BAB II
TINJAUAN PUSTAKA

2.1 Steganografi

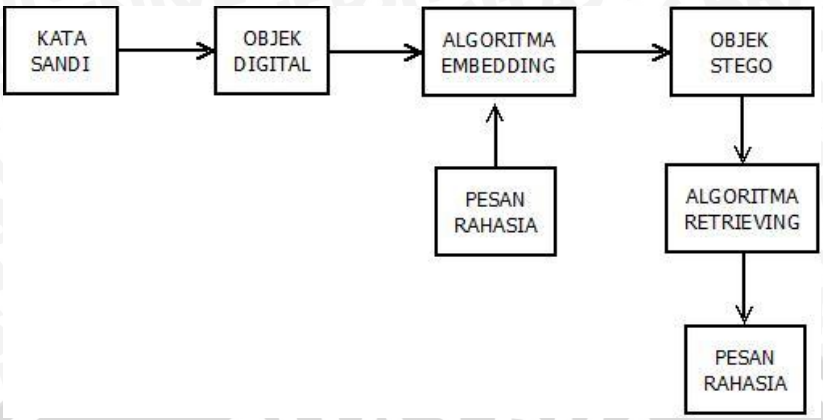
2.1.1 Pengertian Steganografi

Steganografi berasal bahasa Yunani *steganos* (*hidden*) dan *graphein* (*writing*). Jadi, steganografi berarti *hidden writing* atau tulisan tersembuyi. Steganografi adalah seni dan ilmu menyembunyikan pesan ke dalam sebuah media dengan suatu cara sehingga selain pengirim dan penerima, tidak ada seorangpun yang mengetahui atau menyadari bahwa sebenarnya ada suatu pesan rahasia. Pada steganografi modern, arti steganografi berkembang menjadi penyembunyian informasi pada sebuah media berkas digital [FIR-11].

Steganografi membutuhkan dua properti, yaitu media penampung dan pesan rahasia yang akan disembunyikan. Steganografi digital menggunakan media digital sebagai media penampung, misalnya citra, suara (audio), teks, atau *video*. Data rahasia yang disembunyikan juga dapat berupa citra, suara, teks, atau pun *video* [MUN-04].

Steganografi digunakan untuk menyembunyikan suatu data atau informasi ke dalam sebuah media sehingga sulit dideteksi keberadaan data atau informasi tersebut karena hasil dari penyembunyian tersebut tidak berbeda dengan sumbernya. Hal pertama yang perlu dilakukan untuk melakukan steganografi adalah memilih media penampung pesan. Data atau informasi rahasia disembunyikan atau disembunyikan ke dalam media penampung tersebut.

Untuk menyisipkan informasi atau data rahasia ke dalam obyek digital diperlukan suatu algoritma yang disebut dengan algoritma *embedding*. Algoritma tersebut dapat memodifikasi obyek digital sehingga menghasilkan obyek digital baru yang berisi informasi tersembunyi. Dalam proses modifikasi, perubahan yang terjadi antara obyek digital (media asli) dengan obyek digital yang mengandung pesan rahasia tidak boleh terlalu terlihat perbedaannya. Sedangkan untuk mengungkapkan kembali pesan rahasia yang terdapat pada suatu media digital diperlukan algoritma *retrieving* [HAP-09]. Gambar 2.1 menunjukkan gambaran umum proses steganografi.



Gambar 2.1 Proses Steganografi [HAP-09]

2.1.2 Kriteria Steganografi

Dalam menyembunyikan pesan, steganografi harus memenuhi beberapa kriteria sebagai berikut [MUN-04] :

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.

2. *Robustness*

   Data yang disembunyikan harus tahan terhadap berbagai operasi manipulasi yang dilakukan pada media penampung. Bila pada media penampung dilakukan operasi manipulasi, maka data yang disembunyikan seharusnya tidak rusak.

3. *Recovery*

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

2.1.3 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. Audio steganografi pada berkas audio dapat dilakukan dengan berbagai teknik. Beberapa teknik yang dapat digunakan antara lain [KAL-10] :

1. Penggantian bit

   Metode penggantian bit lazim digunakan dalam teknik digital steganografi yaitu mengganti bagian tertentu dari bit-bit datanya dengan data rahasia yang disembunyikan. Dengan metode ini keuntungan yang didapatkan adalah ukuran pesan yang disembunyikan relatif besar, namun berdampak pada hasil audio yang berkualitas kurang dengan banyaknya derau.

2. *Phase coding*

   Teori yang digunakan adalah dengan mensubstitusi awal fasa dari tiap awal segmen dengan fasa yang telah 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 penggantian bit namun memilki kerumitan dalam realisasinya.

3. *Spread spectrum*

   Pada metode ini pesan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan. Maka dari itu akan sangat sulit bagi yang akan mencoba memecahkannya kecuali memiliki akses terhadap data tersebut atau dapat merekonstruksi sinyal acak yang digunakan untuk menyebarkan pesan pada *range* frekuensi.

4. *Echo hiding*

   Teknik ini menyamarkan pesan ke dalam sinyal yang membentuk echo, kemudian pesan disembunyikan dengan menvariasikan tiga parameter dalam echo yaitu besar amplitude awal, tingkat penurunan antenuasi 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.

2.2 Metode *Parity Coding*

2.2.1 *Parity* Bit

Parity bit adalah sebuah bit yang ditambahkan untuk memastikan jumlah bit bernilai 1 pada suatu set bit selalu *even* atau *odd*. *Parity* bit merupakan parameter opsional yang digunakan pada *serial communications* untuk menentukan apakah set bit yang dikirimkan tepat adanya [HER-10].

Parity bit mempunyai spesifikasi salah satu dari beberapa kemungkinan antara lain :

1. *None*

   Pada kondisi ini sistem tidak boleh membentuk *parity* bit pada data karakter yang dikirimkan dengan kata lain tidak ada pengecekan terhadap *parity* bit dari data yang dikirim.

2. *Even*

   Pada kondisi ini jumlah keseluruhan dari bit '1' pada karakter tunggal haruslah genap (*even*). Jika tidak, maka *parity* bit yang ditambahkan haruslah sebuah biner '1' untuk memastikan jumlahnya menjadi genap. Sebagai contoh, jika suatu kata dengan bentuk biner "1100001" ditransmit dalam kondisi *even parity*, maka akan didapati jumlah dari biner '1' berjumlah 3 atau ganjil. Untuk menjaga jumlah biner '1' tetap genap, maka dibuat *parity* bit '1'. Pada kasus lain jika suatu kata dengan bentuk biner "1000001" ditransmisikan pada kondisi yang sama (*even parity*), maka *parity* bit diset menjadi '0' sehingga jumlah biner '1' tetap genap.

3. *Odd*

   Dioperasikan dalam konsep yang sama dengan *even parity* namun dalam kondisi jumlah biner '1' harus ganjil

4. *Space*

   Menspesifikan biner *parity* bit selalu '0'. Lingkup lain yang memakai *space parity* adalah *bit filling*, yang diturunkan dari kegunaannya sebagai sebuah "*filler*" untuk data 7 bit yang ditransmisikan pada sistem yang hanya menerima data 8 bit.

