

Bab II

Landasan Teori

2.1. Pengantar

Bab ini berisi landasan teori perancangan perangkat lunak SSAWW, meliputi : pengertian *Wave* audio, pengertian *Watermarking*, pengertian *Spread Spectrum*, Serangan (*Attack*), dan *Binary String*.

2.2. Pengertian Wave Audio

2.2.1. Format Wave

Format *wave* adalah format penyimpanan media digital audio yang diciptakan dan dipakai oleh perusahaan perangkat lunak besar dunia yaitu Microsoft dan IBM. Format *wave* merupakan variant dari format *bit-stream RIFF* yang digunakan oleh Microsoft, dan untuk Macintosh format *wave* dikenal sebagai *IFF* yang merupakan variant dari format *bit-stream AIFF*. Format *bit-stream RIFF* dan *AIFF* adalah suatu metode yang menyimpan data audio kedalam *chunks*, dengan kata lain data audio yang disimpan dalam format *wave* dan *IFF* akan kompatibel jika digunakan dalam sistem operasi Microsoft maupun Macintosh (en.wikipedia.org, 1999).

Data digital audio dalam file *wave* bisa memiliki kualitas yang bermacam-macam. Kualitas dari suara yang dihasilkan ditentukan dari *bit rate*, *sample rate*, dan jumlah *channel*. *Bit rate* merupakan ukuran bit tiap sampelnya, bisa berjumlah 8 bit atau 16 bit. Dalam 8 bit *wave* setiap sampelnya hanya ada 1 byte saja. Sedangkan untuk 16 bit setiap sampelnya ada 2 byte.

Wave 16 bit setiap sampelnya memiliki nilai antara 32768 sampai 32767, dibandingkan dengan 8 bit yang hanya memiliki nilai antara 0 sampai 255 maka 16 bit wave menghasilkan suara yang lebih baik karena datanya lebih akurat daripada 8 bit. Namun, pada kenyataannya telinga manusia umumnya tidak mampu membedakan suara yang dihasilkan antara file wave 16 bit dengan file wave 8 bit.

Sample rate menyatakan banyaknya jumlah sampel yang dimainkan setiap detik. *Sample rate* yang umum dipakai adalah 8000Hz, 11025Hz, 22050Hz, dan 44100Hz. Untuk 8000Hz biasanya disebut sebagai *telephone quality* karena suara yang dihasilkan menyerupai suara dari telepon. *Sample rate* 11025Hz sering digunakan untuk merekam suara manusia karena dinilai sudah cukup. Tetapi untuk musik agar dapat memperoleh hasil yang baik maka digunakan *sample rate* 22050Hz. Sedangkan 44100Hz merupakan *CD quality*, *sample rate* 44100Hz inilah yang digunakan dalam audio CD karena bagus untuk semua jenis suara. Berbeda dengan *bit rate*, perbedaan *sample rate* dapat dikenali dengan mudah, karena tentu saja suara dari radio dengan suara CD player sangat jauh berbeda.

Jumlah *channel* menentukan suara yang dihasilkan apakah *mono* atau *stereo*. *Mono* hanya memiliki 1 *channel*, sedangkan *stereo* 2 *channel* dan memakan tempat 2 kali lebih banyak daripada *mono*. Untuk merekam suara manusia, *mono* sudah cukup memberikan kualitas yang baik, sedangkan *CD quality* menggunakan 2 *channel* (www.sonicspot.com, 1999).

2.2.2. Struktur Format Wave

Format wave menggunakan struktur standar RIFF (*Resource Interchange File Format*), struktur ini mengatur data dalam file ke dalam bagian-bagian yang masing-masing memiliki *header* dan ukurannya sendiri dan disebut sebagai *chunks*. Struktur ini memungkinkan bagi program bila tidak mengenali bagian tertentu untuk melompati bagian tersebut dan terus memproses bagian yang dikenal. Contoh file yang menggunakan struktur RIFF adalah file Wave dan AVI (www.borg.com, 1999).

