BAB II

LANDASAN TEORI

2.1 Keamanan Data

Dalam era digital, komunikasi melalui jaringan komputer memegang peranan penting. Melalui komunikasi elektronis, seseorang dapat melakukan transaksi atau komunikasi dengan sangat cepat dan praktis. Hal ini merupakan pengaruh dari perkembangan yang sangat signifikan dalam teknologi informasi, dimana *bandwidth* internet yang semakin besar dengan biaya akses yang semakin murah. Konsekuensinya adalah resiko dalam keamanan informasi semakin meningkat.

Komunikasi data elektronis memerlukan perangkat keamanan yang benar-benar berbeda dengan komunikasi konvensional. Dalam lalu lintas informasi di internet, sistem autentikasi (bukti diri) konvensional dengan KTP, SIM dan sebagainya yang bersandar pada keunikan tanda tangan tidak berlaku. Pengawalan petugas keamanan tidak lagi bisa membantu keamanan pengiriman dokumen elektronis.

Keamanan data dapat dibedakan menjadi dua kategori, yaitu keamanan fisik dan keamanan sistem. Keamanan fisik merupakan bentuk keamanan berupa fisik dari *server, terminal/client router* sampai dengan *cabling.* Sedangkan keamanan sistem adalah keamanan pada sistem pengoperasiannya atau lebih khususnya pada lingkup perangkat lunak, misalnya dengan penggunaan

kriptografi dan steganografi. Dalam penelitian ini akan dibahas tentang penggunaan kombinasi steganografi dan kriptografi dalam memberikan keamanan pada data.

2.2 Steganografi

Kata steganografi (*steganography*) berasal dari bahasa Yunani *steganos*, yang artinya tersembunyi atau terselubung, dan *graphia* yang artinya menulis, sehingga arti steganografi adalah "menulis (tulisan) terselubung" (Cvejic, 2004). Dengan steganografi, kita dapat menyisipkan pesan rahasia ke dalam media lain dan mengirimkannya tanpa ada yang menyadari keberadaan pesan tersebut (Krenn, 2004).

Steganografi adalah ilmu yang digunakan untuk menyisipkan data di dalam media lainnya. Steganografi membutuhkan dua media, yaitu penampung dan data yang akan disisipkan. Secara teori, semua berkas digital yang ada di dalam komputer dapat digunakan sebagai media penampung, misalnya citra berformat JPG, GIF, BMP, atau di dalam musik MP3, atau bahkan di dalam sebuah film dengan format WAV atau AVI. Semua dapat dijadikan tempat bersembunyi, asalkan berkas tersebut memiliki bit-bit yang tidak signifikan atau terdapat *redundant bits* yang dapat dimodifikasi. Setelah dimodifikasi, berkas media tersebut tidak akan terganggu fungsinya dan kualitasnya tidak akan jauh berbeda dengan aslinya.

2.2.1 Sejarah Steganografi

Penggunaan steganografi sebenarnya telah digunakan berabad-abad yang lalu bahkan sebelum istilah steganografi itu sendiri muncul. Pada tahun 480 Sebelum Masehi, seseorang berkebangsaan Yunani yaitu Demaratus mengirim pesan kepada polis Sparta yang berisi peringatan mengenai penyerangan Xerxes yang ditunda. Teknik yang digunakan adalah dengan menggunakan meja yang telah diukir kemudian diberi lapisan lilin untuk menutupi pesan tersebut, dengan begitu pesan dalam meja dapat disampaikan tanpa menimbulkan kecurigaan oleh para penjaga.

Pada abad ke 5 Sebelum Masehi, Histaiacus mengirimkan pesan kepada Aristagoras Miletus untuk memberontak terhadap raja Persia. Pesan disampaikan dengan cara mencukur kepala pembawa pesan dan mentato kepalanya dengan pesan tersebut. Setelah pesan dituliskan, pembawa pesan harus menunggu hingga rambutnya tumbuh kembali sebelum dapat mengirimkan pesan kepada pihak penerima. Pihak penerima kemudian akan mencukur rambut pembawa pesan tersebut untuk melihat pesan yang tersembunyi. Bangsa Romawi mengenal steganografi dengan menggunakan tinta tak-tampak (*invisible ink*) untuk menuliskan pesan. Tinta tersebut dibuat dari campuran sari buah, susu, dan cuka. Jika tinta digunakan untuk menulis maka tulisannya tidak tampak. Tulisan di atas kertas dapat dibaca dengan cara memanaskan kertas tersebut.

Ketika terjadi peristiwa 11 September, pemerintah Amerika Serikat mengklaim bahwa Usamah bin Ladin dan jaringan Al-Qaidah menggunakan steganografi untuk mengirim pesan melalui *website* dan *newsgroup*. Akan tetapi

sampai saat ini tidak ditemukan bukti substansial yang mendukung klaim Amerika Serikat tersebut. Hal ini memunculkan dua kemungkinan, apakah Usamah bin Ladin dan Al-Qaidah menciptakan dan menggunakan algoritma steganografi yang benar-benar kokoh, atau klaim tersebut yang salah dan hanya diada-adakan (Krenn, 2004).

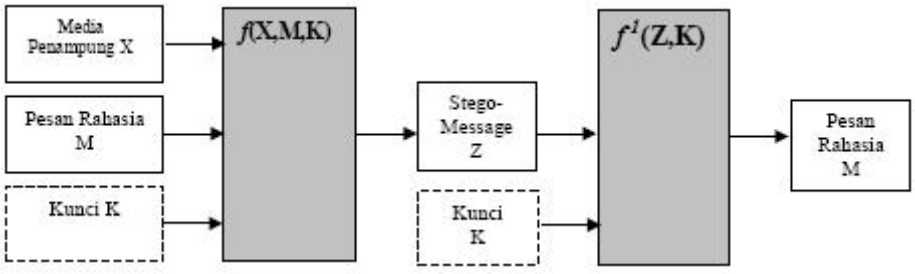
2.2.2 Manfaat Steganografi

Manfaat steganografi yang umum diketahui adalah mengamankan informasi rahasia dalam lalu lintas jaringan komunikasi. Ada banyak negara yang warga negaranya tidak bebas dalam berbicara ataupun mengeluarkan pendapat layaknya negara demokratis. Maka steganografi dapat menjadi sebuah solusi yang memungkinkan terjadinya pertukaran informasi dan berita tanpa dapat disensor dan tanpa perlu khawatir pesan tersebut disadap.

Selain itu steganografi juga dapat dimanfaatkan sebagai tempat penyimpanan informasi yang tidak disadari keberadaannya (*deniable storage)*. Contohnya informasi pribadi tentang perbankan, rahasia militer dan lainnya. Keberadaan media penyimpanan tersebut tidak dapat dideteksi oleh indera manusia, namun ketika dibutuhkan, maka informasi tersebut dapat dengan mudah dikeluarkan dari media steganografi tersebut. *Rubberhose* (http://www.rubberhose.org) telah mengimplementasikan manfaat steganografi ini pada proyeknya (Krenn, 2004).

2.2.3 Karakteristik Steganografi

Dalam steganografi dikenal *data hiding* atau *data embedding*, yang merupakan rangkaian proses dalam menyembunyikan data ke dalam berbagai media, seperti citra, audio atau teks dengan meminimalisir penampakan degradasi sinyal media penampung (Bender *et al*, 1996). Hal ini tampak familiar dengan enkripsi. Namun jika ditelusuri lebih jauh maka penyembunyian data dalam steganografi sangat kontras dengan kriptografi. Perbedaannya terletak pada bagaimana proses penyembunyian data dan hasil akhir dari proses tersebut. Kriptografi merahasiakan makna pesan sementara keberadaan pesan tetap dapat diamati oleh indera manusia. *Kriptografi* melakukan proses pengacakan data asli sehingga menghasilkan data terenkripsi yang benar-benar acak dan berbeda dengan aslinya. Sedangkan steganografi menyembunyikan keberadaan pesan tersebut, data disisipkan dalam media penampung tanpa mengubah keadaan media penampung tersebut. Dengan kata lain keluaran *steganografi* ini memiliki bentuk persepsi yang sama dengan bentuk aslinya.



Gambar 2.1 Diagram Sistem Steganografi

Penyisipan data rahasia ke dalam media digital harus memperhatikan beberapa aspek keamanan sebagai berikut:

1. *Fidelity*

   Mutu media penampung tidak jauh berubah. Setelah penambahan data rahasia, berkas hasil steganografi tidak mengalami degradasi yang signifikan, sehingga perubahan atau degradasi tersebut tidak dapat dipersepsi oleh indera manusia. Pada kasus audio steganografi, audio hasil steganografi masih dapat didengar dengan baik. Pengamat tidak menyadari bahwa di dalam audio tersebut terdapat data rahasia. Atau dengan kata lain penyisipan data rahasia tidak mempengaruhi kualitas sinyal asli, sehingga keberadaan pesan tidak dapat ditangkap oleh pendengaran manusia.

2. *Recovery*

   Data yang disembunyikan harus dapat diungkapkan kembali (*recovery*). Karena tujuan steganografi adalah *data hiding*, maka sewaktu-waktu data rahasia didalam citra penampung harus dapat diambil kembali untuk digunakan lebih lanjut.

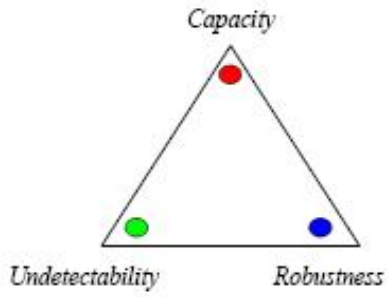
3. *Robustness*

   Robustness merupakan salah satu isu desain algoritma steganografi yang utama. Data rahasia yang disisipkan harus tahan terhadap pengolahan sinyal yang mungkin dilakukan termasuk konversi digital-analog dan analog-digital, *linear* dan *non-linear filtering*, kompresi dan perubahan ukuran (*scaling)*.