Parity bit lazimnya digunakan dalam *error detection*, yaitu pengecekan suatu transmisi data karakter apakah sesuai dengan aslinya atau tidak. Misal, suatu

bit data dengan nilai "1001" ditransmisikan, dengan *parity* bit diikuti pada bagian paling kanan, dan simbol ⊕ mendenotasikan XOR *gate*. Maka :

1. Transmisi pada kondisi *even parity*

   A mentransmisikan : 1001

   A menghitung nilai *parity* bit : $1 \oplus 0 \oplus 0 \oplus 1 = 0$

   A menambahkan *parity* dan mengirimkan : 10010

   B menerima : 10010

   B menghitung nilai *parity* bit : $1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$

   B melaporkan bahwa transmisi benar setelah menganalisa *even parity*.

2. Transmisi pada kondisi *odd parity*

   A mentransmisikan : 1001

   A menghitung nilai *parity* bit : $\sim (1 \oplus 0 \oplus 0 \oplus 1) = 1$

   A menambahkan *parity* bit dan mengirimkan : 10011

   B menerima : 10011

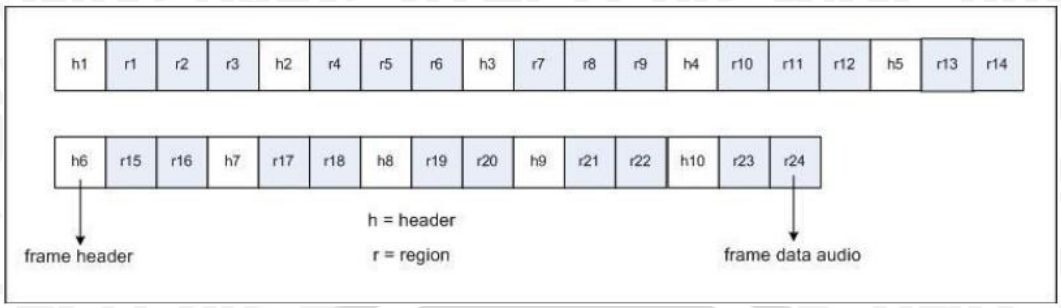
   B menghitung *parity* keseluruhan : $1 \oplus 0 \oplus 0 \oplus 1 \oplus 1 = 0$

   B melaporkan bahwa transmisi benar setelah menganalisa dalam kondisi *odd parity*.

2.2.2 *Parity Coding*

Steganografi audio mengenal sebuah metode yang dinamakan *parity coding*. Metode *parity coding* merupakan salah satu metode steganografi yang cukup populer, setelah metode *low bit encoding*, tetapi memilki kelebihan dari segi keamanan yang ditawarkan. *Parity coding* menggunakan cara kerja *parity* bit, yaitu dengan cara menghitung jumlah bit 1 dalam suatu deret bit.

Pada teknik *parity coding*, sinyal dari berkas audio sudah diencode dibagi menjadi beberapa region terpisah yang mempunyai ukuran statis. Banyaknya region ditentukan oleh panjang isi pesan rahasia. Gambar 2.2 menunjukkan contoh pembagian region pada berkas audio MP3 dengan panjang pesan 24 bit.



Gambar 2.2 Pembagian Region Pada Pesan 24 bit [KAL-10]

Parity bit dari setiap region akan dihitung terlebih dahulu untuk disimpan nilainya.

Dalam kondisi *even parity*, apabila jumlah bit 1 genap, maka akan menghasilkan *parity* bit 0, sebaliknya apabila jumlah bit 1 ganjil, maka akan menghasilkan *parity* bit 1. Lain halnya dalam *odd parity*, apabila jumlah bit 1 ganjil, maka akan akan menghasilkan *parity* bit 0, sebaliknya apabila jumlah bit 1 genap, maka akan dihasilkan *parity* bit 1.

Bit dari pesan rahasia akan disembunyikan secara merata ke dalam region yang ada. Jika bit yang akan dimasukkan ke dalam region nilainya berbeda dengan *parity* bit dari region tersebut, maka susunan dari bit-bit LSB harus harus diubah sedemikian rupa sehingga *parity* bit region nilainya sama dengan bit pesan rahasia yang akan disembunyikan namun jika nilainya sama region tidak perlu diubah. Hal tersebut dilakukan dengan harapan meminimalisir perubahan yang terjadi, sehingga *output* audio yang dihasilkan tidak terlalu memiliki banyak perbedaan dengan berkas aslinya [KAL-10].

Untuk pengambilan kembali pesan rahasia, dibutuhkan informasi panjang region sehingga bisa mengetahui banyaknya pesan yang disembunyikan pada berkas audio. Oleh karena itu dibutuhkan informasi spesifikasi penyembunyian yang disimpan pada byte pertama setiap *frame* media steganografi. Setelah panjang region diketahui, proses dilanjutkan ke pembagian region yang kemudian dilanjutkan dengan perhitungan *parity* bit region dimana nilai *parity* bit inilah yang disusun menjadi isi pesan.

Tabel 2.1 menunjukkan ilustrasi menyisipkan karakter "A" dengan nilai biner 100000111 pada beberapa set bit menggunakan metode *parity coding* dengan kondisi *even parity*.

Tabel 2.1 Ilustrasi *parity coding*

Bit Awal	Hasil *Parity*	"A" (10000011)	Bit Akhir
10001100	1	1	10001100
00011011	0	0	00011011
10100100	1	0	10100101
01010101	0	0	01010101
00110011	0	0	00110011
11110000	0	0	11110000
00011000	0	1	00011001
11000111	1	1	11000111

Secara umum cara kerja penyisipan pesan rahasia adalah mengganti nilai *parity* bit pada tiap region dari berkas audio. Berikut ini merupakan langkah-langkah penyisipan pesan pada metode *parity coding* dalam kondisi *even parity* [SAR-06].

1. Sinyal berkas audio sebagai media penampung (*carrier*) dibagi menjadi beberapa region sebanyak $l(m)$ sesuai dengan banyak bit pesan (m_i) yang akan disembunyikan dengan ($1 \leq i \leq (m)$).
2. Pada setiap region dikodekan menjadi bilangan biner.
3. Nilai *parity* bit setiap region dihitung nilainya.
4. Setiap bit dari pesan rahasia (m_i) disamakan dengan nilai *parity* bit setiap region. Jika nilai *parity* bit tidak sama dengan bit pesan rahasia, maka *parity* bit disesuaikan dengan bit dari pesan rahasia.
5. Sinyal berkas audio yang masih berbentuk bilangan biner diubah kembali menjadi bentuk amplitudo.

Cara kerja umum pengambilan pesan pada metode *parity coding* adalah dengan cara mengurutkan nilai *parity* bit pada setiap region. Berikut ini merupakan langkah-langkah pengambilan pesan rahasia.

1. Berkas audio dibagi menjadi beberapa region.
2. Pada setiap region dikodekan menjadi bilangan biner.
3. Nilai *parity* bit pada setiap region dicari nilainya.
4. Urut nilai *parity* bit sehingga menghasilkan pesan rahasia.