Sesuai dengan struktur file RIFF, file wave diawali dengan 4 byte yang berisi 'RIFF' lalu diikuti oleh 4 byte yang menyatakan ukuran dari file tersebut yang nilainya sama dengan ukuran dari file dikurangi 8. Dan 4 byte lagi yang berisi 'WAVE' yang menyatakan bahwa file tersebut adalah file WAVE. Untuk lebih jelasnya lihat tabel 2.1.

Tabel 2.1. Detail Bagian RIFF

Offset	Size	Description	Value
0x00	4	Chunk ID	"RIFF" (0x52494646)
0x04	4	Chunk Data Size	(file size) - 8
0x08	4	RIFF Type	"WAVE" (0x57415645)
0x0c	Wave chunk		

(www.sonicspot.com, 1999)

Setelah itu barulah informasi format dan data dari file wave disimpan. Bagian format sampel berisi informasi-informasi mengenai bagaimana data disimpan dan memainkannya. Bagian ini dimulai dengan ID 'fmt ', lalu diikuti dengan 4 byte yang merupakan panjang dari informasi dan bernilai 16 untuk PCM. Kompresi kode

menempati 2 byte berikutnya dengan nilai 1 untuk PCM. Dua byte berikutnya menyatakan jumlah *channel* dari file *wave*, lalu 4 byte menyatakan *sample rate* dan 4 byte lagi menyatakan rata-rata byte tiap detiknya. Dua byte setelahnya merupakan *Block align* yang menyatakan ukuran data untuk satu sampel penuh dalam byte. Yang dimaksud dengan satu sampel penuh adalah satu sampel yang mewakili nilai dari sampel pada semua *channel* pada suatu waktu. Dua byte terakhir dari bagian sampel format ini menyatakan *bit rate* dari data yang disimpan, bernilai 8 atau 16. Detailnya ada di tabel 2.2.

Tabel 2.2. Detail Bagian Sampel Format

Offset	Size	Description	Value
0x00	4	Chunk ID	"fmt (0x666D7420)"
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compresion code	1 - 65,535
0x0a	2	Number of channels	1 - 65,535
0x0c	4	Sampel rate	1 - 0xffffffff
0x10	4	Average bytes per second	1 - 0xffffffff
0x14	2	Block align	1 - 65,535
0x16	2	Significant bit per sampel	2 - 65,535

(www.sonicspot.com, 1999)

Bagian berikutnya adalah bagian data audio. Di bagian inilah sampel digital audio disimpan. Bagian ini dimulai dengan ID 'data' dan diikuti dengan 4 byte yang menyatakan besarnya data dalam byte, lalu selebihnya

adalah data digital audio-nya. Detailnya ada di tabel 2.3.

Tabel 2.3. Detail Bagian Data Audio

Offset	Size	Description	Value
0x00	4	Chunk ID	"data"(0x64617461)
0x04	4	Chunk Data Size	Tergantung dari panjangnya sampel data
0x0c	Wave chunk		

(www.sonicspot.com, 1999)

2.3. Pengertian Watermarking

Watermarking (tanda air) merupakan suatu bentuk dari steganografi (Ilmu yang mempelajari bagaimana menyembunyikan suatu data pada data yang lain). Dalam mempelajari teknik-teknik penyimpanan suatu data digital kedalam data host digital yang lain (Istilah *host* digunakan untuk data/sinyal digital yang ditumpangi), *watermarking* ini agak berbeda dengan tanda air pada uang kertas. Tanda air pada uang kertas masih dapat kelihatan oleh mata telanjang manusia (mungkin dalam posisi kertas yang tertentu), tetapi *watermarking* pada media digital disini dimaksudkan tak akan dirasakan kehadirannya oleh indera manusia tanpa alat bantu mesin pengolah digital seperti komputer, dan sejenisnya.

Watermarking memanfaatkan kekurangan-kekurangan sistem indera manusia seperti mata dan telinga. Dengan adanya kekurangan inilah, metode *watermarking* dapat diterapkan pada berbagai media digital. Jadi *watermarking* merupakan suatu cara untuk penyembunyian

atau penanaman data/informasi tertentu (baik hanya berupa catatan umum maupun rahasia) ke dalam suatu data digital lainnya, tetapi tidak diketahui kehadirannya oleh indera manusia (indera penglihatan atau indera pendengaran), dan mampu menghadapi proses-proses pengolahan sinyal digital sampai pada tahap tertentu (Suhono, et all, 2000).