4. *Security*

Data rahasia harus resisten terhadap deteksi pembajakan dan juga diharapkan bisa menyulitkan dari usaha steganalisis. Ketika kerahasiaan itu ingin dibuka, dibutuhkan kunci untuk digunakan dalam proses penyisipan dan ekstraksi.

Dalam menyisipkan informasi ada beberapa faktor yang saling berkompetisi satu sama lain, artinya faktor-faktor tersebut tidak dapat dioptimalkan dalam satu waktu, yaitu kapasitas (*capacity*), anti deteksi (*undetectability)* dan kekokohan (*robustness*).



Gambar 2.2 Faktor-faktor yang Saling Berkompetisi dalam Steganografi

Kapasitas adalah besar pesan rahasia (*embedded message*) yang dapat disembunyikan dalam media penampung (*cover-object)* dengan menggunakan teknik steganografi tertentu. Anti-deteksi (*undetectability)* adalah kemampuan dalam menghindari deteksi. Pesan yang disisipkan tidak dapat dideteksi keberadaannya dalam suatu media. Misalnya jika steganografi menggunakan komponen derau pada citra digital dalam menyisipkan pesan, maka tidak membuat perubahan statistik yang signifikan pada media penampung pesan tersebut. Sedangkan kekokohan (*robustness)* adalah ukuran ketahanan teknik

steganografi dalam menghadapi berbagai macam manipulasi terhadap media penampung.

Seringkali ketiga faktor ini saling menghilangkan satu sama lain. Informasi dalam jumlah kecil dapat disembunyikan secara efektif tanpa dapat dipersepsi dengan mudah. Namun analisis statistik terhadap derau yang ada dapat dengan mudah mengungkap keberadaan informasi rahasia. Namun penyisipan informasi dengan jumlah yang lebih banyak dapat saja mengubah media penampung sehingga keberadaan informasi dapat dengan mudah dideteksi. Sedangkan teknik yang kokoh sering mengorbankan kapasitas.

2.2.4 Audio Steganografi

Audio steganografi adalah teknik penyisipan pesan rahasia dalam media suara (audio). Proses penyisipan pesan rahasia dalam sistem steganografi pada dasarnya dilakukan dengan mengidentifikasi media audio pembawa pesan, yaitu *redundant bit* yang mana dapat dimodifikasi tanpa merusak integritas dari media audio itu sendiri. Dalam mengaplikasikan steganografi pada berkas audio dapat dilakukan dengan berbagai teknik. Berikut adalah beberapa teknik yang dapat digunakan:

1. Penggantian bit. Cara ini lazim digunakan dalam teknik digital steganografi yaitu mengganti bagian tertentu dari bit-bit datanya dengan data rahasia yang disisipkan. Dengan metode ini keuntungan yang didapatkan adalah ukuran

pesan yang disisipkan relatif besar, namun berdampak pada hasil audio yang berkualitas kurang dengan banyaknya derau.

2. Metode kedua yang digunakan adalah merekayasa fasa dari sinyal masukan. Teori yang digunakan adalah dengan mensubstitusi awal fasa dari tiap awal segmen dengan fasa yang telah dibuat sedemikian rupa dan merepresentasikan pesan yang disembunyikan. Fasa dari tiap awal segmen ini dibuat sedemikian rupa sehingga setiap segmen masih memiliki hubungan yang berujung pada kualitas suara yang tetap terjaga. Teknik ini menghasilkan keluaran yang jauh lebih baik daripada metode pertama namun dikompensasikan dengan kerumitan dalam realisasinya.
3. Metode yang ketiga adalah penyebaran spektrum. Dengan metode ini pesan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan. Maka dari itu akan sangat sulit bagi yang akan mencoba memecahkannya kecuali ia memiliki akses terhadap data tersebut atau dapat merekonstruksi sinyal acak yang digunakan untuk menyebarkan pesan pada *range* frekuensi.
4. Metode terakhir yang sering digunakan adalah menyembunyikan pesan melalui teknik *echo*. Teknik menyamarkan pesan ke dalam sinyal yang membentuk *echo*. Kemudian pesan disembunyikan dengan memvariasikan tiga parameter dalam *echo* yaitu besar amplitude awal, tingkat penurunan atenuasi dan *offset*. Dengan adanya *offset* dari *echo* dan sinyal asli maka *echo* akan tercampur dengan sinyal aslinya, karena sistem pendengaran manusia yang tidak memisahkan antara *echo* dan sinyal asli.

Dalam penelitian ini teknik yang akan digunakan adalah teknik penggantian bit (*Parity Coding*), yaitu dengan mengganti byte-byte homogen pada berkas MP3 dengan pesan rahasia.

2.3 Kriptografi

Kriptografi berasal dari bahasa Yunani, *cryptos* yang berarti rahasia dan *graphein* yang berarti tulisan. Kriptografi adalah sebuah cara dalam mengamankan dan mengirim data dalam bentuk yang hanya diketahui oleh pihak yang berhak membukanya. Kriptografi merupakan ilmu dan seni dalam memproteksi informasi dengan mengubahnya ke dalam bentuk himpunan karakter acak yang tidak dapat dibaca. Kriptografi adalah sebuah cara yang efektif dalam mengamankan informasi-informasi penting baik yang tersimpan dalam media penyimpanan maupun yang ditransmisikan melalui jaringan komunikasi.

Walaupun tujuan utama dari kriptografi dan segala mekanisme yang dirancang adalah untuk mengamankan informasi dari pihak yang tidak berhak, sebagian besar algoritma dapat dipecahkan dan informasi dapat didapatkan jika penyerang mempunyai cukup waktu, kesungguhan dan sumber daya. Sehingga arahan realistis dari kriptografi dewasa ini adalah untuk merancang mekanisme algoritma yang kuat dan tahan terhadap serangan para penyerang.

2.3.1 Sejarah Kriptografi

Kriptografi telah digunakan sejak 4000 tahun lalu. Di Romawi, dikisahkan suatu saat Julius Caesar mengirimkan pesan rahasia kepada jenderalnya yang

sedang berada di medan perang. Ia mengacak pesan rahasia tersebut hingga menjadi pesan yang tidak dapat dipahami siapapun selain jenderalnya. Yang dilakukan Julius Caesar adalah menggeser 3 urutan alfabet ke kanan, sehingga a menjadi d, b menjadi e dan seterusnya. Dari ilustrasi tersebut, beberapa istilah kriptografi dipergunakan. Proses mengacak pesan disebut enkripsi. Pada saat jenderal menyusun kembali pesan acak tersebut menjadi pesan seperti sedia kala disebut dekripsi. Pesan awal yang belum diacak disebut *plaintext,* dan pesan yang telah diacak disebut *ciphertext.* Orang Mesir menggunakan kriptografi dengan huruf-huruf *hieroglyph.*



Gambar 2.3 Penggunaan *Hierogliph* di Mesir

Metode kriptografi pun berkembang sesuai dengan perkembangan zaman dan kebutuhan. Pada abad ke-15 Leo Battista Alberti menemukan metode roda kode (*wheel cipher)* yang dipergunakan sebagai alat enkripsi. Pada perang dunia ke-2, militer Jerman menggunakan mesin *cipher* substitusi polialfabetik yang disebut Enigma sebagai sistem pengkodean utama.

Enigma menggunakan rotor mekanis untuk pengkodean dan pendekodean. Seorang Belanda, Hugo Koch mengembangkan mesin ini pada 1919, dan diproduksi untuk pasar komersial pada 1923 oleh Arthur Scherbius. Scherbius

mendapatkan hak paten pada mesin Enigma untuk perusahaan Berlin Chiffriermasschinen Aktiengesellschaft. Pakar kriptanalisis Polandia, Marian Rejewski, bekerja bersama Perancis dari 1928 sampai 1938, berhasil memecahkan pengkabelan sistem 3 rotor yang digunakan Jerman saat itu dan menciptakan berkas kartu yang dapat mengantisipasi 6 kali 17.576 kemungkinan posisi rotor.

Jerman mengubah indikator sistem dan jumlah rotor menjadi 6 pada 1938, sehingga meningkatkan kesulitan untuk memecahkan *cipher* Enigma. Dalam kerjanya pada 1938, Polandia dan Perancis mengkonstruksi mesin prototipe yang disebut “The Bombe” untuk memecahkan *cipher* Enigma. Namanya diturunkan dari bunyi detikan yang dihasilkan oleh mesin. Usaha memecahkan *cipher* Enigma diambil alih oleh Inggris di Bletchley Park Inggris dan dipimpin oleh banyak ilmuwan terkemuka termasuk Alan Turing. Prototipe Bombe Turing muncul pada 1940, dan Bombe berkecepatan tinggi dikembangkan oleh Inggris dan Amerika pada 1943.

Perkembangan komputer dan sistem komunikasi pada tahun 60-an berdampak pada permintaan dari sektor-sektor swasta sebagai sarana untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan. Dimulai dari usaha Feistel pada IBM di awal tahun 70-an dan mencapai puncaknya pada 1977 dengan pengangkatan DES (*Data Encryption Standard*) sebagai standar pemrosesan informasi federal US untuk mengenkripsi informasi. DES merupakan mekanisme kriptografi yang paling dikenal sepanjang sejarah.

Di Amerika Serikat, sejak tanggal 4 November 1952 telah didirikan *National Security Agency* (NSA) yang diresmikan oleh Presiden Harry Truman. Keputusan pendirian NSA diambil setelah memperhatikan kesuksesan dalam perang dunia kedua di Pasifik yang merupakan kontribusi dari keberhasilan dalam memecahkan kode rahasia Jerman dan Jepang. NSA (www.nsa.gov) merupakan organisasi kriptografi nasional Amerika Serikat, dimana para pembuat kode dan pemecah kode berhimpun. Misi utama NSA adalah untuk melindungi sistem keamanan nasional Amerika Serikat dan untuk mengetahui informasi intelijen dari pihak asing.



