Digital Audio Watermarking dengan Fast Fourier Transform

Otniel 13508108
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]if18108@students.itb.ac.id

Penyembunyian sesuatu ke dalam file dapat dilakukan pada media apapun. File audio, adalah salah satu jenis file yang rentan terhadap pembajakan. Oleh karena itu, digital wtermarking pada file audio, yang dalam hal ini berformat wav, akan sangat berguna untuk pelindungan hak cipta.

Pengolahan file audio tersebut menjadi masalah dalam waktu komputasi. Pada algoritma standard DFT (Discrete Fourier Transform) waktu eksekusi yang dibutuhkan relative lama . Untuk itu, dibutuhkan algoritma yang cepat untuk pentransformasiannya. Fast Fourier Transform (FFT) dapat membantu penyisipan dalam ranah frekuensi dengan cepat.

***Kata Kunci : audio, wav, Discrete Fourier Transform, Fast Fourier Transform, waktu eksekusi*.**

I. PENDAHULUAN

Perlindungan akan hak cipta sekarang ini semakin perlu ditingkatkan karena makin makarnya pembajakan, baik itu perangkat lunak, film, gambar, maupun music. Namun, untuk mencapai tujuan itu dibutuhkan suatu teknik yang cukup baik untuk dapat membuat semacam penanda terhadap suatu karya tertentu (watermarking). Teknik yang terkenal digunakan untuk tujuan watermarking tersebut adalah steganography. Pada steganography gambar, kita dapat dengan mudah menggantikan LSB bit dengan kode biner dari penanda yang ingin kita sisipkan. Namun berbeda halnya pada watermarkin pada file audio. File audio akan menghasilkan watermarking yang baik jika penggatian bit-bit code yang merepresentasikan frekuensi suara berada pada range frekuensi < 20Hz dan diatas 20KHz, yakni frekuensi suara yang tidak terdengar oleh manusia

Suara merupakan sesuatu yang berbeda dengan gambar. Jika pada gambar, teknik steganografi memanfaatkan kelemahan mata manusia, maka pada suara, memanfaatkan pendengaran manusia. Watermark dengan cara ini menyembunyikan pesan watermark pada frekuensi rendah, frekuensi yang berada dibawah batas toleransi frekuensi yang dapat didengar manusia.

Pengolahan file audio tersebut menjadi masalah. Dibutuhkan algoritma yang cepat untuk pentransformasiannya. Fast fourier transform dapat membantu penyisipan dalam ranah frekuensi dengan cepat.

II. FOURIER SERIES

Fourier series (deret fourier) dalam matematika digunakan untuk memecah sekumpulan sinyal periodic menjadi set dari fungsi osilasi sederhana. Dalam bentuk matematis, persamaan fourier tersebut adalah

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi k F_0 t} \quad (1)$$

x(t) menyatakan sinyal periodic

c_k menyatakan bentuk dari gelombang

Yang merupakan sinyal periodic dengan periode fundamental $T_p = 1/F_0$dan sinyal eksponensial

$$e^{j2\pi k F_0 t} \quad (2)$$

, dengan k = 0, $\pm 1, \pm 2, \ldots$ sebagai dasar dari building block dimana bisa kita bangun sinyal periodic dari bermacam –macam tipe dengan melakukan pemilihan yang sesuai dari fundamental frekuensi dan koefisien c_k. Namun persamaan diatas masih terhadap waktu. Akan dibentuk persamaan baru dalam domain frekuensi.

Untuk mengetahui transformasi fourier dari persamaan di atas, diperlukan nilai dari c_k. Berikut adalah penurunan besaran c_k.

Dari persamaan (1), kita kalikan kedua ruas dengan eksponensial kompleks $e^{j2\pi k F_0 t}$ kemudian integralkan kedua ruas terhadap t, maka akan didapat

$$\int_{t_0}^{t_0+T_p} x(t) e^{-j2\pi l F_0 t}\, dt = \int_{t_0}^{t_0+T_p} e^{-j2\pi l F_0 t} \left(\sum_{k=-\infty}^{\infty} (c_k e^{j2\pi k F_0 t}) \right) dt \quad (3)$$