Watermarking sebagai suatu teknik penyembunyian data pada data digital lain dapat dimanfaatkan untuk berbagai tujuan, seperti :

1. *Tamper proofing*, *watermarking* digunakan sebagai alat untuk mengidentifikasikan atau alat indikator yang menunjukkan data digital (*host*) telah mengalami perubahan dari aslinya.
2. *Feature location*, menggunakan *watermarking* sebagai alat untuk identifikasikan isi dari data digital pada lokasi-lokasi tertentu, seperti contohnya penamaan objek tertentu dari beberapa objek yang lain pada suatu citra digital.
3. *Annotation/caption*, *watermarking* hanya digunakan sebagai keterangan tentang data digital itu sendiri.
4. *Copyright Labeling*, *watermarking* dapat digunakan untuk menyembunyikan label hak cipta pada data digital sebagai bukti otentik kepemilikan karya digital tersebut.

Dalam dua tahun terakhir terjadi perkembangan yang sangat pesat dalam dunia *watermarking* karena ada beberapa faktor pendukung. Pertama, industri penerbitan (*publishing*) dan penyiaran (*broadcasting*) mulai

tertarik pada teknik untuk menyembunyikan tanda hak cipta terenkripsi dalam nomor seri pada film digital, rekaman audio, buku, dan produk multimedia. Kedua, *watermarking* juga dapat digunakan pada pengawasan otomatis (*automatic monitoring*) iklan radio, hal ini menjadi sangat efektif untuk melakukan pengawasan bahwa iklan diputar sebanyak kontrak yang telah disepakati bersama. Ketiga, pada dunia kedokteran yang mulai menggunakan *watermarking* untuk menanamkan informasi, misal : nama, alamat, tanggal, dan nama dokter yang memeriksa.

Ada beberapa parameter yang perlu diperhatikan dalam penerapan metoda *watermarking*, yaitu :

1. Jumlah data (*bit rate*) yang akan disembunyikan.
2. Ketahanan (*robustness*) terhadap proses pengolahan sinyal.
3. Data yang disisipkan tidak tampak (*Invisiblity*).

Terdapat suatu *trade off* (nilai balik) diantara kedua parameter (*bit rate* dan *robustness*) tersebut dengan *Invisiblity*. Bila diinginkan *robustness* yang tinggi dan *Invisiblity* maka *bit rate* akan menjadi rendah, sedangkan akan semakin *visible* dan *robustness* akan menurun jika *bit rate* tinggi. Jadi harus dipilih nilai-nilai dari parameter tersebut agar memberikan hasil yang sesuai dengan keinginan. Hubungan *Invisiblity* dengan *Robustness* dapat diterangkan sebagai berikut : misalkan suatu data asli disisipi sesedikit mungkin informasi dengan maksud memberikan efek *invisible* yang semakin tinggi, maka dengan adanya sedikit proses pengolahan digital saja, perubahan tadi

akan berubah/hilang. Dengan demikian dikatakan *robustness* rendah, tetapi *invisibility* tinggi.

Watermarking dalam penerapannya terhadap data digital dapat diterapkan pada berbagai kawasan (domain). Misalnya untuk citra dan video pada kawasan spasial, dan audio pada kawasan waktu. atau terlebih dahulu dilakukan transformasi ke dalam kawasan yang lain. Berbagai transformasi yang dikenal dalam pemrosesan sinyal digital seperti:

1. FFT (*Fast Fourier Transform*),
2. DCT (*Discrete Cosine Transform*),
3. DWT (*Discrete Wavelet Transform*).

Penerapan *watermarking* pada berbagai kawasan dengan berbagai transformasi turut mempengaruhi berbagai parameter penting dalam *watermarking* (*bitrate*, *invisible*, dan *robustness*).