Gambar 2.4 Logo NSA (*National Security Agency*) dan CSS (*Central Security Service*)

Pengembangan paling mengejutkan dalam sejarah kriptografi terjadi pada 1976 saat Diffie dan Hellman mempublikasikan *New Directions in Cryptography*. Tulisan ini memperkenalkan konsep revolusioner kriptografi kunci publik dan juga memberikan metode baru dan jenius untuk pertukaran kunci, keamanan yang berdasar pada kekuatan masalah logaritma diskrit. Pada 1978 Rivest, Shamir dan Adleman menemukan rancangan enkripsi kunci publik dan tanda tangan, yang sekarang disebut RSA. Rancangan RSA berdasar pada masalah matematika yang

sulit untuk kriptografi menggiatkan kembali usaha untuk menemukan metode yang lebih efisien untuk pemfaktoran. Kelas lain yang merupakan rancangan kunci publik praktis ditemukan oleh ElGamal pada 1985. Rancangan ini juga berdasar pada masalah logaritma diskret.

Selama bertahun-tahun kriptografi menjadi bidang khusus yang hanya dipelajari oleh kalangan militer, seperti agen keamanan nasional Amerika (*National Security Agency)*, Uni Soviet, Inggris, Perancis, Israel dan negara-negara lain yang rela membelanjakan miliaran dolar untuk mengamankan komunikasi mereka dari pihak luar. Mereka juga selalu mempelajari kode-kode rahasia dari negara lain. Namun dalam 30 tahun terakhir kriptografi tidak hanya dimonopoli kalangan militer saja. Kriptografi meluas di kalangan sipil, yaitu mereka yang menginginkan pesan dan komunikasi mereka aman dari pihak lain. Bahkan kriptografi kini merambah ke dunia sastra dengan dijadikan tema dalam novel maupun film.

2.3.2 Tujuan Kriptografi

Tujuan kriptografi adalah melindungi data dari ancaman yang disengaja atau tidak disengaja. Dewasa ini ancaman bertambah karena semakin meluasnya akses melalui internet atau teknologi bergerak. Aspek-aspek keamanan data dalam kriptografi adalah sebagai berikut (Kurniawan, 2004).

1. *Confidentiality / Privacy*

   Merupakan usaha untuk menjaga kerahasiaan data. Data hanya boleh diakses oleh orang yang berwenang. Contohnya data-data pribadi, data-data bisnis,

daftar gaji, data nasabah dan lainnya. Aspek keamanan data menjadi sangat sensitif dalam *e-commerce* dan militer. Serangan dalam aspek ini antara lain dilakukan dengan penyadapan, misalnya *sniffer* atau *logger*.

2. *Integrity*

   Memastikan bahwa informasi yang dikirim melalui jaringan tidak mengalami modifikasi oleh pihak yang tidak berhak. Serangan dapat berupa pengubahan data oleh orang yang tidak berhak, misalnya dengan *spoofing* yaitu virus yang dapat mengubah berkas.

3. *Availability*

   Informasi harus tersedia ketika dibutuhkan. Serangan dapat berupa meniadakan layanan (Denial of Service/DoS *attack*) atau menghambat layanan dengan membuat *server* lambat.

4. *Non-repudiation*

   Pengirim tidak dapat menyangkal bahwa yang bersangkutan telah melakukan transaksi tersebut.

5. *Authentication*

   Meyakinkan keaslian data, sumber data, orang yang mengakses data, dan *server* yang digunakan. Beberapa cara yang dapat digunakan untuk membuktikan keaslian data antara lain dengan *what you have* (misalnya kartu identitas), *what you know* (misalnya *password* atau PIN) dan *what you are* (misalnya dengan *biometric identity*). Serangan dapat dilakukan dengan menggunakan identitas palsu, terminal palsu ataupun situs gadungan.

6. *Access Control*

   Aspek ini berhubungan dengan mekanisme pengaturan akses ke informasi, untuk mengatur siapa yang boleh melakukan apa. Membutuhkan adanya klasifikasi data, misalnya umum (*public*), pribadi (*private*), rahasia (*confidential*) atau sangat rahasia (*top secret*).

7. *Accountability*

   Dapat dipertanggungjawabkan melalui mekanisme *logging* dan *audit*. Adanya kebijakan dan prosedur (*policy and procedures*).

2.3.3 Komponen Kriptografi

Pada dasarnya kriptografi terdiri dari beberapa komponen sebagai berikut (Ariyus, 2007).

1. Algoritma, merupakan himpunan aturan matematis yang digunakan dalam enkripsi dan dekripsi.
2. Enkripsi, adalah transformasi data ke dalam bentuk yang tidak dapat terbaca tanpa sebuah kunci tertentu.
3. Dekripsi, merupakan kebalikan dari enkripsi, yaitu transformasi data terenkripsi kembali ke bentuknya semula.
4. Kunci, digunakan pada saat melakukan enkripsi dan dekripsi. Pada kriptografi modern, keamanan enkripsi tergantung pada kunci, dan tidak tergantung kepada algoritmanya apakah dilihat orang lain atau tidak.

5. Pesan asli *(Plaintext),* disebut juga dengan *clear-text,* merupakan teks asli yang akan diproses menggunakan algoritma kriptografi tertentu untuk menjadi *ciphertext.*
6. *Ciphertext,* merupakan pesan yang telah melalui proses enkripsi yang merupakan himpunan karakter acak.
7. Kriptologi, merupakan studi tentang kriptografi dan kriptanalisis.
8. Kriptanalis (*Cryptanalysis),* merupakan aksi memecahkan mekanisme kriptografi dengan cara menganalisisnya untuk menemukan kelemahan dari suatu algoritma kriptografi sehingga akhirnya dapat ditemukan kunci atau teks asli.
9. Kriptosistem, adalah perangkat keras atau implementasi perangkat lunak kriptografi yang diperlukan dalam mentransformasi sebuah pesan asli menjadi *ciphertext* dan juga sebaliknya.

2.3.4 Serangan terhadap Kriptografi (*Cryptanalysis*)

Kriptanalisis (c*ryptanalysis)* adalah ilmu untuk mendapatkan pesan asli (*plaintext*) tanpa harus mengetahui kunci secara wajar. Pemecahan sandi rahasia yang berhasil akan menghasilkan *plaintext* atau kunci. Kriptanalisis juga dapat menemukan kelemahan dari kriptosistem yang pada akhirnya dapat menemukan kunci atau *plaintext*. Asumsi dasar dalam kriptanalisis pertama kali diungkapkan oleh Dutchman A Kerckhoffs, yaitu bahwa kerahasiaan harus terletak pada kunci. Kerckhoffs mengasumsikan bahwa kriptanalis mempunyai detail lengkap tentang algoritma kriptografi dan implementasinya (Forouzan, 2008). Kualitas algoritma

dapat diukur melalui kompleksitas dalam melakukan serangan terhadap algoritma kriptografi tersebut. Hal ini dapat diukur melalui beberapa aspek di bawah ini:

1. *Data Complexity*

   Jumlah data yang diperlukan sebagai masukan dalam penyerangan. Semakin sedikit jumlah data yang diperlukan untuk melakukan serangan, berarti kualitas algoritma yang digunakan semakin tidak baik.

2. *Processing Complexity*

   Terkait dengan lama waktu yang tersedia untuk melakukan serangan. Sering juga disebut dengan faktor kerja. Semakin cepat waktu yang dibutuhkan, berarti semakin buruk kualitas algoritma yang digunakan.

3. *Storage Requirements*

   Jumlah memori yang dibutuhkan untuk melakukan serangan.

Ada 3 kondisi yang apabila dipenuhi oleh algoritma kriptografi, maka akan dapat menjamin keamanan komunikasi rahasia yang dilakukan, yaitu:

1. Apabila biaya untuk menyerang atau menembus algoritma kriptografi yang digunakan lebih besar daripada harga informasi yang akan diperoleh dari hasil serangan tersebut. Misalnya, diperlukan sistem komputer senilai 1 milyar untuk menembus algoritma yang digunakan untuk melindungi informasi senilai 500 juta.
2. Apabila waktu yang diperlukan untuk menembus algoritma itu lebih lama daripada waktu validasi informasi yang ingin diperoleh. Misalnya, waktu untuk menembus sebuah kartu kredit adalah 1 tahun, sedangkan sebelum 1 tahun kartu kredit tersebut sudah tidak berlaku lagi.

3. Apabila *ciphertext* yang dihasilkan oleh suatu algoritma kriptografi lebih sedikit dari cipherteks yang diperlukan untuk menembus algoritma tersebut. Misalnya diperlukan sebesar 1000 bit *ciphertext* untuk menebak kunci yang digunakan pada suatu algoritma, sedangkan data yang dihasilkan dari proses enkripsi besarnya kurang dari 1000 bit.

2.3.5 Algoritma Rijndael

Sejak tahun 1976, *Data Encryption Standard* (DES) dipilih sebagai standar kriptografi yang dipakai pada pemerintahan Amerika Serikat. Namun pada tahun 1990, panjang kunci DES dianggap terlalu pendek, dan pada tahun 1998 DES berhasil dipecahkan dalam waktu 96 hari, kemudian di tahun 1999 dapat dipecahkan dalam waktu 22 hari.

Karena alasan tersebut maka kemudian diadakan kompetisi oleh NIST (*National Institute of Standard and Technology)* untuk mencari pengganti DES. NIST mengundang peserta dari seluruh dunia untuk berpartisipasi dengan mengajukan algoritma baru untuk menggantikan DES (Ariyus, 2008). Pada tahun 1997 ada 21 pelamar dan 6 dari mereka gugur karena tidak masuk dalam kriteria pemilihan. Kemudian dipilih 5 kandidat untuk seleksi akhir, yaitu Mars (IBM Amerika), RSA (RSA Corp. Amerika), Rijndael (Belgia), Serpent (Israel, Norwegia dan Inggris) dan Twofish (Counterpane Amerika). Pada tahap ini NIST memberikan kriteria sebagai berikut.

1. Aspek keamanan, NIST memberikan kesempatan dalam 3 tahun kepada komunitas kriptografer untuk menganalisis dan mengevaluasi kandidat-

kandidat tersebut, sehingga dapat diketahui mana algoritma yang kuat dan yang lemah.