Sesuai dengan sifat penjumlahan, jika terdapat penjumlahan $na_1 + na_2 + na_3 + \cdots$ maka dapat kita bentuk menjadi

$$n \sum_{i=1}^{m} a_m$$

Maka, dengan aturan tersebut, kita dapat mempertukarkan pengintegralan pada ruas kanan tersebut

kedalam penjumlahan (sigma). Didapat

$$\sum_{k=-\infty}^{\infty} c_k \int_{t_0}^{t_0+T_p} c_k e^{j2\pi(k-l)F_0 t}$$

$$= \sum_{k=-\infty}^{\infty} c_k \left[\frac{e^{j2\pi(k-l)F_0 t}}{j2\pi(k-l)F_0 t}\right]_{t_0}^{t_0+T_p}$$

Persamaan diatas akan diolah untuk k = l, maka akan didapat

$$\int_{t_0}^{T_p-t_0} dt = T_p$$

maka, persamaan (3) akan tereduksi menjadi

$$\int_{t_0}^{t_0+T_p} x(t)e^{-j2\pi lF_0 t}dt = c_l T_p$$

Maka didapat

$$c_l = \frac{1}{T_p}\int_{t_0}^{t_0+T_p} x(t)e^{-j2\pi lF_0 t}dt$$

Karena l = k, maka persamaan diatas juga dapat diubah menjadi

$$c_k = \frac{1}{T_p}\int_{t_0}^{t_0+T_p} x(t)e^{-j2\pi kF_0 t}dt \qquad (4)$$

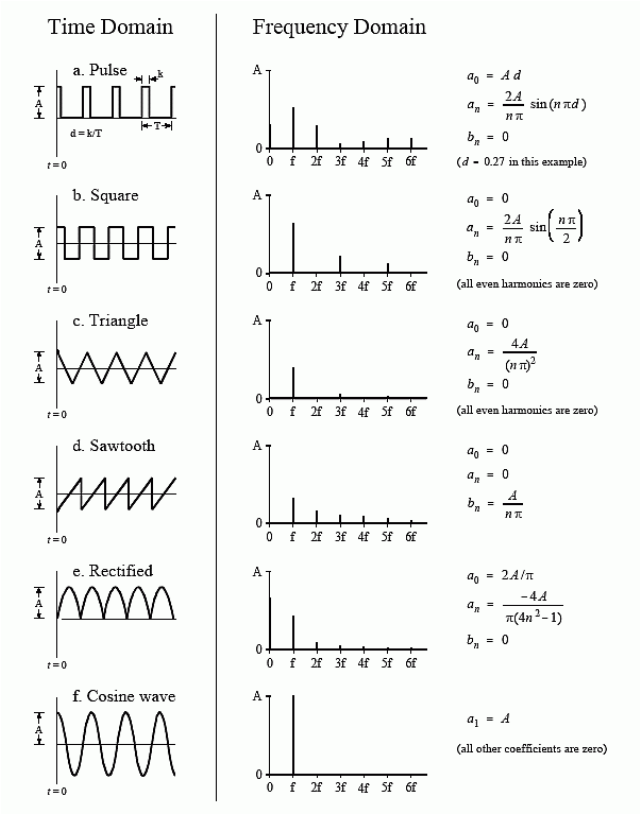
Sekarang, didefinisikan fungsi X(F), yang merupakan transformasi fourier dari x(t) sebagai

$$X(F) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi Ft}dt \qquad (5)$$

X(F) adalah fungsi kontinu dari F dan tidak bergantung pada F_0atau T_p. Jika kita bandingkan persamaan pada persamaan 4 dan 5, maka akan didapat

$$c_k = \frac{1}{T_p}X(kF_0) \qquad (6)$$

Persamaan (5) tersebut menjadi fungsi dalam domain frekuensi. Berikut ini adalah contoh-contoh gambar dari transformasi fourier



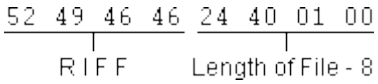
III. FAST FOURIER TRANSFORM

A. WAV FILE FORMAT

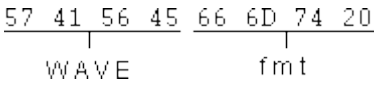
Untuk dapat melakukan steganografi pada file audio, perlu diketahui struktur dari file audio berformat WAV.

```
52 49 46 46 24 40 01 00  57 41 56 45 66 6D 74 20  RIFF$@..WAVEfmt
10 00 00 00 01 00 02 00  11 2B 00 00 44 AC 00 00  .........+..D...
04 00 10 00 64 61 74 61  00 40 01 00 00 00 00 00  ....data.@......
```

