Steganografi Audio untuk penyimpanan lirik lagu

Muhamad Ihsan
13511049
Teknik Informatika ITB
Bandung, Indonesia
13511049@std.stei.itb.ac.id

***Abstrak*—Dewasa ini, industri musik sudah memiliki banyak cara untuk distribusi musik yang telah diproduksi. Jika sebelumnya metode penyimpanan musik adalah dalam bentuk kaset tape, piringan hitam (*vinyl*), maupun CD, sekarang perusahaan rekaman juga sudah banyak yang menjual musik melalui media daring, dan karena itulah, media penyimpanan musik yang paling umum saat ini adalah dalam bentuk digital, dengan format penyimpanan terpopulernya adalah MP3 dan WAV. Lirik untuk lagu-lagu tersebut biasanya pun disimpan didalam suatu tempat (*file*) terpisah. Untuk membuat susunan penyimpanan *file-file* musik lebih rapi, maka alangkah lebih baiknya apabila segala informasi tambahan mengenai lagu tersebut, khususnya liriknya, dapat disimpan kedalam satu *file* lagu itu juga. Salah satu metode penyimpanan tersebut adalah steganografi audio. Untuk menyembunyikan lirik tersebut lebih jauh lagi, dapat digunakan *block cipher*. Penyembunyian lebih jauh dilakukan agar beberapa lirik dan informasi tertentu hanya dapat dibuka oleh suatu program *decoder* yang spesifik saja.**

***Kata kunci*—*MP3, WAV, lirik, steganografi audio,* block cipher**

I. PENDAHULUAN

Steganografi merupakan sebuah metode yang digunakan untuk menyembunyikan suatu pesan dibalik suatu hal lain. Steganografi paling umum diterapkan dalam penyimpanan pesan ke dalam sebuah citra/gambar, tetapi media steganografi tidak terbatas pada citra saja, melainkan dapat juga kedalam media lain seperti musik dan video.

Steganografi audio berarti menyembunyikan pesan kedalam suatu *file* musik, seperti *file* berformat MP3, WAV, atau lainnya. Fungsi utama steganografi adalah untuk menyampaikan pesan dari si pengirim ke penerima tanpa menimbulkan kecurigaan karena pesan yang dikirim tidak terlihat seperti sebuah pesan, melainkan dalam bentuk lain.Tetapi steganografi mempunyai sebuah fungsi sampingan, yaitu sekadar menyisipkan informasi tambahan kedalam suatu *file*, dalam kasus ini, kedalam suatu *file* musik.

Seperti telah diketahui, musik dalam bentuk digital menjadi bentuk distribusi paling populer saat ini, karena internet semakin mudah diakses oleh siapapun dan dimanapun. Dan beberapa orang juga suka menyanyikan lagu tersebut saat dimainkan, dan karena itulah lirik lagu sangat dibutuhkan bersamaan dengan lagu itu sendiri. Menyimpan lagu dan liriknya secara terpisah merupakan hal yang sangat merepotkan, maka dari itu diperlukan suatu metode penyimpanan lagu dan lirik secara otomatis kedalam satu *file*. Hal ini telah dilakukan oleh video, dimana beberapa format video mendukung adanya *embedded subtitle* kedalam satu *file* video. Format-format tersebut diantaranya adalah MP4, FLV, dan MKV.

Untuk memainkan *file* lagu yang sudah disisipkan lirik tentu saja memerlukan pemutar khusus untuk memainkan lagu dan menampilkan lirik yang telah di*decode* melalui *decoder* steganografi secara bersamaan. Pemutar tersebut akan berupa gabungan pemutar biasa dan *decoder* steganografi audio. Tentu saja, karena *file* tersebut merupakan hasil penyisipan menggunakan steganografi, maka suara yang terdengar tidak akan jauh berbeda, walaupun untuk beberapa orang yang peka, sedikit perbedaan ini dapat terdeteksi, tetapi tidak akan terlalu berpengaruh pada kualitas musik aslinya. Karena steganografi pula, *file* hasil *encoding* menggunakan steganografi audio diatas tetap bisa dimainkan di pemutar lagu biasa, untuk orang-orang yang tidak terlalu memerhatikan lirik dari sebuah lagu.

