fungsi-fungsi utama proses enkripsi /dekripsi dengan algoritma dapat dipakai

dengan mudah dan dapat dipakai kembali (*reuseable*) untuk pembuatan program lain yang memakai algoritma enkripsi sapphire II dan RC4 juga. Selain itu fungsi yang dideklarasikan berupa objek *class* akan lebih cepat dalam hal pemrosesan.

Class Module yang dibuat diberi nama clsSapphire II (singkatan dari Class Sapphire II). Class ini berisi rutin-rutin dari algoritma Sapphire II seperti fungsi untuk enkripsi *file*, dekripsi *file*, operasi XOR, ADD, Shift Left, dan AND , serta suatu fungsi untuk mengecek keberadaan suatu *file* pada lokasi *folder* tertentu.

Class Module yang kedua di beri nama clsRC4 (singkatan dari Class RC4). Class ini berisi rutin-rutin dari algoritma RC4 seperti fungsi untuk enkripsi *file*, dekripsi *file*, operasi XOR, ADD, dan Swap, serta suatu fungsi untuk mengecek keberadaan suatu *file* pada lokasi *folder* tertentu.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan dari bab-bab sebelumnya yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Sapphire II dan RC4 merupakan *stream cipher* yang melakukan proses secara per byte dengan panjang kunci maksimum 256 byte sehingga ukuran data input akan sama dengan ukuran data output.
2. Dalam hal Avalanche Effects algoritma Sapphire II secara umum beberapa persen di atas dari algoritma ini lebih unggul sedikit dibandingkan dengan RC4.
3. Hasil uji coba secara umum didapat metode RC4 mempunyai waktu kumputasi yang cepat yang disebabkan oleh algoritma RC4 lebih sederhana dibandingkan dengan Sapphire II.

5.1 Saran

Untuk pengembangan lebih lanjut program kompresi pada *file bitmap* ini, maka dapat diberikan beberapa saran sebagai berikut:

1. Program dapat ditambah dengan beberapa metode algoritma kriptografi yang lain agar Perbandingan yang dilakukan tidak hanya terbatas dua metode saja.
2. Perbandingan dilakukan dengan menggunakan format *file* yang lain dengan ukuran yang lebih bervariasi sehingga hasil pengujian yang di dapat lebih akurat.

[IEF05] http://www.ietf.org/rfc/rfc1321.text
[EFF05] http://www.eff.org/barlow, 2005
[RAH02] Raharjo, Agus, “CyberCrime”, bandug, Citra Aditya Bakti. 2002
[SCH98] Schneir, Bruce “Security Pitfalls in Cryptography”, Counterpane System, 1998
[WEI96] Wei Dai, “Crypto ++: a C++ Class Library Of Cryptographic Primitives Version 2.1, 1996