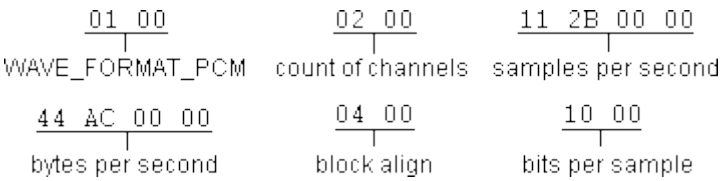
File RIFF dimulai dengan text RIFF, diikuti dengan ukuran panjang keseluruhan file



Kolom berikutnya, menyatakan RIFF mengandung wave data



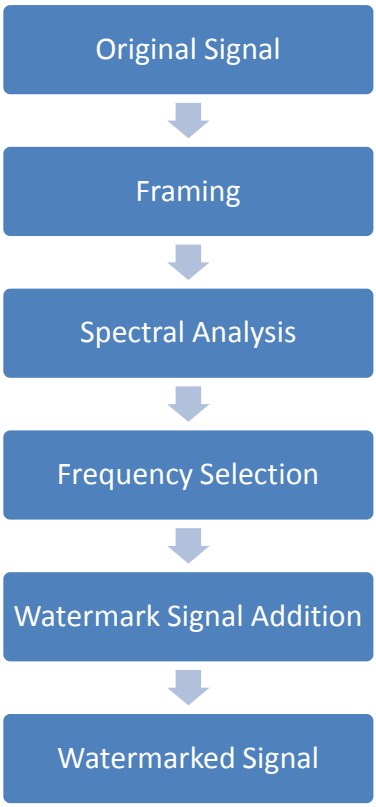
Sekarang, format terinci pada struktur "WAVEFORMATEX"



Data chunk semua sample wave. Itu menunjukkan bahwa hal tersebut merupakan audio data. Perubahan kecil pada data chunk dapat membuat sedikit perubahan, namun tidak akan menghancurkan suaranya secara signifikan.

B. SKEMA WATERMARKING

Berikut ini adalah skema dasar watermarkin pada file audio



Pada skema di atas, terlihat bahwa sinyal original, atau dalam hal ini file audio original, menjadi masukan kedalam fungsi watermark untuk kemudian diubah mencadi file audio ter-watermark-kan. Namun , sebelum diubah ke audio terwaterkark, file audio pertama-tama harus melewati bagian framing.

Framing bertujuan untuk membagi audio menjadi sejumah frekuensi dalam range tertentu. Hasil framing tersebutlah yang nantinya akan ditranformasikan dengan algoritma Fast Fourier Transform.

Setelah file audio diolah dengan FFT, kemudian diseleksi frekuensi yang akan digantikan dengan frekuensi lain. Dalam hal ini, frekuensi tersebut adalah frekuensi yang tidak terlalu signifikan terdengar oleh manusia. Setelah ditentukan yang mana yang akan digantikan, baru penyisipan frekuensi plain dilakukan pada tahap watermarking signal addition. Setelah melewati watermark signal addition,

C. FAST FOURIER TRANSFORM ALGORITHM

Persamaan pada bagian 2 merupakan persamaan dalam waktu yang kontinu. Sekarang, untuk mengoperasikan fast fourier transform, dibutuhkan transformasi fourier aperiodic dalam waktu diskrit. Persamaan tersebut digunakan dalam DFT dan IDFT. Namun, karena FFT merupakan penyempurnaan dari FFT, maka property tersebut tetap akan dimiliki.

Formula pada DFT dan IDFT didefinisikan sebagai

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}$$

dengan k = 0,1,2,3,N-1 (7)

dan

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn}$$

dengan n = 0,1,…,N (8)

dengan $W_n = e^{-j2\pi/N}$ (9)

Terdapat banyak algoritma FFT yang dapat digunakan. Salah satu pendekatannya adalah dengan menggunakan Divide And Conquer radix 2, yang artinya membagi tiap proses menjadi 2 bagian.