Mungkin kelak akan terdapat suatu kasus dimana lirik dari suatu lagu yang telah disisipkan hanya boleh ditayangkan di pemutar tertentu saja. Maka dari itu, sebelum disisipkan dengan menggunakan steganografi, lirik tersebut dapat dienkripsi terlebih dahulu dengan menggunakan *block cipher.* Enkripsi dengan *block cipher* menjamin lirik yang disisipkan ke suatu *file* tidak bisa dibuka melainkan melalui suatu program tertentu, karena walaupun *file* tersebut berhasil dipecahkan dengan menggunakan steganalisis, tetapi pemecahan *block cipher* tentu saja berbeda, dan seharusnya lebih sulit. *Block cipher* yang akan digunakan adalah DES.

II. DASAR TEORI

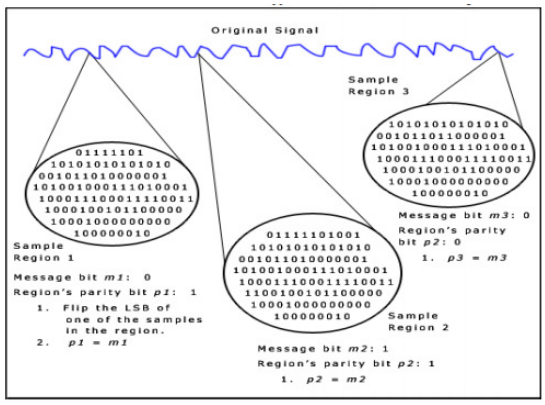
A. *Steganografi Audio*

Steganografi secara umum merupakan proses menyembunyikan suatu pesan ke dalam suatu *file* lain yang tidak mencurigakan dan tidak terlihat seperti menyembunyikan apapun. Steganografi dapat menggunakan medium apapun, termasuk didalamnya citra, audio, maupun video. Metode steganografi berbeda-beda tergantung pada medium yang digunakan, tetapi metode paling dasar dari steganografi pasti mengandung modifikasi LSB (*least significant bit*) dari potongan terkecil dari suatu *file*. Untuk *file* citra, potongan

terkecilnya berupa satu *pixel* gambar, sedangkan untuk *file* audio, potongan terkecilnya berupa sebuah sampel individu yang telah diambil dari sinyal utama audionya.

Tentu saja masih banyak metode steganografi audio selain modifikasi LSB, diantaranya adalah Parity Coding, Phase Coding, Spread Spectrum, dan Echo Hiding.

Parity Coding merupakan teknik yang memecah sinyal audio menjadi beberapa kelompok sampel, bukan memecah menjadi tiap sampel individu. Seperti menyisipkan bit ke LSB, tetapi untuk Parity Coding, bit tersebut disisipkan ke *parity bit* dari setiap kelompok sampel. Ketika *parity bit* dari kelompok tersebut tidak sesuai dengan bit yang ingin disisipkan, maka LSB salah satu sampel diubah.