2. Implementasi software, mempunyai kategori kecepatan eksekusi, kemampuan antar-platform dan variasi dari kecepatan dengan ukuran kunci.
3. Ruang lingkup yang tidak terbatas, algoritma dapat digunakan pada banyak aplikasi seperti *smart card, Random Acces Memory* (RAM) dan sebagainya.
4. Implementasi perangkat keras, algoritma mampu mengoptimalkan kecepatan atau ukuran, tidak membutuhkan memori yang besar untuk menjalankannya, sehingga sumber daya tidak banyak terpakai.
5. Algoritma untuk enkripsi dan dekripsi berbeda.
6. Ketahanan terhadap serangan, tidak ada jenis serangan yang dapat memecahkan algoritma yang ada. Dari segi kunci tidak mudah dipecahkan.
7. Kemampuan kunci, mengacu kepada kemampuan kecepatan pertukaran kunci dan memiliki sumber yang sedikit.
8. Potensial untuk instruksi paralel, mengacu kepada kemampuan prosesor masa depan.

Melalui seleksi yang sangat ketat dan persyaratan yang tidak mudah maka pada 2 Oktober 2000 terpilih algoritma Rijndael yang dibuat oleh Joan Daemen dan Vincent Rijmen sebagai pemenang.

2.3.6 Unit Data pada Algoritma Rijndael

Rijndael menggunakan 5 unit pengukuran data yaitu bit, byte, word, blok dan *state*. Bit adalah sebuah *binary digit* dengan nilai 0 atau 1 yang merupakan unit pengukuran terkecil. Sebuah byte terdiri dari 8 bit. 1 word terdiri dari 4 byte (32 bit). Rijndael mengenkripsi dan mendekripsi blok data. Untuk menyesuaikan dengan AES, maka sebuah blok pada Rijndael terdiri dari 128 bit. Rijndael menggunakan beberapa putaran dimana setiap putaran terdiri dari beberapa tahapan. Blok data ditransformasi dari satu tahapan ke tahapan selanjutnya. Sebelum dan sesudah menjalani tiap tahapan, blok data disebut sebagai *state* (Forouzan, 2008).

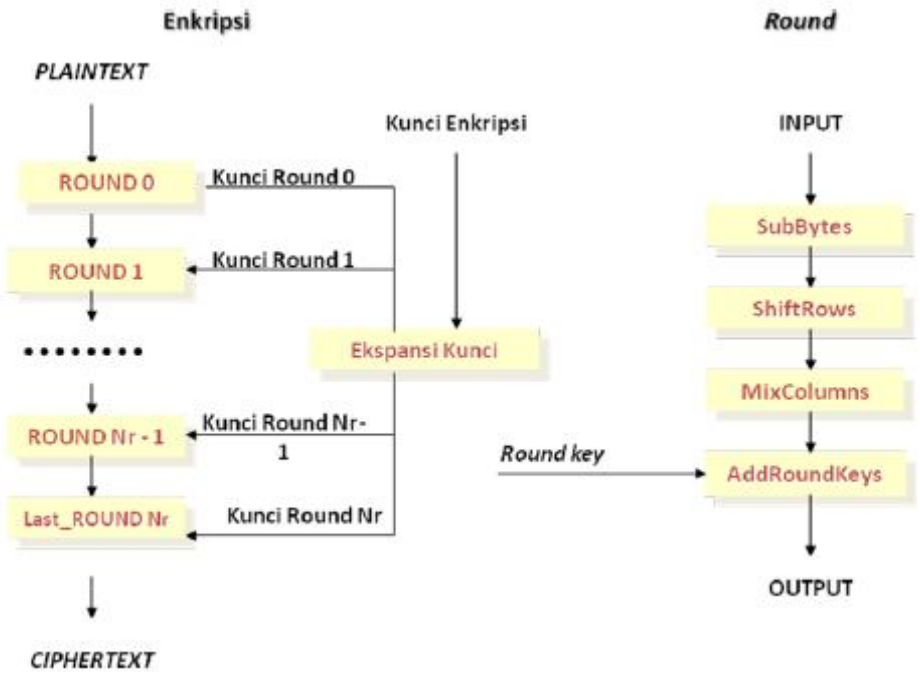
2.3.7 Transformasi pada Algoritma Rijndael

Algoritma Rijndael menggunakan substitusi, permutasi dan sejumlah putaran yang dikenakan pada tiap blok yang akan dienkripsi/dekripsi. Untuk setiap putarannya, Rijndael menggunakan kunci yang berbeda. Kunci setiap putaran disebut *round key*. Pada Rijndael dimungkinkan untuk menggunakan panjang kunci yang berbeda tergantung pada tingkat keamanan yang dibutuhkan untuk suatu aplikasi. Rijndael dapat dikategorikan sebagai *iterated block cipher* dengan panjang kunci yang dapat dipilih secara independen sebagai 128, 192, atau 256 bit. Dalam desimal ada kira-kira 3.4 x 1038 kemungkinan kunci 128 bit, 6.2 x 1057 kemungkinan kunci 192 bit, dan 1.1 x 1077 kemungkinan kunci 256 bit. AES menspesifikasikan tiga ukuran kunci, 128, 192, dan 256 bit dengan ukuran blok tetap 128 bit. (Forouzan, 2008).

Tabel 2.1 Parameter Rijndael

	Kunci 128	Kunci 192	Kunci 256
Panjang Kunci	4 word (16 byte)	6 word (24 byte)	8 word (32 byte)
Panjang *Plaintext*	4 word (16 byte)	4 word (16 byte)	4 word (16 byte)
Jumlah Putaran	10	12	14
Ukuran *Round Key*	4 word (16 byte)	4 word (16 byte)	4 word (16 byte)
Ukuran Kunci Ekspansi	44 word (176 byte)	52 word (208 byte)	60 word (240 byte)

Pada intinya, tiap blok masukan (*array state*) dikenakan empat fungsi utama, yaitu SubBytes, ShiftRows, MixColumns dan AddRoundKey (Daemen *et al*, 1999). *SubBytes* adalah transformasi substitusi pada setiap byte menggunakan tabel kotak S. *ShiftRows* adalah transformasi dengan menggeser baris ke-r dalam *array state* sebanyak r byte ke kiri. *MixColumns* adalah mengacak *array state* dengan cara melakukan perkalian matriks yang merupakan transformasi dari perkalian polinom antara tiap kolom dengan polinom empat suku pada GF(28), dimana GF adalah singkatan dari Galois Field. Dan terakhir adalah *AddRoundKey* yaitu melakukan *XOR* antara *array state* sekarang dengan *round key*.



Gambar 2.5 Skema Enkripsi Rijndael

Algoritma transformasi pada Rijndael dapat dilihat dalam *pseudocode* berikut.

```
Rijndael_Cipher (byte [] block_data, byte [] kunci)
begin
     Expand_Key(kunci, kunci_ekspansi)
     Add_Key(State[],kunci_ekspansi[0])
     DO (Nr – 1 putaran)
          ROUND(State, kunci_ekspansi[i])
          Last_Round (State, kunci_ekspansi[Nr])
          end
Round(State, kunci_ekspansi[i])
begin
```

```
     SubBytes(State)
     ShiftRows(State)
     MixColumns(State)
     AddRoundKey(State[],kunci_ekspansi[i])
end.
Last_Round (State, kunci_ekspansi[Nr])
begin
     SubBytes(State)
     ShiftRows(State)
     AddRoundKeys(State[],kunci_ekspansi[i])
End.
```

State yang dimaksud pada algoritma di atas merupakan hasil sementara tahapan algoritma Rijndael. *State* dapat direpresentasikan dengan notasi matriks. Struktur matriks selalu terdiri dari 4 baris, namun jumlah kolom tergantung pada banyaknya bit sesuai dengan panjang kunci. Misalnya untuk panjang blok masukan 128 bit direpresentasikan dengan matriks (4x4) sebagai berikut :

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

Jumlah kolom pada blok masukan disebut dengan Nb, yang sama dengan panjang blok dibagi dengan 32. Sedangkan parameter Nk digunakan untuk

menyatakan jumlah kolom pada matriks kunci. Sebagai contoh diperlihatkan pada blok masukan dan kunci berikut :

Masukan	32	43	F6	A8	88	5A	30	8D	31	31	98	A2	E0	37	07	34
Kunci	2B	7E	15	16	28	AE	D2	A6	AB	F7	15	88	09	CF	4F	3C

Karena panjang masukan dan kunci adalah 128 bit, diperoleh nilai Nk = Nb = 4, direpresentasikan dalam bentuk matriks sebagai berikut.

$$Input = \begin{bmatrix} 32 & 88 & 31 & E0 \\ 43 & 5A & 31 & 37 \\ F6 & 30 & 98 & 07 \\ A8 & 8D & A2 & 34 \end{bmatrix} \quad Key = \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$$

2.3.7.1 Substitusi (SubBytes)

Pada operasi SubBytes, Rijndael menggunakan sebuah kotak-S. Penggunaan kotak-S ditujukan agar enkripsi kebal dari serangan kriptanalisis yang menggunakan metode diferensial dan linier serta serangan yang menggunakan manipulasi aljabar lainnya. Koordinat x merepresentasikan digit pertama dari bilangan heksadesimal, sedangkan koordinat y merepresentasikan digit kedua. Misalnya untuk bilangan heksadesimal 08, maka 0 berada di koordinat x dan 8 berada di koordinat y. Ilustrasi kotak-S dapat dilihat pada gambar di bawah ini.