Untuk dapat menggunakan FFT, perlu diketahui duu prinsip kerja DFT karena FFT menggunakan DFT dalam pengomputasiannya.

Untuk mendapatkan hasil yang lebih optimal, penggunaan direct computing mulai dialihkan ke penggunaan algoritma divide and conquer.

1. Discrete Fourier Transform dengan Divide and Conquer

Misalnya, terdapat N buah komputasi DFT, koomputasi tersebut dapat dianggap sebagai hasil dari perkalian dua buah bilangan, L dan M, maka N = LM. Sejumlah data tersebut dapat direpresentasikan sebagai N kolom array 1 dimensi dengan panjang L, yang isinya x(i)

x(0)	x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(N-1)

Kemudia, data tersebut dipindahkan kedalam matrix berukuran LM, seperti berikut

l \ m	0	1	M-1
0	x(0,0)	x(0,1)	…
1	…	…	…
N-1	…	…	…

Jika ingin mengambil index yang bersesuaian dengan index pada array 1 dimensi, maka digunakan rumus dibawah ini :

$n = Ml + m$ untuk memilih berdasarkan kolom

dan $n = l + mL$ untuk memilih berdasarkan baris, namun keduanya sebenarnya sama saja. Jika data tetap dipetakan pada matrix eperti diatas, maka untuk pengomputasian dapat digunakan persamaan berikut

$$X(p,q) = \sum_{m=0}^{M-1}\sum_{l=0}^{L-1} x(l,m) W_N^{(Mp+q)(mL+l)} \quad (12)$$

Langkah pengomputasiannya adalah sebagai berikut

1. komputasikan sejumlah M DFT dengan persamaan
$$F(l,q) = \sum_{m=0}^{M-1} x(l,m) W_M^{mq}$$
dengan $0 \le q \le M-1$ (13)
2. Kemudian, komputasikan matriks baru G(l,q) yang merupakan hasil perkalian antara F(l,q) dengan W_N^{lq}, dan dapat durumuskan menjadi
$G(l,q) = W_N^{lq} F(l,q)$ (14)
3. Kemudian kalkulasikan N DFT
$$X(p,q) = \sum_{l=0}^{L-1} G(l,q) W_L^{lp} \quad (15)$$

Dari sekian banyak data yang dihasilkan pada tahap DFT diatas, kita bias memilih frekuensi mana yang akan kita pertukarkan nilainya, yang dalam hal ini nilai dalam bit yang merepresentasikan suara pada frekuensi F. Dengan menggunakan struktur data file wav seperti pada bab 3, frekuensi suara dapat diambil dan bit LSBnya dapat digantikan (jika tidak menggunakan metode spread spectrum).

Setelah hasil transformasi ke ranah frekuensi, yakni dari x(n) menjadi X(n) didapat, maka dengan menggunakan Inverse DFT atau IDFT bisa didapat kembali bentuk awal spectrum audio.

Pada metode penghitungan DFT dilakukan dengan direct computation. Idenya adalah dengan mengubah komponen komlpleks kedalam persamaan trigonometri dibawah ini

$$X_R(k) = \sum_{n=0}^{N-1}\left[x_R(n)cos\frac{2\pi kn}{N} + x_1(n)sin\frac{2\pi kn}{N}\right] \quad (10)$$

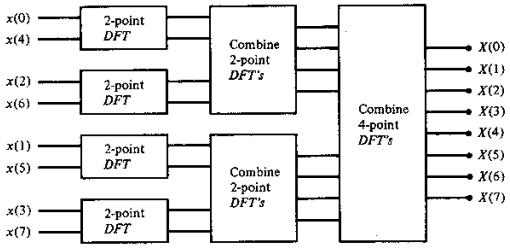
$$X_I(k) = -\sum_{n=0}^{N-1}\left[x_R(n)sin\frac{2\pi kn}{N} + x_1(n)cos\frac{2\pi kn}{N}\right] \quad (11)$$

Untuk mengomputasikan persamaan – persamaan

a. $2N^2$ penghitungan fungsi trigonometri
b. $4N^2$ perkalian real
c. $4N(N-1)$ penjumlahan real
d. Indexing