2.4. Pengertian Spread Spectrum

Lahirnya sistem komunikasi *spread spectrum* pada pertengahan tahun 1950 dilatarbelakangi oleh kebutuhan akan sistem komunikasi yang dapat mengatasi masalah interferensi, dapat menjamin kerahasiaan informasi yang dikirim dan dapat beroperasi pada tingkat S/N (*signal to noise ratio*) yang rendah atau tahan terhadap derau yang besar. Dalam sistem komunikasi sekarang ini, dimana penggunaan frekuensi sudah cukup padat sehingga interferensi dan *noise* dari transceiver lain cukup besar. Dalam komunikasi radio sering terjadi penyadapan pembicaraan pada pesawat radio oleh pesawat radio lain.

Namun dengan sistem *spread spectrum* ketakutan yang dialami pada sistem komunikasi diatas akan dapat di atasi karena data yang ditransmit pada sistem *spread spectrum* adalah data acak yang dikenal sebagai noise. Jadi jika penerima tidak mengetahui code yang digunakan untuk melebarkan data maka penerima hanya akan menerima sinyal noise saja. Istilah *spread spectrum* digunakan karena pada sistem ini sinyal yang ditransmisikan memiliki *bandwidth* yang jauh lebih lebar dari *bandwidth* sinyal informasi (mencapai ribuan kali). Proses penebaran *bandwidth* sinyal informasi ini disebut *spreading*.

Kode yang digunakan pada sistem *spread spectrum* memiliki sifat acak tetapi periodik sehingga disebut sinyal acak semu (*pseudo random*). Kode tersebut bersifat sebagai noise tapi deterministik sehingga disebut juga noise semu (*pseudo noise*). Pembangkit sinyal kode ini disebut *Pseudo Random Generator* (PRG) atau *Pseudo Noise Generator* (PNG). PRG inilah yang akan melebarkan dan sekaligus mengacak sinyal data yang akan dikirimkan. Dalam komunikasi *spread spectrum* semakin lebar *bandwidth* akan semakin tahan terhadap *jamming* dan akan semakin terjamin tingkat kerahasiaannya (Bender, et all, 1996).

Jika diterapkan dalam audio *watermarking* teknik *spread spectrum* akan menyebarkan seluruh sinyal/bit-bit data yang akan disisipkan kedalam kawasan frekuensi. Dalam proses penyebarannya diatur oleh *pseudo random* dengan kunci berupa *password* yang dimasukkan oleh user, sehingga jika ingin merekonstruksi kembali bit-bit data yang disisipkan membutuhkan *password* yang sama pada

saat penyisipan. Di kawasan frekuensi, bit *watermark* akan menyebar pada seluruh koefisien dan mengakibatkan adanya tambahan *noise* di beberapa kawasan frekuensi setelah proses penyisipan data, tetapi distorsi (*noise*) diusahakan tidak dapat diketahui/diperhatikan oleh *Human Auditory System* (HAS).

Untuk menerapkan teknik *spread spectrum* dibutuhkan dua metode yang akan dijelaskan dibawah ini.

2.4.1. Metode *Fast Fourier Transform* (FFT)

FFT merupakan penyederhanaan dari *Discrete Fourier Transform* (DFT), yang digunakan untuk mendapatkan kawasan frekuensi dari sinyal waktu diskrit $x(n)$. Dengan FFT, maka jumlah titik data N merupakan bilangan yang dapat difaktorkan sehingga seluruh jumlah titik FFT dapat dipecah ke dalam kelompok-kelompok yang makin lama makin kecil. FFT dihitung menggunakan persamaan :

$$X(q) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi qn/N} \quad (2.1)$$

Karena terdapat nilai e^{-jx} atau e^{jx} maka akan menghasilkan dua nilai, yaitu : nilai *Real* dan nilai *Imaginer* dan dalam persamaan **Euler** dinyatakan :

$$e^{-jx} = \cos x - j \sin x \quad (2.2)$$

$$e^{jx} = \cos x + j \sin x \quad (2.3)$$

Berdasarkan persamaan Euler diatas maka persamaan (2.1) bisa disederhanakan menjadi :

$$X(q) = \sum_{n=0}^{N-1} x(n)(\cos 2\pi qn / N - j \sin 2\pi qn / N) \quad (2.4)$$