hex	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

x

Gambar 2.6 Kotak-S untuk Algoritma Rijndael

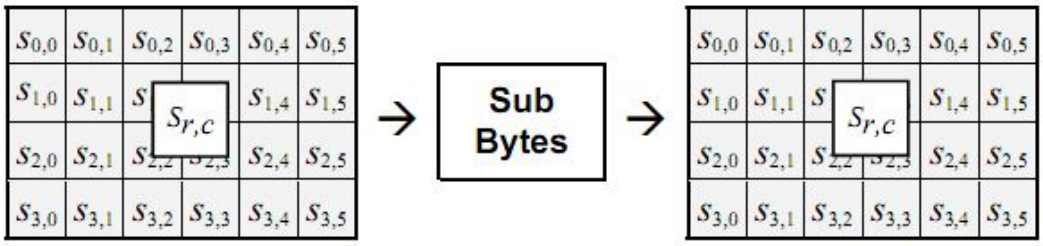
Secara sederhana, transformasi SubBytes menggunakan kotak-S dapat dinyatakan sebagai berikut.

```
State [row, column] = S-Box [state [row, column]]
```

Sebagai contoh operasi SubBytes sebagai berikut :

$$s = \begin{bmatrix} 19 & A0 & 9A & E9 \\ 3D & F4 & C6 & F8 \\ E3 & E2 & 8D & 48 \\ BE & 2B & 2A & 08 \end{bmatrix}$$ Maka $$s' = \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix}$$

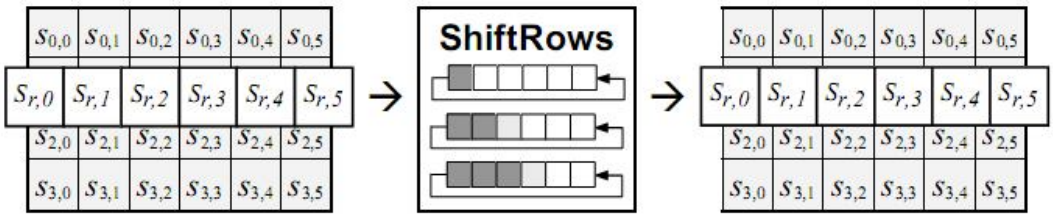
Tidak seperti kotak-S pada DES yang bersifat rahasia, rancangan dari kotak-S pada Rijndael bersifat publik. Transformasi SubByte adalah substitusi byte non linier, dimana operasi pada tiap *State* berjalan secara independen. Kotak-S juga bersifat dapat dibalikkan (*invertible*) dan dibangun dengan komposisi dari dua transformasi.



Gambar 2.7 Ilustrasi Transformasi SubBytes

2.3.7.2 Permutasi (ShiftRows)

Transformasi selanjutnya adalah pergeseran. Transformasi pergeseran pada Rijndael dilakukan pada tingkat byte, sehingga susunan pada tingkat bit tidak berubah. Transformasi pergeseran ini disebut ShiftRows dan pergeseran dilakukan ke sebelah kiri. Jumlah pergeseran tergantung pada urutan baris (0, 1, 2 atau 3) dari matriks *state*. Artinya baris 0 tidak mengalamami pergeseran dan baris terakhir digeser 3 byte ke kiri. Pada proses dekripsi transformasi disebut InvShiftRows dan pergeseran dilakukan ke sebelah kanan. Junlah pergeseran sama dengan proses enkripsi.



Gambar 2.8 Ilustrasi Transformasi ShiftRow

Algoritma transformasi ShiftRows dapat dinyatakan dalam *pseudocode* berikut.

```
ShiftRows(State)
begin
     for (r = 1 to 3)
          shiftrow(sr, r) // sr adalah baris ke-r
end
Shiftrow (row, n)
begin
     CopyRow(row, t) // t adalah baris (row)sementara
     for (c = 0 to 3)
          row(c-n)mod 4 ← tc
end.
```

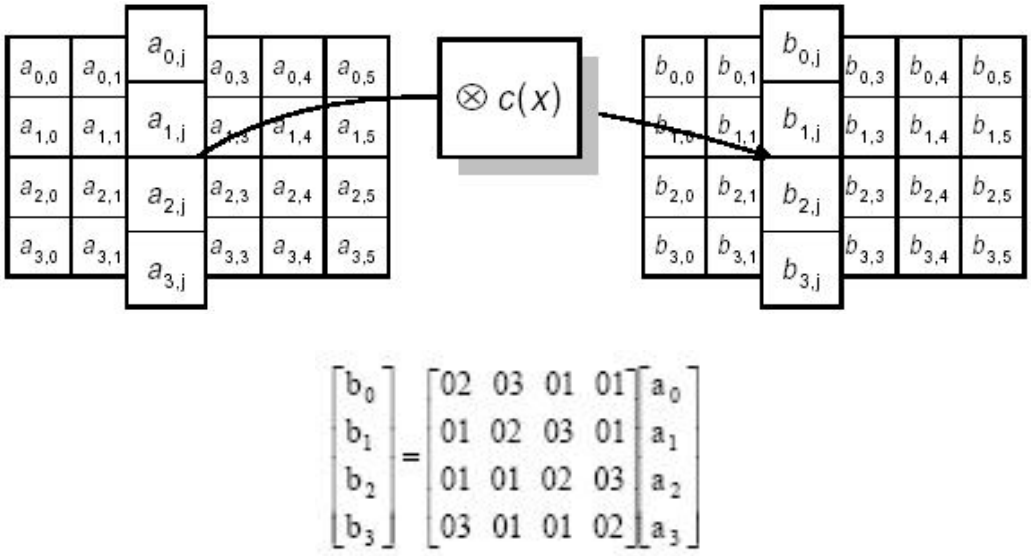
Contoh operasi ShiftRows pada *state* adalah sebagai berikut:

$$s = \begin{bmatrix} D4 & E0 & B8 & 1E \\ 27 & BF & B4 & 41 \\ 11 & 98 & 5D & 52 \\ AE & F1 & E5 & 30 \end{bmatrix} \quad s' = \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix}$$

2.3.7.3 *Mixing (*MixColumn)

MixColumn mengubah nilai dari setiap byte dengan mengambil 4 byte dan mengkombinasikannya untuk menghasilkan 4 byte baru. Untuk menjamin bahwa setiap byte berbeda (sekalipun keempat byte adalah sama), proses pengkombinasian dilakukan dengan mengalikan tiap byte dengan konstanta yang berbeda.

Transformasi MixColumn beroperasi pada tingkat kolom, yang mentransformasi setiap kolom menjadi kolom yang baru. Transformasi ini menggunakan perkalian matriks antara kolom *state* dengan matriks konstanta. Byte-byte pada kolom *state* dan matriks konstanta diinterpretasikan dalam 8 bit. Perkalian dilakukan pada GF(28) dengan modulus (10001101) atau (x8 + x4 + x3 + x + 1). Penjumlahan yang dimaksud adalah proses XOR dari 8 bit (Forouzan, 2008)



Gambar 2.9 Ilustrasi Transformasi MixColumn

Contoh matriks yang mengalami transformasi MixColumn dapat dilihat sebagai berikut.

$$s = \begin{bmatrix} D4 & E0 & B8 & 1E \\ BF & B4 & 41 & 27 \\ 5D & 52 & 11 & 98 \\ 30 & AE & F1 & E5 \end{bmatrix} \qquad s' = \begin{bmatrix} 04 & E0 & 48 & 28 \\ 66 & CB & F8 & 06 \\ 81 & 19 & D3 & 26 \\ E5 & 9A & 7A & 4C \end{bmatrix}$$

2.3.7.4 Ekspansi Kunci (AddRoundKey)

Transformasi yang paling penting adalah transformasi yang melibatkan kunci *cipher*. Seluruh transformasi sebelumnya menggunakan algoritma yang *invertible*. Jika kunci *cipher* tidak ditambahkan pada *state* di setiap putaran, maka akan sangat mudah bagi kriptanalis untuk mendapatkan *plaintext.* Rijndael menggunakan mekanisme ekspansi kunci yang memberikan Nr+1 kunci putaran dari kunci *cipher*. Setiap kunci putaran adalah 128 bit yang dinyatakan dalam 4 word.

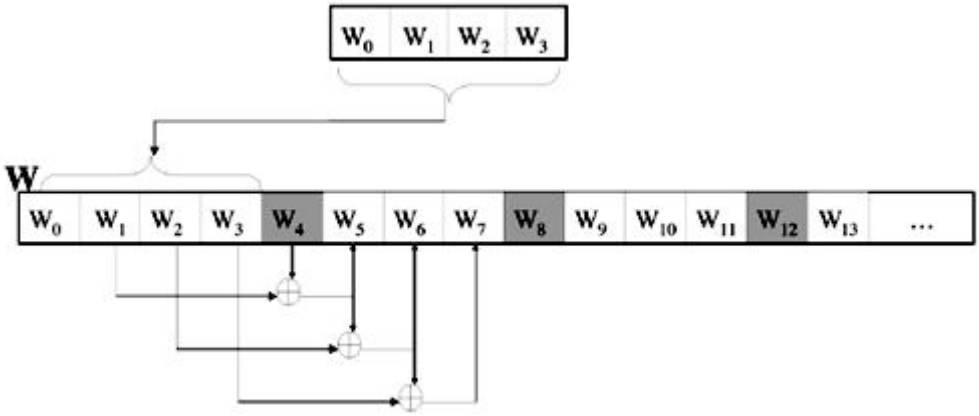
Untuk mendapatkan kunci untuk setiap putaran, Rijndael menggunakan mekanisme ekspansi kunci. Jika jumlah putaran adalah Nr, akan diperoleh rutin ekspansi kunci sebanyak Nr+1, dimana kunci setiap putaran adalah 128 bit yang diturunkan dari 128 bit kunci *cipher* utama.

Ekspansi kunci diperlukan untuk memenuhi kebutuhan kunci yang dapat mencapai ribuan bit untuk melakukan enkripsi, sementara kunci enkripsi yang disediakan hanya 128 hingga 256 bit saja. Total kunci yang diperlukan Rijndael adalah Nb(Nr+1) word. Sehingga jika kita menggunakan kunci 128 bit maka akan dibutuhkan 4(10+1) = 44 word = 44 x 32 bit = 1408 kunci. Dimana Nb adalah jumlah blok masukan yaitu 4 word (128 bit), dan Nr adalah jumlah putaran yang disesuaikan dengan ukuran kunci. Dengan kata lain kunci *cipher* sebanyak 128 bit diekspansi hingga menjadi 1408 bit.