2. Radix 2 FFT

Pada pengolahan frekuensi dengan DFT diatas, pengolahan tersebut sebenarnya sudah mengaplikasikan divide and conquare. Namun, N hanya difaktorkan menjadi dua bagian, yaitu M dan L. Pada radix 2 FFT, N akan difaktorisasai menjadi sejumlah bilangan yang memiliki aturan N = $a_1\ a_2\ a_3 \ldots\ a_n$ dengan $a_1\ a_2\ a_3 \ldots\ a_n \ldots$= a, maka N = a^n, yang dalam hal ini, a merupakan radix. Bilangan a ini menentukan banyaknya pembagian operasi dalam pemrosesan dengan divide and conquer. Contoh, jika terdapat sejumlah n frekuensi yang ingin dikerjakan tiap 2 frekuensi, maka divide and conquer yang digunakan radix 2 dan akan L = 2 dan M = N/2 pada W_L^M sehingga menjadi $W_{N.2}^2$.



Pada gambar di atas, terlihat, x(0) sampai x(8) diolah secara divide and conquer. Tiap data dibagi menjadi 2 dan akhirnya dijadikan menjadi satu lagi data yang sudah diproses. Kemudian, dengan radix yang diambil adalah 2, maka persamaan 15 dapat diubah menjadi

$$X(k+N/2) = \sum_{l=0}^{L-1} F_1(k) + W_N^k F_2(k)$$

Dari persamaan di atas, direct computing yang terjadi adalah sebanyak $(\frac{N}{2})^2$ penjumlahan real dan $\frac{N}{2}$ perkalian komples pada F_1dan F_2. Jika dibandingkan dengan yang terjadi pada DFT biasa, jumlah proses yang terjadi jauh lebih sedikit.

3. Radix 4 FFT

Skema pada FFT tidak jauh berbeda dengan yang dilakukan pada FFT radix dua. Bedanya, data yang diproses dengan divide and conquer adalah tiap 4 partisi.

Jika pada FFT radix 2, radix yang digunakan adalah 2, maka pada FFT radix 4 yang digunakan adalah radix 4 yang artinya 4^n.

D. PENYISIPAN DATA

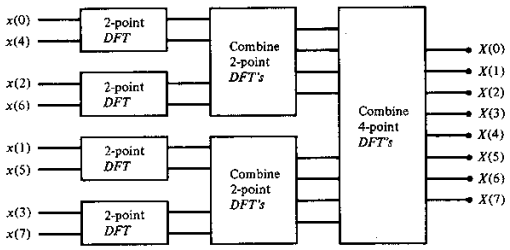
Setelah pemrosesan FFT sudah selesai dikerjakan, langkah selanjutnya adalah menyisipkan kode atau symbol rahasia kita kedalam frekuensi hasil olahan FFT. Misalkan, didapat table frekuensi olahan seperti berikut

F(0)	F(1)	F(2)	F(3)	F(4)	F(5)
F(6)	F(7)	F(8)	F(9)	F(10)	F(11)
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	…
…	…	…	…	…	F(N)

Misalkan data tersebut adalah data frekuensi yang didapat dari hasil framing yang merupakan frame pertama. Akan dipilih elemen yang memiliki nilai frekuensi paling kecil. Pencariannya dapat menggunakan bruteforce atau dengan menggunakan divide and conquer

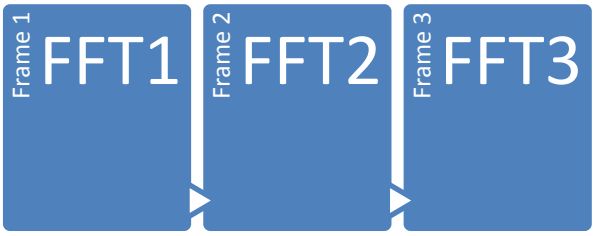
radix 2. Dibentuk table dengan jumlah NL X M data seperti yang dilakukan pada DFT dengan divide and conquer. Setelah didapat data 1 frekuensi, selanjutnya akan digantikan bit LSBnya dengan bit pada file yang ingin disembunyikan. Misalkan, data wave yang didapat, yang sudah dikonversikan ke biner, didapat 0001010101101, dan misalkan data yang ingin disembunyikan merupakan file text dengan karakter pertama yang ingin disembunyikan 011110101, maka bit dari data yang ingin disembunyikan diambil satu persatu mulai dari belakang dan digantikan pada tiap data frekuensi pada 1 frame. Jika data frekuensi pada 1 frame habis, maka pindah ke frame berikutnya.