2.2.3 LSB (*Least Significant Bit*)

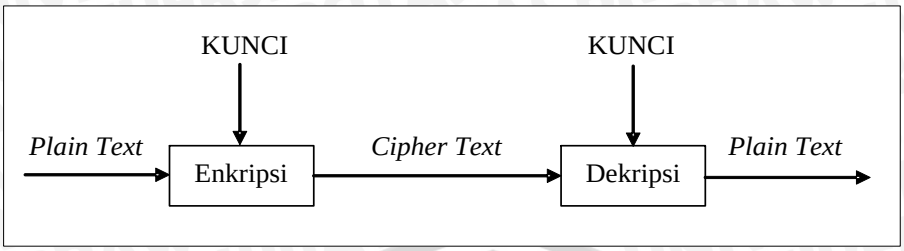
Pada susunan bit di sebuah byte dimana 1 byte bernilai 8 bit, terdapat bit yang paling berarti (*Most Significant Bit* atau MSB) dan bit yang paling kurang berarti (*Least Significant Bit* atau LSB). Misalnya pada byte 11010010, bit 1 yang pertama (digarisbawahi) adalah MSB dan bit 0 yang terakhir (digarisbawahi) adalah LSB. LSB dikatakan paling kurang berarti karena apabila terjadi modifikasi pada bit LSB, maka hanya mengubah nilai byte tersebut satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya [MUN-04].

2.3 Kriptografi

2.3.1 Pengertian Kriptografi

Kata kriptografi berasal dari bahasa Yunani. Dalam bahasa Yunani kriptografi terdiri dari dua buah kata yaitu *cryptos* dan *graphia*. Kata *cryptos* berarti rahasia sedangkan *graphia* berarti tulisan. Berarti secara umum makna dari kata kriptografi adalah tulisan rahasia. Arti sebenarnya dari kriptografi adalah ilmu yang mempelajari tentang bagaimana menjaga kerahasiaan suatu pesan, agar isi pesan yang disampaikan aman sampai ke penerima pesan [ARI-08].

Secara umum, kriptografi merupakan teknik pengamanan informasi yang dilakukan dengan cara mengubah informasi awal (*plaintext*) dengan suatu kunci tertentu menggunakan suatu metode enkripsi tertentu sehingga menghasilkan suatu informasi baru (*ciphertext*) yang tidak dapat dibaca secara langsung. *Chipertext* tersebut dapat dikembalikan menjadi informasi awal melalui proses dekripsi. Gambaran umum kriptografi dapat dilihat pada gambar 2.3.



Gambar 2.3 Gambaran Umum Proses Kriptografi

Menurut Wibowo [WIB-09], ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi, yaitu :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atau mengupas informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubstitusian data lain ke dalam data yang sebenarnya.
3. Autentifikasi, adalah berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentifikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. Non-repudiasi, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan.

2.3.2 Komponen Kriptografi

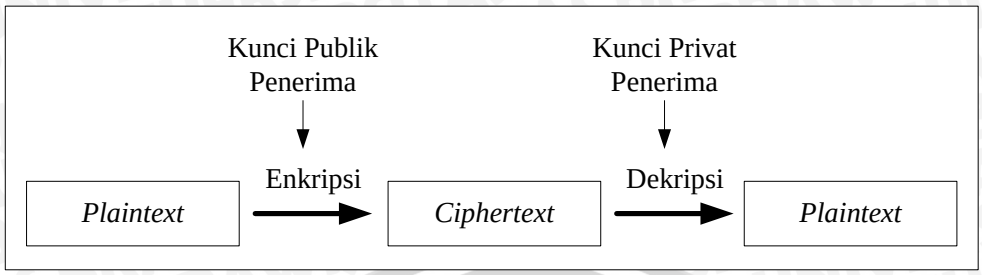
Menurut Ariyus [ARI-07], komponen kriptografi terdiri dari beberapa komponen, seperti :

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 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 dilihat orang lain atau tidak.
5. Pesan asli (*Plaintext*), merupakan teks asli yang akan diproses menggunakan algoritma kriptografi tertentu untuk menjadi *chipertext*.
6. *Ciphertext*, merupakan pesan yang telah melalui proses enkripsi yang merupakan himpunan karakter acak.
7. Kriptologi, merupakan studi tentang kriptografi dan kriptanalis.
8. Kriptanalis (*Cryptanalis*), merupakan aksi memecahkan mekanisme kriptografi dengan cara menganalisanya untuk menemukan kelemahan dari suatu algoritma kriptografi sehingga akhirnya dapa ditemukan kunci atau teks asli.

2.3.3 Kriptografi Asimetris

Kriptografi asimetris atau biasa disebut kriptografi kunci publik dirancang sedemikian sehingga kunci yang digunakan untuk enkripsi dan dekripsi berbeda. Kriptografi asimetris menggunakan dua buah kunci sehingga kunci dekripsi tidak dapat dihitung dari kunci enkripsi. Kriptografi tersebut disebut *public key* karena kunci enkripsi dapat dibuat secara *public*. Orang asing dapat menggunakan kunci enkripsi tersebut untuk mengenkripsi sebuah pesan, tetapi hanya orang tertentu dengan kunci dekripsi sepadan yang dapat mendekripsi pesan tersbut. Dalam sistem ini, kunci enkripsi sering disebut *public key* sedangkan kunci dekripsi sering disebut *private key* [KUR-04]. Gambar 2.4 memperlihatkan skema kriptografi asimetris yang membutuhkan dua buah kunci.



Gambar 2.4 Skema Kriptografi Asimetris

Kelebihan teknik kriptografi asimetri adalah :

- Hanya *private key* yang benar-benar rahasia.
- Sangat jarang untuk perlu merubah *public key* dan *private key*.

Kelemahan teknik kriptografi asimetri adalah :

- Ukuran kunci lebih besar dari pada algortima kunci simetri.
- Tidak ada jaminan bahwa *public key* benar-benar aman [MUN-04].

2.4 Metode Kriptografi RSA

2.4.1 Pengertian RSA

Algoritma RSA pertama kali ditemukan pada tahun 1977 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman, sekelompok ilmuwan dari Massachusetts Institute of Technology. Mereka kemudian mempublikasikan temuan mereka pada tahun 1978. Nama RSA sendiri diambil dari inisial nama ketiga penemunya tersebut. Pada tahun 1983, Massachusetts Institute of Technology menerima hak paten atas sebuah makalah berjudul "*Cryptography Communication System and Method*" yang mengaplikasikan penggunaan algoritma kriptografi RSA.

RSA termasuk algoritma kriptografi asimetris, oleh karena itu RSA mempunyai dua kunci, yaitu kunci publik dan kunci privat. Kunci publik boleh dikatahui oleh siapa saja dan digunakan untuk proses enkripsi. Kunci privat hanya boleh diketahui oleh pihak-pihak tertentu saja yang boleh mengetahuinya dan digunakan untuk proses dekripsi.

Keamanan dari algoritma RSA ini terletak pada sulitnya memfaktorkan bilangan yang sangat besar menjadi faktor prima. Selama pemfaktoran bilangan yang besar menjadi faktor-faktor prima belum ditemukan algoritma efektif, maka

selama itu pula kemanan algoritma RSA tetap terjamin keamanannya. Oleh karena itu faktor kemanan tersebut, RSA merupakan algoritma asimetri yang paling populer [KUR-04].