Tabel 2.2 Parameter Jenis Kunci dan Jumlah Putaran Rijndael

Jenis Kunci	Panjang Kunci (word)	Jumlah Putaran Nr
128	4	10
192	6	12
256	8	14

Proses ekspansi kunci dapat dijelaskan dalam gambar berikut :



Gambar 2.10 Ilustrasi Ekspansi Kunci

Kunci *cipher* utama adalah sebuah *array* 16 byte (k0-k15). 4 byte pertama (k0-k3) menjadi w0, (k4-k7) menjadi w1 dan seterusnya. Dari kunci *cipher* utama didapatkan 4 word pertama (w0-w3). Nilai word selanjutnya yaitu w4-w43 dihasilkan dari proses sebagai berikut :

a. Jika (i mod 4) = 0, maka wi = t ⊕ wi-4. Dimana t adalah sebuah word sementara (*temporary*) yang merupakan hasil dari dua rutin, yaitu SubWord dan RotWord pada wi-1 dan hasil XOR dengan sebuah konstanta putaran Rcon (*Round Constants*).
b. Jika (i mod 4) = 0, maka wi = wi-1 ⊕ wi-4.

Algoritma transformasi AddRoundKey dapat dinyatakan dalam *pseudocode* berikut.

```
AddRoundKey(State)
begin
      for (c = 0 to 3)
      sc ← sc ⊕ Wround + 4c
end
KeyExpansion ([k0 to k15], [W0 to W15])
begin
      for (i = 0 to 3)
            Wi ← k4i + k4i+1 + k4i+2 + k4i+3
      for (i = 4 to 43)
      begin
            If (i mod 4 ≠ 0)
                  Wi ← Wi-1 + Wi-4
            else
            begin
                  t ← SubWord(RotWord(Wi-1)) ⊕ Rconi/4
                  Wi ← t + Wi-4      // t adalah word sementara
            end
      end
end.
```

Proses transformasi pada ekspansi kunci terdiri dari beberapa tahapan sebagai berikut.

1) *RotWord (Rotate Word)*

   Merupakan sebuah rutin yang mirip dengan transformasi ShiftRows, terdiri dari sebuah word yang dinyatakan dalam array 4 byte, dan menggeser tiap byte ke kiri.

2) *SubWord (Substitute Word)*

   Rutin ini mirip dengan transformasi SubByte yang dijalankan pada 4 byte. Rutin mengambil tiap byte pada word dan mensubstitusinya dengan byte lainnya.

3) *RCon (Round Constant)*

   Tiap konstanta putaran adalah 4 byte dimana 3 byte sebelah kanan selalu 0. Berikut ini adalah nilai konstanta putaran untuk kunci 128 bit.

Tabel 2.3 Konstanta Putaran (RCon)

Putaran ke-	Konstanta Putaran (RCon)	Putaran ke-	Konstanta Putaran (RCon)
1	(**01** 00 00 00)	6	(**20** 00 00 00)
2	(**02** 00 00 00)	7	(**41** 00 00 00)
3	(**04** 00 00 00)	8	(**81** 00 00 00)
4	(**08** 00 00 00)	9	(**1B** 00 00 00)
5	(**10** 00 00 00)	10	(**36** 00 00 00)

2.3.8 DCPcrypt *Cryptographic Component Library* v2 Beta 3

Dalam kompetisi yang diadakan oleh NIST, salah satu persyaratan yang diajukan adalah persyaratan lisensi, yaitu siapa saja yang akan menjadi pemenang dari kompetisi, algoritmanya akan dipublikasikan oleh NIST di internet secara gratis. Maka kini algoritma Rijndael yang terpilih sebagai pemenang dipublikasikan dan dapat digunakan secara bebas.

Pada tugas akhir ini proses enkripsi dengan algoritma Rijndael akan memanfaatkan DCPcrypt, yaitu kumpulan dari komponen kriptografi untuk bahasa pemrograman Borland Delphi, C++ Builder dan Kylix yang dikembangkan oleh *David Barton*. Untuk bahasa pemrograman Borland Delphi, DCPCrypt dapat dijalankan pada versi 4, 5, 6 dan 7. DCPCrypt merupakan program *open source* di bawah lisensi MIT dan telah mendapat sertifikasi dari OSI.

2.4 Metode *Parity Coding*

Metode Parity Coding merupakan salah satu metode Steganography yang cukup populer, setelah metode Low Bit Encoding, tetapi memiliki kelebihan dari segi keamanan yang ditawarkan. Dengan menggunakan cara kerja *Parity*, yaitu dengan cara menghitung jumlah bit 1 dalam suatu deret bit. Dalam *Even Parity*, apabila jumlah bit 1 genap, maka akan menghasilkan angka 0, sebaliknya apabila jumlah bit 1 ganjil, maka akan dihasilkan angka 1. Lain halnya dalam *Odd Parity*, apabila jumlah bit 1 ganjil, maka akan menghasilkan angka 1, sebaliknya apabila jumlah bit 1 genap, maka akan dihasilkan angka 0 dari proses tersebut.

Hasil dari *parity* yang dilakukan apabila tidak sama dengan data yang akan

disimpan, maka akan dilakukan perubahan dalam LSB (Least Significant Bit), yaitu dengan cara mengubah 1 bit terakhir dari 1 byte data audio. Hal ini dilakukan dengan harapan meminimalisir perubahan yang terjadi, sehingga output suara yang dihasilkan tidak terlalu memiliki banyak perbedaan dengan file aslinya.

Contoh: pada file audio setelah di ubah menjadi file biner dan menyimpan huruf A yang binernya 1000 0011, maka proses yang terjadi akan menjadi seperti pada tabel 2.4 berikut:

Tabel 2.4 *Parity Coding*

Bit Awal	Hasil *parity*	A(1000 0011)	Bit akhir
1000 1100	1	1	1000 1100
0001 1011	0	0	0001 1011
1010 0100	1	0	1010 0101
0101 0101	0	0	0101 0101
0011 0011	0	0	0011 0011
1111 0000	0	0	1111 0000
0001 1000	0	1	0001 1001
1100 0111	1	1	1100 0111

2.5 *Pseudorandom Number Generator*

Pseudorandom merupakan suatu algoritma yang digunakan untuk menghasilkan nilai *random.* Algoritma tersebut sebenarnya menghasilkan nilai

random dengan perhitungan khusus, jadi tidak benar-benar nilai *random* yang muncul karena mempunyai distribusi *uniform* yang berarti hasil *random* yang muncul sama apabila menerima inputan yang sama saat melakukan proses *pseudorandom*. *Pseudorandom* ini biasanya digunakan pada enkripsi RC4. Untuk menghasilkan nilai yang seakan-akan *random* berdasarkan dari inputan yang diisi pada *pseudorandom*, maka dalam merancang algoritma *pseudorandom* digabungkan dengan algoritma untuk membuat *subtitution box* (S-*Box*) yang pengisian nilai dalam S-*Box* berdasarkan *key* yang diisi.

Proses inisialisasi S-*Box* awal, langkah pertama yang harus dilakukan yaitu membuat S-*Box* yang diisi secara berurutan dari 0 sampai 255. Kemudian isi S-*BoxKey* secara berurutan dari dan perulangan dari nilai *key char* 1 sampai nilai *key char* terakhir. Langkah selanjutnya yaitu :

```
For i = 0 to 255
    j = (j+S-Box[i]+S-BoxKey[i] mod 256
    swap S-Box[i] dan S-Box[j]
```

Setelah selesai menginialisasi S-*Box* maka dapat *generate* nilai *random* sebanyak jumlah yang dibutuhkan dengan menggunakan algoritma *pseudorandom.* Langkah pertama dalam membuat *pseudorandom* yaitu menginialisasi i = 0 dan j = 0. Dan langkah berikutnya yaitu :

```
For n = 1 to (banyak random yang dibutuhkan)
    i = (i+1) mod 256
    j = (j+S-Box[i]) mod 256
     swap S-Box[i] dan S-Box[j]
       t = (S-Box[i] + S-Box[j]) mod 25
       random = S-Box[t]
```

2.6 MPEG Audio Layer 3 (MP3)

Asal-usul MP3 dimulai dari penelitian IIS-FHG (Institut Integriette Schaltungen Fraunhofer Gesellschaft), sebuah lembaga penelitian terapan di Munich, Jerman dalam penelitian *coding audio perceptual*. Penelitian mengenai pemampatan berkas audio ini dipimpin langsung oleh Karl Heinz Brandenburg, dan menghasilkan sebuah algoritma MPEG-1 Layer 3 yang kemudian dikenal sebagai MP3. Penelitian tersebut menghasilkan suatu algoritma yang menjadi standar sebagai ISO-MPEG Audio Layer-3 (MP3), yang merupakan berkas dengan teknik *lossy compression*.

Dalam dunia kompresi digital dikenal dengan dua macam metode yakni *lossless compression* dan *lossy compression*. Pada *lossless compression* terjadi penahanan (tidak menghilangkan) semua informasi yang ada dan dekodernya mampu merekonstruksi sinyal yang telah terkompresi berubah menjadi bentuk berkas asli. Kompresi *lossless* menghasilkan kompresi data yang hampir mirip dengan kualitas audio aslinya, hasil kompresi ini juga menghasilkan ukuran file yang lebih kecil (yakni sekitar 50-75% saja) dan tentunya masih bisa

dikembalikan ke bentuk aslinya tanpa menghilangkan sesuatu apapun informasi yang terkandung didalamnya. Format-format lossless compression bisa dalam bentuk FLAC (*Free Lossless Audio Codec*) dan Monkey Audio (*APE*).