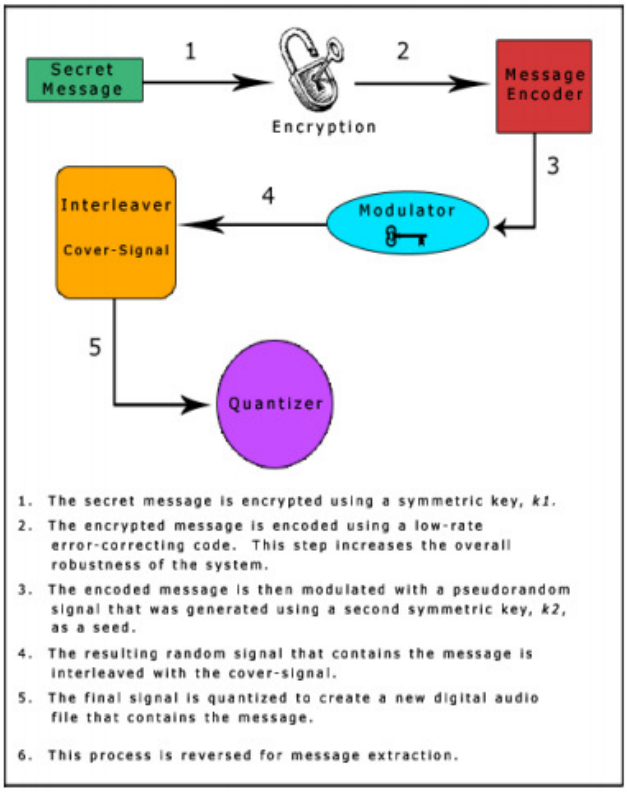


Gambar 1. Parity Coding

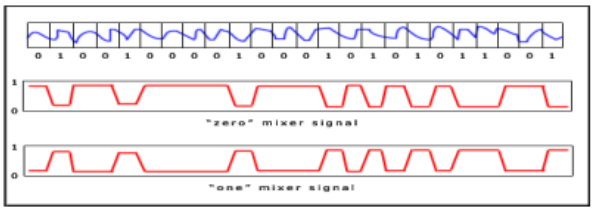
Phase Coding memanfaatkan perpindahan *phase* antara beberapa segmen didalam *file* audio. Pertama-tama, setiap segmen dijadikan matriks terlebih dahulu dengan menggunakan DFT (*Discrete Fourier Transform*), kemudian dihitung perbedaan antara *phase* yang ada. Perbedaan *phase* tersebut menjadi pertimbangan untuk penyisipan bit kedalam setiap segmen.

Spread Spectrum menyisipkan data dengan cara memasukkan beberapa buah kopi data tersebut kedalam berbagai tempat didalam spektrum frekuensi suara. Hal ini dilakukan agar apabila terdapat keanehan di beberapa tempat penyimpanan data, masih terdapat penyimpanan cadangan untuk data tersebut agar tetap dapat diambil (*robustness* tinggi).

Echo Hiding menyisipkan data ke dalam audio dengan memasukkannya sebagai *echo* (gema). Tetapi, satu *file* audio hanya akan menghasilkan satu *echo*, dan dari itu juga, satu bit pesan. Maka dari itu, sebelumnya audio dipecah menjadi beberapa blok, yang diakhir akan digabungkan lagi menjadi satu *file* utuh.

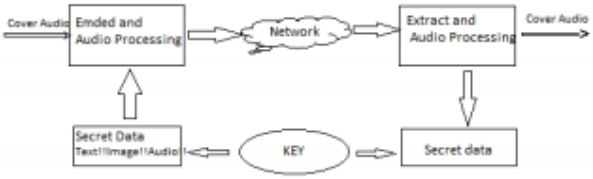


Gambar 2 Spread Spectrum



Gambar 3 Echo Hiding

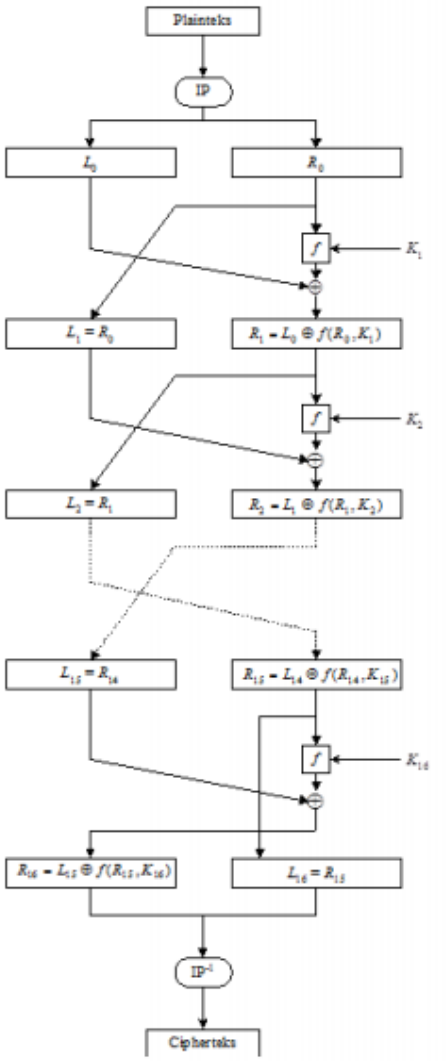
Walaupun terdiri dari banyak metode, tetapi pada dasarnya steganografi audio terdiri dari beberapa langkah sederhana. Diagram keseluruhan langkah steganografi audio secara umum dapat dilihat dibawah ini :



Gambar 4. Skema umum Steganografi Audio

B. DES(Data Encryption Standard)

DES (Data Encryption Standard) merupakan standard yang dikembangkan di IBM pada tahun 1972. DES tergolong kedalam kriptografi kunci-simetri dan tergolong juga kedalam jenis cipher blok. DES beroperasi pada ukuran blok 64 bit. Panjang sebuah kunci eksternal DES adalah 64 bit, tetapi hanya 56 bit yang dipakai dalam perhitungan.