2.4.2 Konsep Matematis Pada RSA

RSA merupakan algoritma kriptografi yang menggunakan perhitungan matematis dengan pangkat besar, untuk itu dibahas konsep matematis dari RSA :

1. Aritmetika modulo

   Aritmetika modulo (*modular arithmetic*) memainkan peranan yang penting dalam komputasi integer, khususnya pada aplikasi kriptografi. Operator yang digunakan pada aritmetika modulo adalah *mod*. Operator *mod* memberikan sisa pembagian (Munir, 2003). Misal 23 dibagi 5 memberikan hasil 4 dengan sisa 3, sehingga bisa dituliskan $23\ mod\ 5 = 3$. Definisi operator *mod* dinyatakan sebagai $a\ mod\ m = r$ sedemikian sehingga :

   $$a = mq + r, \quad 0 \leq r < m \tag{2.1}$$

2. Kongruen

   Misalkan dua bilangan bulat *a* dan *b* mempunyai sisa yang sama bila dibagi dengan bilangan positif *m* (Munir, 2003). Hal ini dikatakan bahwa *a* dan *b* kongruen dalam modulo *m* dan dilambangkan sebagai :

   $$a \equiv b\ mod\ m \tag{2.2}$$

   Dapat pula dituliskan dalam hubungan :

   $$a = b + km \tag{2.3}$$

   dimana *k* bilangan bulat. Contoh :

   $38 \equiv 13\ mod\ 5$ dapat dituliskan $38 = 13 + 5 * 5$.

3. Bilangan prima

   Bilangan prima adalah bilangan bulat positif yang lebih besar dari 1 yang hanya habis dibagi oleh dirinya sendiri. Sebagai contoh, 23 adalah bilangan prima karena habis dibagi oleh 1 dan 23. Salah satu cara untuk menguji bilangan prima adalah dengan teorema bahwa bilangan bulat dikatakan prima jika tidak habis dibagi oleh bilangan-bilangan prima yang lebih kecil darinya.

Sebagai contoh, bilangan bulat 11 adalah bilangan prima, sebab tidak habis dibagi 2, 3, 5, dan 7.

4. Prima relatif

   Dua buah bilangan a dan b dikatakan relatif prima jika $FPB(a,b) = 1$. Sebagai contoh, 20 dan 3 relatif prima sebab $FPB(20,3) = 1$ tetapi 20 dan 5 tidak relatif prima karena $FPB(20,5) = 5 \neq 1$.

5. Fungsi *phi euler*

   Fungsi *phi euler* $\varphi(n)$ merupakan fungsi terhadap bilangan bulat positif *n* yang menyatakan banyaknya elemen Z_n yang mempunyai invers terhadap operasi pergandaan. Diketahui bahwa Z_n belum tentu merupakan grup terhadap operasi pergandaan. Dengan kata lain, $\varphi(n)$ adalah banyaknya elemen $\{x, 0 \leq x < n | FPB(x,n) = 1\}$ [RIY-08]. Jika $n = pq$ dengan p dan q adalah bilangan prima, maka :

$$\varphi(n) = (p-1)(q-1) \tag{2.4}$$

   Jika *n* adalah bilangan prima, maka :

$$\varphi(n) = n - 1 \tag{2.5}$$

6. Algoritma euclide

   Algoritma ini digunakan untuk mencari nilai pembagi persekutuan terbesar dari dua bilangan bulat. Algoritma ini didasarkan pada pernyataan berikut ini. Diberikan bilangan bulat r_0 dan r_1, dengan $r_0 \geq r_1$, kemudian dihitung menggunakan algortima pembagian berikut ini :

$$\begin{aligned} r_0 &= q_1 r_{1+} r_2, & 0 < r_2 < r_1 \\ r_1 &= q_2 r_{2+} r_3, & 0 < r_3 < r_2 \\ &\ldots \\ r_{n-2} &= q_{n-1} r_{n-1+} r_n, & 0 < r_n < r_{n-1} \\ r_{n-1} &= q_n r_n \end{aligned} \tag{2.6}$$

   Dari pernyataan di atas, dapat ditunjukkan bahwa :

$$FPB(r_0, r_1) = FPB(r_1, r_2) = \cdots = FPB(r_{n-1}, r_n) = FPB(r_n, 0) = r_n \tag{2.7}$$

   Pada contoh perhitungan di bawah ini dijelaskan bagaimana mencari FPB dari 81 dan 57 menggunakan algoritma euclide. Jika diketahui $r_0 = 81$ dan $r_1 = 57$, maka :

$81 = 1 * 57 + 24$

$57 = 2 * 24 + 9$

$24 = 2 * 9 + 6$

$9 = 1 * 6 + \mathbf{3}$

$6 = 2 * 3$

7. Algoritma euclide diperluas

Algoritma ini merupakan perluasan dari algoritma euclide, digunakan untuk mencari invers terhadap operasi pergandaan. Algoritma ini didasarkan pada pernyataan berikut. Diberikan bilangan bulat positif r_0 dan r_1, dimana $r_0 > r_1$:

- jika $FPB(r_0, r_1) = 1$, maka ${r_1}^{-1} \ mod \ r_0$ ada
- jika $FPB(r_0, r_1) \neq 1$, maka ${r_1}^{-1} \ mod \ r_0$ tidak ada

Dari pernyataan diatas digunakan rekurensi berikut ini :

$$t_1 = 0$$
$$t_2 = 1$$
$$t_j = t_{j-2} - q_{j-2} t_{j-1} \ mod \ r_0, \ j > 2 \qquad (2.8)$$

q_j diperoleh dari perhitungan $FPB(r_0, r_1)$ menggunakan algoritma euclide. Saat perhitungan algoritma euclide untuk mencari $FPB(r_0, r_1)$, apabila sisa bagi terakhir yang tidak nol (r_n) muncul pada langkah ke-n dimana $n \geq 1$, maka hasil dari ${r_1}^{-1} \ mod \ r_0$ akan diperoleh pada t_{n+2}. Pada contoh perhitungan di bawah ini akan dijelaskan bagaimana menghitung $15^{-1} \ mod \ 26$ menggunakan algortima euclide diperluas. Jika diketahui $r_0 = 26$ dan $r_1 = 15$, maka :

- Dihitung $FPB(26, 15)$ menggunakan algoritma euclide.

$26 = 1 * 15 + 11 \quad n = 1, \quad q_1 = 1$

$15 = 1 * 11 + 4 \quad n = 2, \quad q_2 = 1$

$11 = 2 * 4 + 3 \quad n = 3, \quad q_3 = 2$

$4 = 1 * 3 + 1 \quad n = 4, \quad q_4 = 1$

$3 = 3 * 1 + 0 \quad n = 5, \quad q_5 = 3$

Jadi, $FPB(26, 15) = 1$, dengan kata lain $15^{-1} \ mod \ 26$ ada.

- Dihitung $15^{-1} \ mod \ 26$ menggunakan persamaan 2.8.