Pada metode dengan *lossy compression*, dimana menghasilkan kompresi dengan ukuran yang jauh lebih kecil dibandingkan berkas asli. Kompresi jenis *lossy* tentunya lebih terkenal dalam dunia portable audio karena ukuran filenya jauh lebih kecil dibanding dengan penggunaan jenis *lossless compression*. Hal itu terjadi karena, pemampatan data dilakukan dengan cara mengurangi atau menghilangkan informasi-informasi yang dianggap tidak penting atau yang kurang *audible* bagi telinga. Hasil kompresi dengan teknik *lossy* tidak dapat dibalikkan ke dalam bentuk semula. Sebagai contoh berkas audio yang menggunakan teknik *lossy* adalah MP3. Sebuah lagu (WAV) dengan durasi 3 menit dapat menyita alokasi *hard-disk* sebesar 30 MB. Lagu yang sama dengan format MP3 hanya membutuhkan ruang sebesar 3 MB dengan penurunan kualitas suara yang minimum.

Dalam upaya menghasilkan MP3, Brandenburg menganalisis bagaimana otak dan telinga manusia menangkap suara. Teknik yang digunakan berhasil memanipulasi telinga dengan membuang bagian yang kurang penting pada suatu file musik. Sebagai contoh, apabila terdapat dua nada yang mirip, atau apabila nada tinggi dan rendah muncul secara bersamaan, otak hanya akan memproses salah satunya. Sehingga algoritma MP3 akan memilih sinyal yang lebih penting dan membuang sisanya. Hasilnya adalah file MP3 yang memiliki ukuran file audio orisinal hingga 10 kali lebih kecil. Berkas ini sering digunakan di internet

karena ukurannya yang cukup kecil dibandingkan ukuran berkas audio yang tidak terkompresi. Teknologi ini kemudian distandarisasi pada tahun 1991.

Dalam hal audio steganografi, ada beberapa keuntungan dalam menggunakan MP3. Pertama, keberadaannya yang bersifat kosmopolit dan populer sebagai sarana hiburan di kalangan pengguna komputer. Karena keberadaannya yang sangat umum, diharapkan dapat meminimalisir kecurigaan akan adanya pesan rahasia di dalamnya. Selain itu, MP3 dapat didistribusikan dengan mudah dan hampir tanpa biaya walaupun sebenarnya hak paten dari MP3 telah dimiliki dan penyebaran MP3 seharusnya dikenakan biaya. Namun pemilik hak paten dari MP3 telah memberikan pernyataan bahwa penggunaan MP3 untuk perorangan tidak dikenai biaya. Keuntungan lainnya adalah kemudahan dalam mengakses MP3, dimana banyak *software* yang dapat menghasilkan berkas MP3.

Kesuksesan MP3 dimulai pada 1998, ketika WinAmp, sebuah mesin pemutar MP3 yang dibuat oleh sepasang mahasiswa bernama Justin Frankel dan Dmitry Boldyrev, ditawarkan secara cuma-cuma di internet. Dalam waktu singkat, pengguna musik di seluruh dunia terhubung dalam satu jaringan pusat bernama MP3, dan saling menawarkan musik-musik yang memiliki hak cipta secara gratis. Dalam rentang waktu yang tidak terlalu lama, banyak *programmer* lain yang tidak mau ketinggalan, mereka menciptakan berbagai perangkat lunak pendukung untuk para pengguna MP3. *Encoder, ripper,* dan *player* terbaru dirilis setiap minggunya, dan pertumbuhannya bergerak semakin kencang. Mesin-mesin pencari pun membuat proses pencarian file MP3 tertentu yang dikehendaki menjadi semakin

cepat. Selain itu, *player portabl*e seperti Rio dan iPod membuat MP3 dapat didengarkan sambil berjalan.

2.6.1 Teknik Kompresi MP3

Teknik kompresi MP3 memanfaatkan keterbatasan telinga manusia dalam mengenali berbagai macam suara yang ada disekitarnya. Suara-suara yang tak dapat didengar telinga manusia dihilangkan/diabaikan. Dengan metode tersebut maka dapat dihasilkan suatu berkas audio terkompresi dengan kualitas suara yang tidak berbeda jauh dari aslinya. Hal ini disebabkan oleh kelemahan dari sistem pendengaran manusia yang dapat dieksploitasi. Beberapa kelemahan manusia dapat dimanfaatkan dalam mengolah kompresi file audio ke dalam bentuk format MP3. Beberapa model yang ada antara lain sebagai berikut (Rassol, 2002).

1. Model Psikoakustik.

   Salah satu karakteristik pendengaran manusia adalah memiliki batas frekuensi 20 Hz s.d. 20 kHz, dimana suara yang frekuensinya berada di luar ambang batas ini tidak dapat didengar oleh manusia, sehingga suara seperti itu tidak perlu dikodekan.

2. *Auditory Masking*

   Manusia tidak mampu mendengarkan suara pada frekuensi tertentu dengan amplitudo tertentu jika pada frekuensi di dekatnya terdapat suara dengan amplitudo yang jauh lebih tinggi. Amplitudo dengan nilai yang lebih tinggi ini akan mengalahkan frekuensi dengan amplitudo yang jauh lebih rendah.

3. *Critical Band*

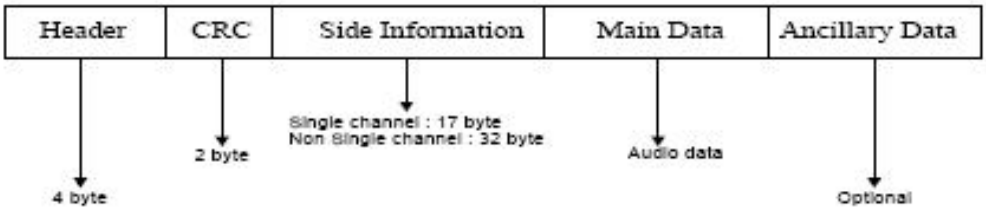
   Merupakan daerah frekuensi tertentu dimana pendengaran manusia lebih peka pada frekuensi-frekuensi rendah, sehingga alokasi bit dan alokasi *sub-band* pada filter *critical band* lebih banyak dibandingkan frekuensi lebih tinggi. Dengan mengalokasikan bit pada filter *critical band,* manusia bisa mendengar audio pada frekuensi rendah.

4. *Joint Stereo*

   Terkadang *dual channel stereo* mengirimkan informasi yang sama. Dengan menggunakan *joint stereo*, informasi yang sama ini cukup ditempatkan dalam salah satu *channel* saja dan ditambah dengan informasi tertentu. Dengan teknik ini *bitrate* dapat diperkecil.

2.6.2 Struktur Berkas MP3

Berkas MP3 terdiri dari bagian-bagian kecil yang disebut *frame*. Setiap *frame* memiliki waktu yang konstan yaitu 0,026 detik (26 milidetik). Tapi ukuran dari sebuah *frame* (dalam byte) bervariasi bergantung pada *bitrate*. Sebuah *frame* MP3 secara umum terdiri atas 5 bagian, yaitu *Header*, CRC, *Side Information*, *Main Data* dan *Ancillary Data* (Rassol, 2002).



Gambar 2.11 Struktur *Frame* MP3

1. ***Header***

Header adalah bagian dari sebuah *frame* dengan panjang 4 byte (32 bit) yang berisi bit-bit sinkronisasi dan deskripsi tentang *frame* tersebut. Bagian *header* yang sangat menentukan dalam steganografi ini adalah pada bagian *Error Protection* (*CRC*). Karena terdapat atau tidaknya bit *CRC* ini sangat mempengaruhi panjang *frame* sebuah MP3 dan juga mempengaruhi dalam penentuan jumlah kapasitas yang dapat ditampung MP3 tersebut. Anatomi dari *header* sebuah *frame* MP3 adalah sebagai berikut.



Gambar 2.12 Struktur *Header* pada *Frame* MP3

Keterangan tentang struktur *header* pada *frame* MP3 dapat dilihat pada tabel berikut.

Tabel 2.5 Keterangan Struktur *Header* pada *Frame* MP3

Posisi	Keterangan Fungsi	Panjang (Bit)
Sync	*Frame* sinkronisasi	11
ID	Versi MPEG audio (MPEG-1, MPEG-2, dll)	2
Layer	MPEG layer (Layer I, II, III dll)	2
Prot. bit	Bit Proteksi, jika bit ini diset maka CRC akan digunakan.	1
Bit rate	Indeks *Bitrate*	4
Frequency	Frekuensi *sampling rate*	2
Posisi	**Keterangan Fungsi**	**Panjang (Bit)**
Pad. bit	Bit p*adding*	1
Priv. bit	*Private* bit	1
Mode	Mode *channel* (*stereo, joint stereo,* dll)	2
Mode Extension	Mode *extension*	2
Copy	Hak Cipta (*Copyright*)	1
Home	Original	1
Emphasis	Emphasis	2

2. *CRC* (*Cyclic Redundancy Check*)

CRC adalah bagian dari sebuah *frame* dengan panjang 2 byte. Bagian ini hanya akan ada jika bit proteksi pada *header* diset dan memungkinkan untuk memeriksa data-data sensitif. Bit CRC ini oleh sebuah *frame* digunakan untuk memeriksa data-data sensitif pada *header* dan *Side Information*. Jika nilai-nilai pada *CRC* tersebut mempunyai kesalahan maka *frame* tersebut oleh MP3 *player* akan dinyatakan *corrupt*, dan akan digantikan dengan *frame* sebelumnya. CRC adalah algoritma untuk memastikan integritas data dan memeriksa kesalahan pada suatu data yang akan ditransmisikan atau disimpan. CRC bekerja secara sederhana, yakni dengan menggunakan

perhitungan matematika terhadap sebuah bilangan yang disebut sebagai *Checksum*, yang dibuat berdasarkan total bit yang hendak ditransmisikan atau yang hendak disimpan. *Checksum* akan dihitung terhadap setiap frame yang hendak ditransmisikan dan ditambahkan ke dalam frame tersebut sebagai informasi dalam *header*. Penerima *frame* tersebut akan menghitung kembali apakah *frame* yang ia terima benar-benar tanpa kerusakan, dengan membandingkan nilai *frame* yang dihitung dengan nilai *frame* yang terdapat dalam *header frame*. Jika dua nilai tersebut berbeda, maka *frame* tersebut telah berubah.