Pemilihan data dengan algoritma divide and conquer akan ditunjukkan seperti gambar berikut:



Pertama-tama, per dua data frekuensi diambil dari table frekuensi yang sudah di FFT-kan, missal x(0) dan x(1). Setelah itum dibandingkan mana frekuensi yang lebih kecil. Kemudian, pembandingan dengan frekuensi lainpun dilakukan, sekarang x(2) dan x(3). Hasil dari pembandingan pada x(0) dengan x(1) dan x(2) dengan x(3) dibandingkan. Hasil disimpan sebagai m. Setelah itu, x(5) dan x(6) dibandingkan, hasil yang terkecil disimpan untuk kemudian dibandingkan dengan hasil pembandingan antara x(7) dan x(8). Setelah didapat yang terkecil, anggap disimpan sebagai n. Setelah itu, n dan m dibandingkan, hasil yang terkecil akan dipilih untuk diencodingkan.

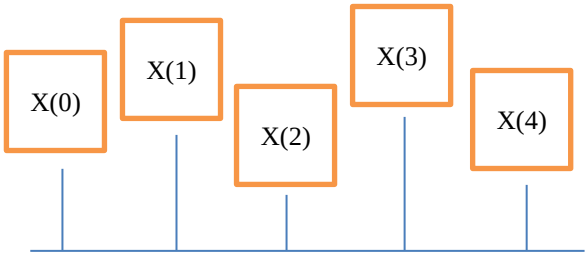
Cara tersebut berlaku untuk framing dengan panjang data tiap frame adalah 8. Yang harus dilakukan pada pembangolahan dengan FFT adalah membagi data menjadi jumlah yang merupakan factor dari 2^n. Sekarang anggap gambar di atas adalah penyeleksian dari satu frame. Jika terdapat banyak frame, maka akan terdapat banyak penyeleksian seperti cara di atas.



Kemudian, untuk semua X(0) sampai X(n), yakni nilai yang didapat dari transformasi fourier menggunakan persamaan DFT (persamaan 7), maka nilai tersebut harus ditransformasikan ke bentuk semula dengan menggunakan Inverse DFT. Misalkan, terdapat sejumlah n data yang keluarannya dalam bentuk decimal, maka bisa dibuat fungsi

```
function doDFT(input:wave data) {
    for i=0 to length of wave data{
        transform each wave data with DFT formula
    }
    → transformed wave data
}
```

Kemudian, dari transformasi diatas, akan didapat spectrum berbentuk garis-garis yang merepresentasikan amplitude dari tiap-tiap periode.



Tiap garis tersebut merupakan data array yang dapat diambil nilainya dan diolah.

Kemudian, setelah nilai – nilai tersebut disisipkan kode rahasia, nilai X(0), X(1), X(2), X(3), X(4),….X(n) tersebut harus dikembalikan kebentuk awal, takni x(n) dengan menggunakan persamaan IDFT. Spectrum hasil pembalikan dengan IDFT akan mendekati bentuk awalnya, namun akan sedikit berbeda karena efek dari penggantian bit dengan kode rahasia.

Dengan menggunakan IDFT, semua data akan diiterasi satu persatu untuk diubah menggunakan IDFT

```
function doIDFT(input:wave data) {
    for i=0 to length of wave data{
        transform each wave data with IDFT formula
    }
    → transformed wave data to normal form
}
```

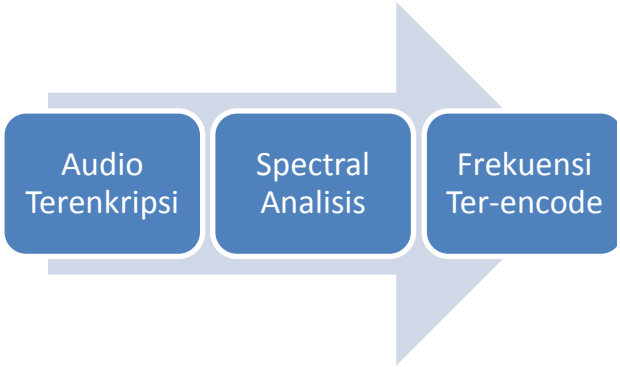
E. DEKRIPSI DATA

Pada mode enkripsi, data yang digantikan adalah bit LSB pada frekuensi terkecil dari tiap frame. Maka pengekstrakan informasi dapat dilakukan dengan mengomputasikan FFT pada file suara yang sudah diencode-kan dan diambil frekuensi yang paling kecilnya. Dengan cara tersebut akan didapat sekumpulan data frekuensi terkecil dari tiap frame yang sudah disisipkan dengan kode rahasia.