$t_1 = 0$

$t_2 = 1$

$t_3 = t_1 - q_1 t_2 \; mod \; r_0 = 0 - 1 * 1 \; mod \; 26 = -1 \; mod \; 26 = 25$

$t_4 = t_2 - q_2 t_3 \; mod \; r_0 = 1 - 1 * 25 \; mod \; 26 = -24 \; mod \; 26 = 2$

$t_5 = t_3 - q_3 t_4 \; mod \; r_0 = 25 - 2 * 2 \; mod \; 26 = 21$

$t_6 = t_4 - q_4 t_5 \; mod \; r_0 = 2 - 1 * 21 \; mod \; 26 = -19 \; mod \; 26 = 7$

Dari hasil perhitungan di atas, diperoleh $15^{-1} \; mod \; 26 = 7$.

8. Metode *fast exponentiation*

Metode ini digunakan untuk menghitung operasi pemangkatan besar bilangan bulat modulo dengan cepat. Metode *fast exponentiation* memanfaatkan ekspansi biner dan eksponennya dan didasarkan pada persamaan 2.9.

$$g^{2^{j+1}} = (g^{2^j})^2 \qquad (2.9)$$

Pada contoh perhitungan di bawah ini akan dijelaskan bagaimana menghitung $6^{73} \; mod \; 100$ menggunakan metode *fast exponentiation*.

$(6)^1 = 6 \; mod \; 100 = 6$

$(6)^2 = 36 \; mod \; 100 = 36$

$(6)^4 = (6^2)^2 = (36)^2 = 1296 \; mod \; 100 = 96$

$(6)^8 = (6^4)^2 = (96)^2 = 9216 \; mod \; 100 = 16$

$(6)^{16} = (6^8)^2 = (16)^2 = 256 \; mod \; 100 = 56$

$(6)^{32} = (6^{16})^2 = (56)^2 = 3136 \; mod \; 100 = 36$

$(6)^{64} = (6^{32})^2 = (36)^2 = 1296 \; mod \; 100 = 96$

Jika 73 dapat diubah menjadi :

$73 = 1 * 2^6 + 1 * 2^3 + 1 * 2^0$ karena bentuk biner dari 73 adalah (01001001) sehingga :

$6^{73} = 6^{2^0} * 6^{2^3} * 6^{2^6}$

$= 6 * 16 * 96$

$= 9216 \; mod \; 100$

$= 16$

2.4.3 Enkripsi dan Dekripsi Pada RSA

Algoritma RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi, dan proses dekripsi [RIY-08].

1. Pembentukan kunci

   Ukuran kunci dalam algoritma RSA menunjuk kepada ukuran dari *n* yang digunakan pada operasi modulo. Berikut ini adalah proses pembentukan kunci. Proses ini dilakukan oleh pihak penerima.

   a. Dipilih bilangan prima *p* dan *q*.

   b. Dihitung nilai *n* menggunakan persamaan 2.10.

   $$n = pq \tag{2.10}$$

   Sebaiknya $p \neq q$, sebab jika $p = q$ maka nilai $n = p^2$ dan nilai p dapat diperoleh dengan menarik akar pangkat dua dari *n*.

   c. Dihitung $\varphi(n)$ menggunakan persamaan 2.6.

   d. Dipilih bilangan bulat positif b secara acak dimana $1 < b < \varphi(n)$ dan prima relatif terhadap (n) atau dengan kata lain $FPB(b, \varphi(n)) = 1$.

   e. Dihitung bilangan bulat positif *a* menggunakan persamaan 2.11.

   $$a = b^{-1} \, mod \, \varphi(n) \tag{2.11}$$

   f. Didapatkan kunci publik 1 (n), kunci publik 2 (b) dan kunci rahasia (a).

2. Enkripsi

   Berikut ini adalah proses enkripsi RSA. Proses ini dilakukan oleh pihak pengirim. Seluruh perhitungan pemangkatan bilangan modulo dilakukan menggunakan metode *fast exponentiation*.

   a. Diambil kunci publik 1 (n) dan kunci publik 2 (b).

   b. Dipilih *plaintext m*, dimana $0 \leq m \leq n - 1$.

   c. Dihitung *ciphertext c* menggunakan persamaan 2.12.

   $$c = m^b \, mod \, n \tag{2.12}$$

   d. Diperoleh *ciphertext c* dan dikirimkan kepada penerima.

3. Dekripsi

   Berikut ini adalah proses dekripsi RSA. Dilakukan oleh pihak penerima *ciphertext*.

   a. Diambil kunci publik 1 (n) dan kunci rahasia (a).

b. Dihitung kembali *plaintext m* menggunakan persamaan 2.13.

$$m = c^a \ mod \ n \tag{2.13}$$

2.5 *Sieve of Eratosthenes*

Dalam beberapa permasalahan, diperlukan cara yang cepat untuk mencari beberapa bilangan prima yang terletak dalam rentang batas tertentu. Permasalahan mencari bilangan prima tersebut digeneralisasi menjadi mencari bilangan prima antara 1 sampai dengan batas atas dari rentang tersebut, sebut saja *N*.

Cara *trial division*, yaitu mengetes semua bilangan tersebut untuk mencari semua prima tidaklah mangkus, karena cara terbaik untuk mengetes apakah sebuah bilangan prima atau komposit masih memerlukan pengecekan dengan semua bilangan dari satu sampai dengan akar dari bilangan tersebut.

Dalam dunia komputer sudah ditemukan beberapa cara untuk mencari bilangan prima tersebut, cara yang termudah dinamakan *sieve of eratosthenes*, sesuai dengan nama orang yang menemukannya, yaitu Eratosthenes, ahli matematika dari Cyrene yang hidup antara 276 SM hingga 195 SM [CHR-12].

Sieve of eratosthenes adalah sebuah algortima klasik untuk menemukan seluruh bilangan prima sampai ke sebuah *N* yang ditentukan. Mulai dari *array* berisi bilangan bulat yang belum dicoret dari 2 ke *N*. Bilangan bulat pertama yang belum dicoret yaitu 2, adalah bilangan prima pertama. Coret seluruh kelipatan dari bilangan prima ini. Ulangi pada bilangan bulat selanjutnya yang belum dicoret.

Urutan langkah dari algortima *sieve of eratosthenes* adalah sebagai berikut [HIM-10] :

1. Buat daftar bilangan dari 2 sampai sebuah bilangan terbesar yang kita nyatakan dengan *N*.
2. Eliminasi atau hilangkan dari daftar semua bilangan kelipatan 2.
3. Angka selanjutnya yang belum dihilangkan adalah bilangan prima.
4. Eliminasi atau hilangkan dari daftar semua bilangan yang merupakan kelipatan dari angka pada langkah sebelumnya.
5. Ulangi langkah 3 dan langkah 4 sampai angka yang lebih besar dari angka yang tersisa setelah langkah 5 adalah bilangan prima.

Sebagai contoh, berikut ini adalah *array* yang berisi bilangan bulat 2 hingga *N* yaitu 27.

2	3	4	5	6	7	8	9	10	11	12	13	14
15	16	17	18	19	20	21	22	23	24	25	26	27

Karena 2 belum dicoret, maka 2 adalah bilangan prima pertama. Selanjutnya, dicoret seluruh kelipatan 2, yaitu 4, 6, 8, 10, 12, dan seterusnya.