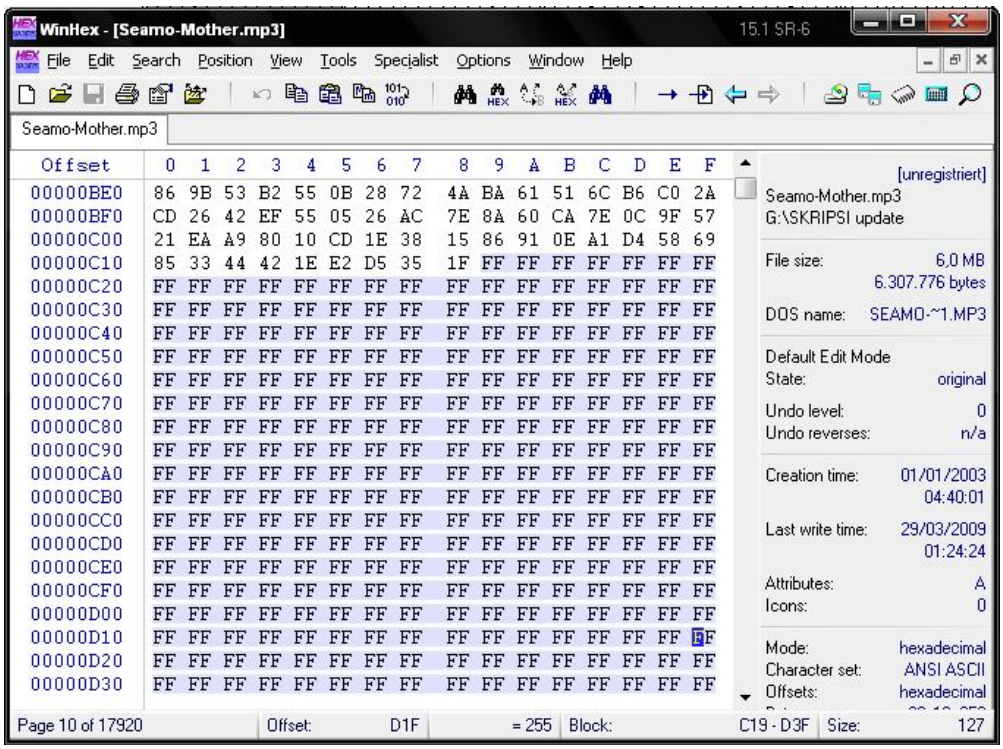
3. *Side information*

Side Information adalah bagian dari sebuah *frame* dengan panjang 17 byte untuk *mode single channel* dan 32 byte untuk mode yang lain. *Side Information* mengandung informasi yang dibutuhkan untuk melakukan dekoding *Main Data*. Untuk mendapatkan kualitas audio yang normal, maka *Side Information* ini tidak boleh rusak dalam proses steganografi.

4. *Main data*

Main Data adalah bagian dimana data audio dari sebuah file MP3 berada dan mempunyai panjang yang bervariasi. Disini terdapat *Huffman code bits*, informasi untuk dekoding *Huffman code bits* ini terdapat pada bagian *Side Information*. Bagian *frame* yang akan digunakan dalam steganografi ini adalah dengan mengganti isi dari *Main Data* tersebut. Teknik kompresi MP3 juga menerapkan metode *Huffman coding* untuk mendapatkan hasil kompresi yang lebih baik. Jika sebuah *frame* MP3 mempunyai suara/audio yang sama

persis (biasanya terdapat pada *frame* yang tidak mempunyai nada/suara apapun) maka isi dari *Main Data*nya cukup ditulis dalam satu macam nilai saja dan informasi selebihnya akan disimpan pada *Side Information*. Nilai homogen yang dimaksud disini dapat berupa nilai heksadesimal berapapun.



Gambar 2.13 Contoh Byte-Byte Homogen Pada MP3

Jika seluruh byte dalam satu *frame* MP3 adalah homogen, artinya *frame* tersebut dapat digunakan sebagai tempat penyisipan data. Pada *file* MP3 yang tidak menggunakan CRC, umumnya letak *Main Data* dimulai setelah byte ke 36 hingga *frame* selanjutnya, sehingga metode penggantian byte pada steganografi ini akan dilakukan mulai dari byte ke 36 tersebut hingga menyentuh *frame* selanjutnya. Diharapkan dengan mengganti isi dari *Main Data* yang homogen tersebut, suara yang terdapat pada *frame* tersebut tidak

akan mengalami perubahan. Karena sebuah *frame* hanya menyumbangkan waktu selama 0,026 detik saja, maka perubahan yang terjadi juga akan sangat kecil dan hampir tidak dapat terdeteksi oleh telinga manusia.

5. ***Ancilary data***

   Ancillary Data merupakan bagian opsional, tidak banyak terdapat informasi pada bagian ini dan pada umumnya sebuah *frame* tidak mempunyai *Ancillary Data*.

2.7 Model Analisis Perangkat Lunak.

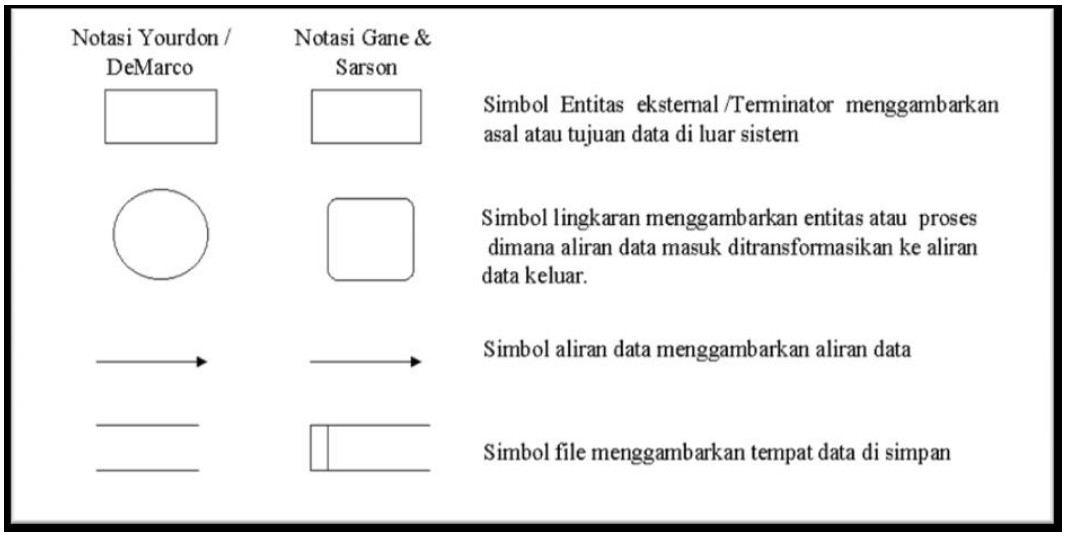
Dalam pengembangan perangkat lunak, dibutuhkan pembuatan model analisis perangkat lunak agar kita dapat melakukan koreksi atau membuat model yang lebih baik dan memenuhi kriteria yang diinginkan tanpa resiko berarti dibandingkan jika membuat sistem secara langsung. Dalam merancang sebuah perangkat lunak, mungkin saja diperlukan lebih dari satu cara pemodelan, dan setiap model difokuskan pada aspek tertentu saja yang sifatnya terbatas. Karakteristik pemodelan yang sebaiknya digunakan secara garis besar adalah sebagai berikut:

1. Dibuat dalam bentuk grafis dan tambahan keterangan secara tekstual. Dengan grafis, lebih banyak yang dapat dijelaskan dibandingkan pemodelan dengan narasi. Grafis juga mempermudah identifikasi komponen dan interaksi antar komponen.
2. Dapat diamati dengan pola *top-down* dan *partitioned*. Hal ini sangat penting bagi sistem yang berukuran besar. Maka perangkat pemodelan

yang digunakan harus mampu difokuskan pada bagian-bagian terpisah dari sistem yang secara bersama-sama merupakan sistem yang terintegrasi. Dapat dianalogikan seperti peta yang umumnya dipetakan dari bagian terbesar sampai ke bagian detail.

3. Dapat mempresentasikan tingkah laku sistem dengan cara yang transparan. Sebagai model yang baik, persyaratan yang lain adalah mudah dibaca dan mudah dipahami.

Pemodelan fungsional adalah proses untuk mendeskripsikan seluruh fungsi yang terlibat di dalam perangkat lunak. Piranti yang digunakan untuk menjelaskan pemodelan fungsional ini adalah Diagram Konteks (*Context Diagram*) dan Diagram Aliran Data (*Data Flow Diagram)*. Pemodelan ini diperlukan untuk memperlihatkan proses-proses yang dimiliki aplikasi dan bagaimana proses tersebut mentransformasikan data menjadi informasi. Berikut ini adalah notasi yang digunakan dalam pemodelan *Data Flow Diagram* (DFD).



Gambar 2.14 Notasi pada *Data Flow Diagram*

Diagram Konteks (*Context Diagram*) merupakan diagram yang memperlihatkan sistem sebagai suatu proses yang berinteraksi dengan lingkungannya. Ada pihak luar yang memasukkan informasi ke dalam sistem dan ada yang menerima informasi dari sistem. Pihak luar dapat berupa sistem lain, perangkat keras, individu atau organisasi. Level berikutnya merupakan Diagram Aliran Data (*Data Flow Diagram)* yang merupakan pengembangan dari diagram konteks. DFD memberikan gambaran tentang aliran data yang lebih terperinci terhadap tiap-tiap entitas yang terdapat pada level sebelumnya. Pengembangan DFD dilakukan untuk memperlihatkan aliran data pada internal sistem atau perangkat lunak. DFD juga menggambarkan fungsi-fungsi yang dimiliki sistem atau perangkat lunak tersebut.