Untuk nilai real :

$$\text{Re}(q) = \sum_{n=0}^{N-1} x(n) \cos 2\pi qn / N \quad (2.5)$$

Untuk nilai imajiner :

$$\text{Im}(q) = -j \sum_{n=0}^{N-1} x(n) \sin 2\pi qn / N \quad (2.6)$$

Pada teknik *spread spectrum* proses penyisipan bit-bit data dilakukan pada kawasan frekuensi. Maka setelah mendapatkan nilai *Real* dan nilai *Imajiner* kemudian kedua nilai tersebut digunakan untuk mencari nilai frekuensi yang sebenarnya dengan menggunakan persamaan:

$$F(q) = \sqrt{\text{Re}(q)^2 + \text{Im}(q)^2} \quad (2.7)$$

Setelah dilakukan penyisipan bit-bit data pada frekuensi kemudian nilai-nilai frekuensi tersebut akan dikembalikan ke kawasan waktu $x(q)$ dengan menggunakan metode Invers FFT (IFFT), dengan kata lain IFFT adalah kebalikan dari FFT. Pada IFFT dibutuhkan nilai *Real* dan *Imajiner* baru yang mencerminkan nilai frekuensi yang telah disisipi, untuk mendapatkan nilai *Real* dan *Imajiner* tersebut menggunakan persamaan :

$$\text{Re}(q) = F(q) \cos \theta(q) \quad (2.8)$$

$$\text{Im}(q) = F(q) \sin \theta(q) \quad (2.9)$$

Pada persamaan (2.8) dan (2.9) terdapat variabel θ (theta). Dalam hal ini θ digunakan untuk menentukan sudut yang terbentuk dari nilai *Real* dan *Imaginer* sebelum proses penyisipan, tujuannya supaya nilai *Real* dan *Imaginer* baru berada pada sudutnya semula. Untuk mencari θ digunakan persamaan :

$$\theta(q) = \arctg \frac{\text{Im}(q)}{\text{Re}(q)} \quad (2.10)$$

Setelah mendapatkan θ , nilai tersebut harus dikonversi berdasarkan kuadrannya. Kuadran-kuadran yang ada, yaitu:

1. Kuadran I :

Kondisi ini terpenuhi jika nilai real dan imajiner positif.

Nilainya : $\theta = 0 - 90^\circ$

2. Kuadran II :

Kondisi ini terpenuhi jika nilai real negatif dan nilai imajiner positif.

Nilainya : $\theta = 180^\circ - \theta$

3. Kuadran III :

Kondisi ini terpenuhi jika nilai real dan nilai imajiner negatif.

Nilainya : $\theta = 180^\circ + \theta$

4. Kuadran IV :

Kondisi ini terpenuhi jika nilai real positif dan nilai imajiner negatif.

Nilainya : $\theta = -\theta$

Setelah didapat nilai *Real* dan *Imaginer* baru, kemudian kedua nilai tersebut dimasukkan kedalam persamaan IFFT untuk mendapatkan sinyal waktu diskret $X(q)$. IFFT dihitung dengan menggunakan persamaan:

$$X(q) = \frac{1}{N} \sum_{n=0}^{N-1} \text{Re}(n) * (\cos 2\pi qn / N) + \text{Im}(n) * (\sin 2\pi qn / N) \quad (2.11)$$

Proses FFT dilakukan secara bertahap yang dibagi menjadi blok-blok, jumlah sampel yang diambil perbloknya harus kelipatan pangkat dua (minimal empat) dan setiap sampel berukuran satu byte. Pada tugas akhir ini jumlah sampel yang diambil sebanyak delapan (2^3) buah perbloknya, karena jika mengambil jumlah sampel terlalu banyak akan terlalu beresiko karena *noise* yang dihasilkan akan semakin terdengar dan beban komputasi akan semakin berat, dan apabila jumlah seluruh sampel dalam file audio kurang dari jumlah sampel yang diambil maka kekurangan tersebut akan diset dengan nol, dan ini mengakibatkan bertambahnya data, durasi waktu, dan *noise*.

2.4.2. Metode *Pseudo Random*

Pseudo random adalah suatu metode untuk menghasilkan urutan bit-bit secara acak. Di hampir semua aplikasi kriptografi dan steganografi membutuhkan bit-bit acak yang digunakan sebagai kunci. *Pseudo random* memiliki sifat acak tetapi periodik sehingga disebut sinyal acak semu (Wade Trape, et all, 2002).