Gambar 5. Skema umum algoritma DES

Setiap blok awalnya dipermutasi dengan menggunakan IP (Initial Permutation), kemudian di enkripsi sebanyak 16 putaran, dan terakhir dipermutasi lagi dengan menggunakan inverse IP. Hasil akhirnya merupakan cipherteks dari blok plainteks yang dijadikan input DES. Setiap putaran pada tahap enkripsi menggunakan kunci internal yang berbeda-beda. Kunci internal dibangkitkan dari kunci eksternal, dan besarnya 56 bit.

Sebuah putaran enciphering merupakan jaringan Feistel, dimana hasil Li dan Ri masing-masing adalah :

Li = Ri-1

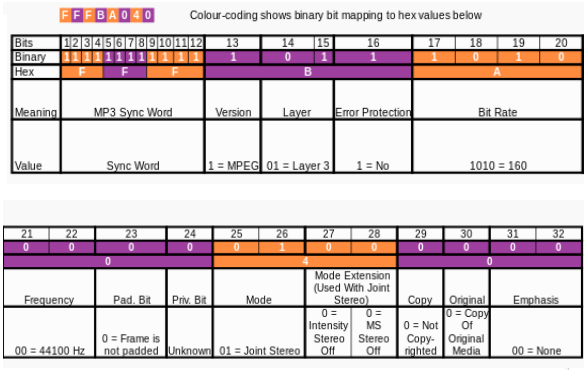
Ri = Li-1 (xor) f(Ri-i, Ki)

Pertama, Ri-1 diekspansi dengan menggunakan matriks ekspansi, dari awalnya berukuran 32 bit menjadi 48 bit. Kemudian hasilnya di-xor-kan dengan kunci internal untuk putaran tersebut (Ki). Hasilnya akan berupa sebuah vektor A berukuran 48 bit. A kemudian dipecah menjadi 8 buah vektor kecil berukuran 6 bit, dan sesuai urutannya, dimasukkan satu per satu ke s-box (substitution box) yang akan menerima input 6 bit dan mengeluarkan hasil 4 bit. Hasil substitusi tersebut adalah vektor B yang berukuran 32 bit. B kemudian diacak lagi dengan menggunakan matriks permutasi P. Hasil permutasi terakhir tersebut lah yang menjadi keluaran fungsi f.

Untuk dekripsi algoritma DES, prosedurnya sama persis dengan enkripsi, tetapi terletak di urutan penggunaan kuncinya. Pembangkitan kunci eksternal menghasilkan kunci internal K1,K2,...,K16, dan pada enkripsi, kunci K1 digunakan untuk putaran enkripsi pertama, K2 untuk putaran enkripsi kedua, dan seterusnya. Sedangkan pada dekripsi, kunci K16 lah yang digunakan untuk putaran dekripsi pertama, K15 untuk putaran kedua, dan seterusnya.

C. Struktur file MP3

Struktur *file* MP3 berupa perulangan dua buah *frame*, yaitu MP3 Header dan MP3 Data. MP3 Data tentu merupakan sinyal lagu tersebut. Sedangkan MP3 Header formatnya dapat dilihat dibawah ini :



Gambar 6 Struktur Header MP3

III. STEGANOGRAFI AUDIO UNTUK PENYIMPANAN LIRIK

Untuk menyisipkan lirik kedalam sebuah *file* mp3, dibutuhkan algoritma steganografi audio yang tepat. Untuk itu, metode steganografi audio yang digunakan adalah metode Parity Coding. Sebelum itu, tentu saja lirik yang ingin disisipkan dienkripsi terlebih dahulu dengan menggunakan algoritma DES.

Proses Parity Coding dimulai di Inner Loop. Inner loop menghitung seberapa besar data yang akan disisipkan dan

menyesuaikan ukurannya sampai data yang ada dapat dimasukkan ke bit-bit audio yang tersedia. Terdapat satu buah *loop* tambahan yang akan mengecek perubahan-perubahan yang terjadi, apakah terlalu jauh berbeda dengan kualitas *file* awal atau tidak. Psychoacoustic Model (oleh AT&T) digunakan untuk membantu menentukan hal tersebut.

Lalu terdapat variabel yang menyimpan jumlah bit yang digunakan untuk *scalefactor* dan data Huffman di bit *stream* MP3. Bit-bit yang ada kemudian disisipkan sebagai *parity bit*, dengan mengubah kondisi berhenti Inner Loop yang telah disebutkan diatas. Hanya nilai-nilai tertentu yang akan diubah. Nilai-nilai tersebut didapat dengan menggunakan *pseudo-random bit generator* dari SHA-1.