Misalnya, terdapat table frekuensi yang sudah terseleksi sebagai hasil dari FFT dan memiliki nilai frekuensi terkecil

F(0)	F(1)	F(2)	F(3)	F(4)
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
…	…	…	…	F(n)

Hasil dari table disamping adalah hasil yang didapat dari persamaan 13. F(0) sampai F(n) bukan berasal dari posisi yang berurutan pada file audio asli. F(0) bisa saja terdapat di frekuensi ke m. Namun, dengan menggunakan persamaan yang sama seperti ketika melakukan proses FFT, data yang sama akan berhasil didapat lagi. Proses pengekstrakan data dapat digambarkan seperti berikut ini



Sama seperti pada tahap penyisipan data, untuk mentransformasikan ke dalam ranah frekuensi digunakan persamaan DFT dan untuk mengembalikan ke ranah spasial digunakan IDFT.

Algoritma yang digunakan juga mirip dengan yang terjadi pada penyisipan data. Perbedaannya hanya pada pengekstrakan data, bit paling belakang diambil untuk kemudian dikumpulkan menjadi satu.

IV. KESIMPULAN

Pada enkripsi data di file audio, model steganografi juga dapat dilakukan dengan menggunakan bit LSB. Namun, jika penyebarannya hanya disekitar frekuensi yang memiliki nilai terkecil, ketika file audio tersebut di crop, maka data akan hilang jika frekuensi yang bersesuaian ada di dalam area yang di crop tersebut. Untuk mengatasi hal tersebut bias digunakan metode spread spectrum.

Penyisipan data kedalam audio dapat dilakukan, namun akan menimbulkan sedikit noise karena ada nilai yang sedikit berubah. Namun hal tersebut tidak terlalu terdengar karena penyeleksian data frekuensi tiap sample dipilih frekuensi yang terkecil. Namun kekurangan dari tehnik ini, jika pada suatu sample, nilai frekuensi terkecil masih termasuk frekuensi dominan yang dapat didengar dengan jelas, maka noise yang ditimbulkan pun akan sedikit bertambah besar. Bunyi desisan akan muncul pada file audio ter-encode dengan cara ini.

Hal lain tentang steganografi dengan audio adalah, pengolahan sinyal audio dengan DFT biasa akan berdampak pada konsumsi waktu pada saat kompilasi. Penerapan FFT menjadi solusinya, meskipun FFT sendiri punya tingkatan hirarkis berdasarkan kecepatan kompilasi seperti FFT radix 2, radix 4, dan split radix. Hal tersebut dikarenakan pengolahan DFT standard hanya berdasarkan direct computing yang memerlukan banyak penghitungan real didalamnya.

Berdasarkan penghitungan jumlah perkalian dan penjumlahan kompleks, metode FFT paling tidak yang menggunakan radix 2 jika dibandingkan dengan FFT biasa memiliki jumlah proses yang lebih banyak karena dalam direct computingnya, yang dikerjakan hanya

- $(\frac{N}{2})^2$ penghitungan fungsi trigonometri
- $2(\frac{N}{2})^2$ perkalian real

Jadi, operasi penjumlahan dan pengalian bilangan real lebih sedikit.karena jumlah penghitungan penambahan real sebanyak $(3N/2)log_2N$ dan $(3N/8)log_2N$.

REFERENCES

Proakis John G, “Digital Signal Processing,”. Edition 2, J. New Jersey: Prentice Hall, 1996, pp. 212–484.

Whttp://www.relisoft.com/science/physics/sampling.html

http://paulbourke.net/miscellaneous/dft/

http://www.ece.uvic.ca/~aupward/w/watermarking.htm

http://www.codeproject.com/KB/security/steganodotnet8.aspx

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Maret 2011

ttd

Otniel 13508108