2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~	11	~~12~~	13	~~14~~
15	~~16~~	17	~~18~~	19	~~20~~	21	~~22~~	23	~~24~~	25	~~26~~	27

Bilangan bulat selanjutnya yang belum dicoret adalah 3, maka 3 adalah prima dan seluruh kelipatan 3 dicoret, seperti 6, 9, 12, dan seterusnya.

2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~
~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~

Bilangan bulat selanjutnya yang belum dicoret adalah 5, maka 5 adalah prima selanjutnya. Satu-satunya kelipatan 5 pada *array* yang belum dicoret adalah 25.

2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~
~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~

Sisa bilangan bulat pada *array* yang belum dicoret adalah bilangan bilangan prima.

2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~	11	~~12~~	13	~~14~~
~~15~~	~~16~~	17	~~18~~	19	~~20~~	~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~

Algortima *sieve of eratothenes* dapat dikembangkan lebih lanjut sehingga lebih optimal. Pengembangan tersebut meliputi [CHR-12] :

1. Penelusuran untuk mencari bilangan prima dapat dilakukan dari bilangan bulat 2 sampai dengan $\sqrt{N}$. Bilangan komposit yang berada pada rentang $\sqrt{N}$ hinggan N pasti sudah tereliminasi pada saat penelusuran pada rentang 2 hingga $\sqrt{N}$. Oleh karena itu bilangan yang belum tereliminasi pada rentang $\sqrt{N}$ hinggan N pasti bilangan prima.
2. Misalkan I adalah bilangan prima, maka pada saat proses eliminasi semua kelipatan I dapat dimulai dari $I * I$ karena semua kelipatan $I < I * I$ pasti sudah tereliminasi.

2.6 Audio MP3

2.6.1 Pengertian Audio

Audio atau yang lebih dikenal dengan suara adalah suatu getaran yang mengandung frekuensi yang mampu dideteksi oleh telinga manusia. Batasan dari frekuensi tersebut dibagi menjadi beberapa fase seperti yang terlihat pada tabel 2.2 :

Tabel 2.2 Klasifikasi audio berdasarkan frekuensi

Audio	Frekuensi (Hz)
Infrasound	0 - 20
Pendengaran manusia	20 – 20.000
Ultrasound	20.000 – 1000.000.000
Hypersound	1000.000.000 – 10.000.000.000

Suara yang bisa didengar merupakan representasi dari sinyal analog. Sinyal analog terlebih dahulu harus diubah ke dalam sinyal digital dengan mengubah amplitude gelombang bunyi ke dalam waktu interval tertentu. Proses ini dikenal sebagai istilah *Analog To Digital Conversion* (ADC). Sebaliknya jika ingin mengubah sinyal digital ke sinyal analog bisa dilakukan dengan proses *Digital To Analog Converter* (DAC).

Proses ini mengubah sinyal analog mengalami beberapa tahapan agar bisa diubah sedemikian rupa menjadi sinyal digital yang diinginkan. Proses-proses tersebut adalah pencuplikan, kuantisasai, dan pengkodean. Pencuplikan adalah proses perhitungan ulang sampel dari berkas audio pada nilai *sampling rate* yang berbeda pada saat berkas dibuat. Dalam pencuplikan dilakukan pengolahan sinyal analog dengan *sampling* yang ada untuk menghasilkan suatu sinyal yang disebut *discrete-time signal*. Sinyal diskrit ini akan diproses lagi pada tahapan kuantisasi dimana hasil perhitungan dibulatkan ke dalam nilai diskrit [HER-09].

2.6.2 Format Berkas Audio

Format berkas audio adalah sebuah *container* untuk menyimpan data audio pada sistem komputer. Ada tiga kelompok utama dalam pembagian format berkas audio, yaitu [HER-09] :

1. Format audio dengan kompresi *lossless*

   Pada kompresi lossless hasil dekompresi terhadap data yang sudah dikompres tepat sama persis dengan data sebelum dikompres, contoh : *WavPack* dengan berkas WV.
2. Format audio dengan kompresi *lossy*

   Pada kompresi *lossy* hasil dekompresi terhadap data yang sudah dikompres tidak sama persis tetapi persepsi terhadap semantik data tetap sama, contoh : MP3.
3. Format audio dengan tidak terkompresi

   Sinyal digital tidak mengalami kompresi apapun, contoh : *Waveform* Audio *Format* (WAV) pada Microsoft ataupun Audio *Interchange File Format* (AIFF) pada Mac OS.

2.6.3 MPEG Audio Layer 3 (MP3)

Asal mula MP3 dimulai dari penelitian IIS-FHG (Institut Intergriette Schaltungen Fraunhofer Gessellschaft), sebuah lembaga penelitian terapan di Munich, Jerman dalam penelitian terapan *coding* audio *perceptual*. Penelitian mengenai pemampatan berkas audio ini dipimpin 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 sebaga ISO-MPEG Audio Layer-3 (MP3), yang merupakan berkas dengan teknik *lossy compression*.

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 berkas audio. Sebagai contoh, apabila terdapat dua nada yang mirip, atau apabila nada tinggi dan rendah muncul secara bersamaan, otak hanya akan memproses salah satunya. Berdasarkan keadaan tersebut, maka algoritma pada MP3 hanya akan memilih sinyal yang lebih penting dan membuang sisanya sehingga berkas audio MP3 memiliki ukuran berkas audio orisinal hingga 10 kali lebih kecil. Teknologi ini kemudian distandardisasi pada tahun 1991.

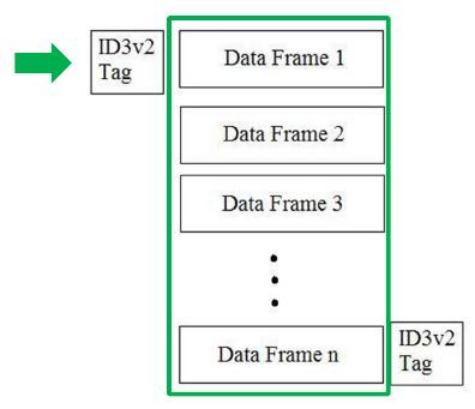
Teknik kompresi MP3 dapat menghasilkan suatu berkas audio terkompresi dengan kualitas suara yang tidak berbeda jauh dari aslinya karena memanfaatkan kelemahan atau keterbatasan dari sistem pendengaran manusia. Berikut adalah beberapa kelemahan dari sistem pendengaran yang digunakan dalam pemodelan kompresi MP3 [CHA-09] :

1. Terdapat beberapa suara yang tidak dapat didengar manusia, yaitu suara yang memiliki frekuensi di luar rentang 20 Hz sampai dengan 20000 Hz.
2. Terdapat suara yang dapat terdengar lebih baik bagi pendengaran manusia dibandingkan suara lainnya.
3. Bila terdapat dua suara yang dikeluarkan secara simultan, maka pendengaran manusia akan mendengar yang lebih keras, sedangkan yang lebih pelan akan tidak terdengar.