IV. Implementasi dan Pengujian

Program dibuat dengan menggunakan bahasa C, dengan IDE Microsoft Visual Studio. Terdapat dua buah program utama, yaitu untuk *decoding* dan *encoding*. Untuk *encoding*, program menerima input berupa *file* teks berisi lirik yang akan disisipkan, *file* mp3 dimana teks tersebut akan disisipkan, *file* mp3 output dari *encoding* tersebut, dan terakhir key untuk algoritma DES. Untuk *decoding*, input yang dibutuhkan hanya *file* mp3 yang ingin diekstrak pesan didalamnya, dan juga key untuk algoritma DES.

Aspek yang paling penting untuk diujicoba adalah mengenai kapasitas satu *file* lagu terhadap lirik yang ingin disisipkan. Untuk itu, cukup satu lagu yang dijadikan tolok ukur, dengan harapan jika lagu tersebut dapat dimuati oleh liriknya sendiri, maka lagu lain pun bisa. Lagu yang akan diujicoba harus lah lagu dengan lirik terpadat (rasio kata/detik nya tinggi)

Menurut Guinness Book of World Record, lagu dengan rasio kata/detik tertinggi adalah lagu "Rap God" yang dinyanyikan oleh Eminem dari USA. Lagu tersebut mengandung 1560 kata, yang dinyanyikan dalam rentang waktu 6 menit dan 4 detik. Rata-rata kata/detik nya adalah 4.28.

Untuk uji coba, lirik tersebut ditulis kedalam sebuah *file* .txt sebesar 8257 bytes. Karena ukuran output dari algoritma DES sama besarnya dengan ukuran input, maka ukuran pesan yang ingin disisipkan setelah dienkripsi sama besarnya dengan ukuran awal. Kemudian, ukuran *file* MP3 yang ingin disisipkan adalah 5681 KB, dengan panjang lagu 6'4".

Encoding berhasil dilakukan dengan sempurna, yakni seluruh pesan yang ingin disisipkan berhasil disisipkan (kapasitas *file* MP3 lebih besar dari ukuran pesan). *Encoding* memakan waktu 56060 ms.

Proses *Decoding* juga berhasil dilakukan dengan sempurna, yakni berhasil mengektrak pesan yang disisipkan secara utuh. Proses ini memakan waktu 26025 ms.

V. Kesimpulan

Penyisipan lirik kedalam *file* MP3 berhasil dilakukan dengan menggunakan algoritma steganografi audio dan dienkripsi dengan menggunakan DES. Kapasitas *file* MP3 sudah teruji, karena setelah diuji dengan lagu yang memiliki rasio kata/detik yang tinggi, liriknya masih bisa disisipkan ke dalam *file* MP3. Masalah paling besarnya ada pada waktu *decoding*. Dengan *decoding* yang memakan waktu puluhan detik, pemutar lagu harus mencari cara agar lirik dapat dimuat sebelum lagu diputar, jika tidak, pengguna terpaksa harus menunggu selama puluhan detik sebelum lirik lagu tersebut berhasil dimuat.

Referensi

[1] Ross J. Anderson and Fabien A.P. Petitcolas. On The Limits of Steganography. IEEE Journal of Selected Areas in Communications, 16(4):474-481, May 1998. Special Issue on Copyright & Privacy Protection. ISSN 0733-8716.

[2] M. Swati, S. Manish, and K. Anubhuti. Audio Steganography by Different Methods. International Journal of Emerging Technology and Advanced Engineering, vol. 2, Issue 7, July 2012. ISSN 2250-2459.

[3] P. Jayaram, H. R. Ranganatha, and H. S. Anupama. Information Hiding Using Audio Steganography – A Survey. The International Journal of Multimedia & Its Applications (IJMA) vol. 3, no. 3, August 2011.

[4] K. Gopalan. Audio Steganography Using Bit Modification. ICME 2003.

[5] Supurovic, Predrag (22 December 1999). "MPEG Audio Frame Header". Retrieved 29 May 2009.

[6] "Most Words in a Hit Single." Guinness World Records. Guinness, 1 Jan. 2013. Web. 11 May 2015.

[7] D. Coppersmith. The Data Encryption Standard (DES) and its strength against attacks. IBM Journal of Research and Development vol. 38, Issue 3, May 1994. ISSN 0018-8646.