Metode ini membutuhkan parameter yang digunakan sebagai kunci untuk menghasilkan urutan byte acak. Parameter tersebut berupa *password* yang dimasukkan oleh

user. Terdapat beberapa tahap untuk mendapatkan byte acak, yaitu :

1. Mengubah inputan *password* ke dalam byte-array K(0 sampai 255). Byte-array K diisi dengan byte *password* per karakter sampai karakter terakhir dan diulangi dari awal karakter sehingga seluruh byte-array K terisi semua.

Inisialisasi :

For i = 0 to 255

K[i] = (i mod *password*)

2. Proses pembuatan S-Box yang merupakan permutasi dari bilangan 0 sampai 255. S-Box diisi secara berurutan, contoh : $S_0=0$, $S_1=1$, ..., $S_{255}=255$.

Inisialisasi :

For i = 0 to 255

S[i] = i

3. Pembangkitan kunci menggunakan S[i] dan K[i].

Inisialisasi :

j=0

For i = 0 to 255

S[j] = (j+S[i]+K[i]) mod 255

Swap(S[i], S[j])

4. Pembangkitan *random* byte menggunakan S[i] yang akan digunakan sebagai *byte-key*.

Inisialisasi :

i=0, j=0

i=(i+1) mod 255

j=(j+S[i]) mod 255

Swap(S[i], S[j])

Key=S(S[i]+S[j]) mod 255

5. Pada proses *Encode* data *hidden* akan di Xor dengan *byte-key* untuk mendapatkan data *hidden* acak. Dan pada proses *Decode* data *hidden* acak akan di Xor dengan *byte-key* untuk mendapatkan data *hidden* awal.
- Inisialisasi :

Encode => DataHiddenAcak = DataHidden Xor Key

Decode => DataHidden = DataHiddenAcak Xor Key

2.4.3. Proses *Encode* dan *Decode*

Secara garis besar ada dua proses penting dalam aplikasi ini, yaitu :

1. Proses *Encode*, adalah proses untuk menyisipkan data *hidden* kedalam file audio *host*. Proses ini terdiri dari beberapa langkah, yaitu :

- a. FFT

Adalah proses untuk mengubah file audio *host* dari kawasan waktu menjadi kawasan frekuensi. Setelah diubah menjadi kawasan frekuensi bit-bit data *hidden* baru dapat disisipkan.

- b. Konvert dari byte ke bit

Adalah proses untuk mengubah data *hidden* menjadi bit-bit. Data *hidden* harus diubah dahulu kedalam bit-bit karena proses penyisipan dilakukan per bit bukan per byte.

- c. *Pseudo Random*

Adalah proses untuk mengacak bit-bit data *hidden*, yang bertujuan supaya data *hidden* sulit untuk ditebak. Proses ini membutuhkan suatu kunci agar bit-bit data *hidden* acak membentuk suatu pola dan dapat direkonstruksi kembali.

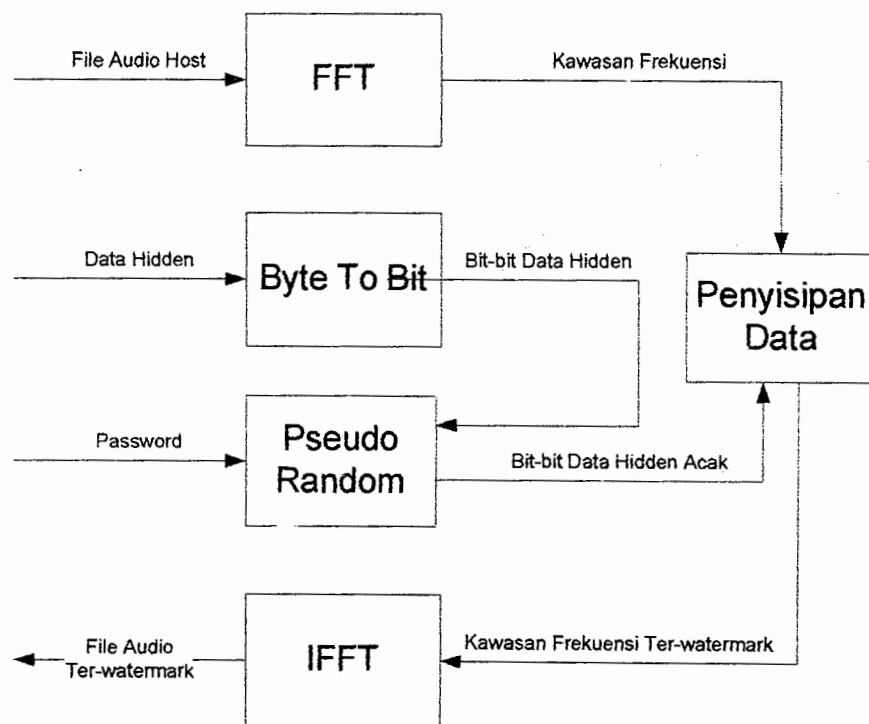
Kunci yang digunakan adalah *password* yang dimasukan oleh user.