Kepopuleran dari MP3 yang sampai saat ini belum tersaingi disebabkan oleh beberapa hal. Salah satunya adalah MP3 dapat didistribusikan dengan mudah hampir tanpa biaya, walaupun sebenarnya hak paten dari MP3 telah dimiliki dan penyebaran MP3 seharusnya dikenai biaya. Walaupun begitu, pemilik hak paten dari MP3 telah memberikan pernyataan bahwa penggunaan MP3 untuk keperluan perorangan tidak dikenai biaya. Keuntungan lainnya adalah kemudahan akses MP3, dimana banyak perangkat lunak yang dapat menghasilkan berkas MP3 dari CD dan bersifat kosmopolit.

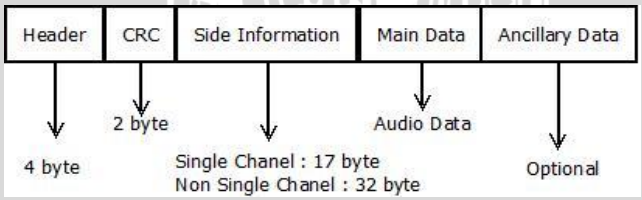
2.6.4 Struktur Berkas MP3

Menurut [RAS-02], berkas MP3 terdiri dari bagian-bagian kecil yang disebut *frame*. Setiap *frame* memiliki waktu yang konstan yaitu 0.026 detik (26 milidetik). Ukuran dari sebuah *frame* (dalam byte) memiliki nilai yang bervariasi tergantung dari *bitrate*. Gambar 2.5 menunjukkan struktur berkas MP3 yang terdiri dari beberapa *frame*.



Gambar 2.5 Struktur Berkas MP3

Sebuah *frame* MP3 secara umum terdiri atas 5 bagian, yaitu *header*, CRC, *side information*, *main data*, dan *ancilarry data*. Gambar 2.6 menunjukkan struktur dari sebuah *frame* MP3.



Gambar 2.6 Struktur *Frame* MP3 [RAS-02]

Menurut Baskara [BAS-08], setiap *tag* MP3 atau awal berkas MP3 diawali dengan 3 byte bilangan heksadesimal 49, 44, dan 33. Untuk setiap *frame* pada MP3 selalu diawal *header* yang ditandai dengan bilangan heksadesimal FF dan FB. Menurut Sripada [SRI-06], ketika melakukan pembacaan berkas MP3 panjang dari masing-masing *frame* harus dihitung terlebih dahulu. Penghitungan

panjang *frame* dilakukan untuk menemukan lokasi dari *frame* berikutnya. Panjang dari masing-masing *frame* pada suatu berkas MP3 dapat berubah-ubah dari satu *frame* ke *frame* yang lain tergantung dari bit *padding* dan *bitrate* pada masing-masing *frame*. Persamaan untuk menghitung panjang suatu *frame* ditunjukkan pada persamaan 2.14.

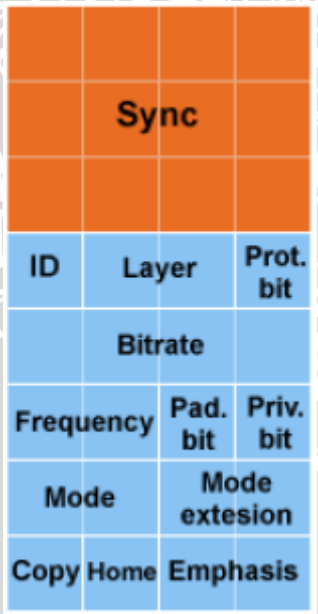
$$FLB = 144 * \frac{bitrate}{sampling\ rate} + padding \qquad (2.14)$$

dimana *FLB* merupakan panjang dari *frame* dalam byte.

Berikut ini merupakan bagian-bagian dari *frame* MP3 [RAS-02] :

1. *Header*

   Header adalah bagian dari sebuah *frame* dengan panjang 4 byte (32 bit) yang berisi bit-bit sinkronisasi dan deskripsi tentang *frame* tersebut. Struktur dari *header* MP3 ditunjukkan pada gambar 2.7 dibawah ini :



Gambar 2.7 Struktur *Header* MP3 [RAS-02]

Keterangan mengenai fungsi dan kebutuhan bit *header* ditunjukkan pada tabel 2.3.

Tabel 2.3 Fungsi dan kebutuhan bit *header*

Posisi	Keterangan Fungsi	Panjang (Bit)
Sync	*Frame* sinkronisasi	12
Id	Versi MPEG audio (MPEG-1, MPEG-2, dll)	1
Layer	MPEG *layer* (*Layer* I, II, III dll)	2
Prot. Bit	Bit proteksi, jika bit ini diset maka CRC akan digunakan	1
Bitrate	Indeks *bitrate*	4
Frequency	Frekuensi *sampling rate*	2
Pad. bit	Bit *padding*	1
Priv. bit	*Private* bit	1
Mode	Mode *channel* (*stereo, joint stereo*, dll)	2
Mode Extension	Mode *extension*	2
Copy	Hak cipta	1
Home	Orisinalitas	1
Emphasis	*Emphasis*	2

Beberapa keterangan *bit position* antara lain :

a. Bit sinkronisasi

Sinkronisasi terdiri dari 12 bit yang harus bernilai 1111 1111 1111.

b. Bit *id*

Menentukan versi dari MPEG. Ketika bit *id* bernilai 1, maka *frame* akan dikodekan menggunakan standar MPEG-1. *Frame* akan dikodekan menggunakan standar MPEG-2 ketika bit *id* bernilai 0. Terdapat beberapa standar lain yang hanya menggunakan 11 bit sinkronisasi sehingga sisa satu bit dapat digunakan untuk *id*. Tabel 2.4 menunjukkan penggunaan bit *id* yang terdiri dari 2 bit.

Tabel 2.4 Bit *id* 2 bit

00	MPEG-2.5 (Pengembangan dari MPEG-2)
01	*Reserved*
10	MPEG-2
11	MPEG-1

c. *Layer*

Bit *layer* terdiri dari dua bit. Tabel 2.5 menunjukkan beberapa *layer* yang ditentukan oleh bit *layer*.

Tabel 2.5 Bit *layer*

Nilai Bit	*Layer*
00	Tidak terdefinisi
01	Layer III
10	Layer II
11	Layer I

d. *Bitrate*

Bitrate merupakan informasi jumlah data yang akan disimpan dalam satuan detik pada audio yang belum terkompresi. *Bitrate* ditentukan sebelum dilakukan proses *encoding* sehingga menentukan kualitas dari hasil *encoding*. Pada *layer* III ditetapkan nilai *bitrate* antara 8kbit/s hingga 320 kbit/s dimana *default*-nya bernilai 128 kbit/s. Pembagian nilai *bitrate* ditunjukkan pada tabel 2.6.

Tabel 2.6 *Bitrate*

Bitrate Index	MPEG 1		
	Layer I	Layer II	Layer III
0000	Free	Free	Free
0001	32	32	32
0010	64	48	40

0011	96	56	48
0100	128	64	56
0101	160	80	64
0110	192	96	80
0111	224	112	96
1000	256	128	112
1001	288	160	128
1010	320	192	160
1011	352	224	192
1100	384	256	224
1101	416	320	256
1110	448	384	320
1111	Reserved	Reserved	Reserved

e. Bit *frequency*

Bit *frequency* atau *sampling frequency* didefinisikan sebagai suatu nilai sampel per detik yang dihasilkan dari sinyal *continuous* menjadi sinyal diskrit. Notasinya adalah hertz (Hz). Kebalikan dari *sampling frequency* adalah *period sampling*, yaitu waktu yang dibutuhkan untuk melakukan *sampling*. Tidak ada aturan baku yang mengatur berapa batas *sampling* yang diperkenankan pada satu *period sampling*. Pembagian nilai bit frekuensi ditunjukkan pada tabel 2.7.

Tabel 2.7 Bit frekuensi

Nilai Bit	MPEG 1	MPEG 2
00	44100 Hz	22050 Hz
01	48000 Hz	24000 Hz
10	32000 Hz	16000 Hz

f. Bit *mode*

Bit *mode* dibedakan menjadi empat, yaitu :

1. *Mono*

   Mono memiliki kanal audio hanya satu. Audio *mono* hanya menghasilkan satu suara yang didengar oleh kedua telinga kita, sehingga suara yang diterima oleh kedua telinga kita selalu sama. Sinyal *mono* adalah R+L (*right and left*), dimana R dan L digabungkan sehingga menjadi satu sinyal R+L. Penggabungan tersebut dibuat agar bisa mendengan kedua sinyal dalam satu sumber suara.

2. *Stereo*

   Audio *stereo* dapat menghasilkan efek suara seperti efek *doppler*. Musik yang berbeda antar kanal, misalnya kanal R suara bass, kanal L suara gitar.

3. *Joint stereo*

   Joint stereo adalah suatu trik untuk mengkodekan sinyal frekuensi rendah secara *mono* sementara sisanya tetap *stereo*.

4. *Dual channel*

   Salah satu cara untuk menggabungkan audio jenis *stereo* dan juga audio *mono*.

Pembagian bit mode ditunjukkan pada tabel 2.8

Tabel 2.8 Bit *mode*

Nilai Bit	*Mode*
00	*Stereo*
01	*Joint stereo*
10	*Dual channel*
11	*Mono*

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.

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 *decoding* bagian *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 berkas MP3 berada dan mempunyai panjang yang bervariasi. Disini terdapat *huffman code bits*, informasi untuk *decoding huffman code bits* ini terdapat pada bagian *side information*. Bagian *frame* akan digunakan dalam steganografi ini adalah dengan mengganti isi dari *main data*. Teknik kompresi MP3 juga menetapkan metode *huffman coding* untuk mendapatkan hasil kompresi yang lebih baik. Pada berkas MP3 yang tidak menggunakan CRC, umumnya letak *main data* dimulai setelah byte ke-36 hingga *frame* selanjutnya.

5. *Ancillary data*

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

2.7 PSNR (*Peak Signal to Noise Ratio*)

Dasar yang paling penting pada audio steganografi adalah penyembunyian data tidak boleh mengubah kualitas suara dari sinyal audio yang digunakan sebagai media penampung. Jadi pesan rahasia yang disembunyikan tidak boleh terdeteksi oleh pendengar [SAR-06]. Oleh karena itu, perlu dilakukan perbandingan antara dua buah berkas audio untuk mengetahui tingkat perbedaan

kualitas antara berkas audio asli dengan berkas audio setelah proses steganografi. Ada beberapa metode untuk menghitung metrik perbandingan antara dua buah sinyal, diantaranya adalah PSNR.

Peak Signal to Noise Ratio (PSNR) adalah perbandingan antara nilai maksimum dari sinyal yang diukur dengan besarnya derau yang berpengaruh pada sinyal tersebut. PSNR biasanya diukur dalam satuan desibel. (Alatas, 2009). PSNR digunakan untuk mengukur rasio antara berkas audio asli dengan berkas audio yang telah disisipi pesan. PSNR yang rendah menunjukkan bahwa berkas audio telah mengalami distorsi yang cukup besar. Kualitas audio yang baik memiliki nilai PSNR minimal 30 dB [NEY-11].

Persamaan PSNR dapat dilihat pada persamaan 2.15.

$$PSNR = 10\log_{10}\left(\frac{{MAX_I}^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (2.15)$$

Jika sinyal suara yang ingin dihitung mempunyai *bits per sample*, maka MAX_I dapat dirumuskan pada persamaan 2.16.

$$MAX_I = 2^I - 1 \quad (2.16)$$

2.7.1 MSE (*Mean Square Error*)

MSE adalah salah satu cara untuk menghitung nilai kedekatan antara estimator dengan nilai yang diestimasi. Jika *n* adalah jumlah sampel, *I* dan *K* masing-masing adalah estimator dan nilai yang ingin diestimasi untuk tiap sampel, maka MSE dapat dirumuskan pada persamaan 2.17.

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1} \|I(i) - K(i)\|^2 \quad (2.17)$$

2.8 *Bit Error Rate*

Bit error rate didefinisikan sebagai perbandingan bit hasil pengungkapan yang berbeda dari bit pesan yang disembunyikan. *Bit error rate* digunakan untuk menghitung persentase bit pesan yang berbeda saat proses pengungkapan pesan. *Bit error rate* dihitung menggunakan persamaan 2.18.

$$BER = \frac{100}{B}\sum_{n=0}^{B-1}\begin{cases}1, X(n) \neq Y(n)\\ 0, X(n) = Y(n)\end{cases} \quad (2.18)$$

Pada persamaan 2.17, *B* adalah jumlah bit pesan rahasia, *X* adalah bit pesan yang disembunyikan dan *Y* adalah bit pesan hasil pengungkapan [NEY-11].

2.9 Penambahan Derau

Derau merupakan suara-suara yang tidak diinginkan [DIA-11]. Munculnya derau dapat menurunkan kualitas suatu berkas audio. Penambahan derau dapat dilakukan pada dua domain yaitu domain waktu dan domain frekuensi. Penambahan derau pada domain waktu dilakukan dengan menambahkan sinyal data dengan frekuensi derau yang telah dimultiplikasi dengan amplitudo tertentu. Untuk penambahan derau pada domain frekuensi, dapat dilakukan dengan mengubah sinyal ke domain frekuensi menggunakan transformasi fourier dan menambahkan sinyal hasil transformasi tersebut dengan frekuensi derau yang telah dimultiplikasi dengan amplitudo tertentu, kemudian ditransformasi lagi menjadi domain waktu dengan transformasi fourier. Neyman [NEY-11] menambahkan derau dengan intensitas tertentu untuk mengevaluasi ketahanan steganografi terhadap operasi penambahan derau.

Derau *gaussian* adalah derau yag bersifat aditif. Sinyal audio asli yang ditambah dengan derau *gaussian* membentuk sinyal berderau *gaussian*. Derau *gaussian* adalah derau yang memiliki nilai persebaran secara acak yang dapat diperkirakan secara statistik mengikuti distribusi normal. Derau *gaussian* dibangkitkan menggunakan fungsi distribusi normal pada persamaan 2.19.

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}}\ e^{-(z-\pi)^2/2\sigma^2} \quad (2.19)$$