d. *Penyisipan Data*

Inti dari fungsi *Encode* ada pada proses ini, pada proses ini kawasan frekuensi dari file *host* akan disisipi bit-bit data *hidden* acak. Setiap satu blok kawasan frekuensi hanya bisa disisipi dua bit saja, yaitu diawal blok dan ditengah blok.

e. *IFFT*

Setelah proses penyisipan data sukses dilakukan maka kawasan frekuensi harus diubah lagi kedalam bentuk kawasan waktu supaya file audio bisa dijalankan. Proses ini merupakan proses *invers* dari proses *FFT*.



Gambar 2.1. Proses Encode

2. Proses *Decode*, adalah proses untuk menyaring kembali data *hidden* yang disisipkan sebelumnya kedalam file audio *host*. Proses ini terdiri dari beberapa langkah, yaitu :

a. *FFT*

Seperti pada proses *Encode* pada proses *Decode* juga membutuhkan proses *FFT* yang digunakan untuk mengubah file audio dari kawasan waktu menjadi kawasan frekuensi supaya bit-bit yang disisipkan sebelumnya bisa dibaca kembali.

b. *Penyaringan Data*

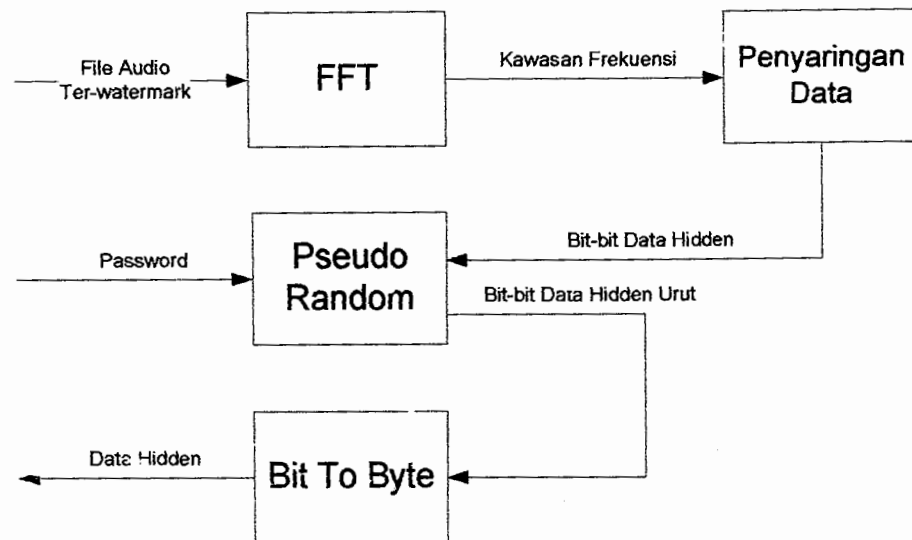
Proses ini merupakan inti dari fungsi *Decode*, pada proses ini sistem akan mengambil kembali bit-bit yang disisipkan didalam kawasan frekuensi.

c. *Pseudo Random*

Karena hasil dari proses penyaringan data merupakan bit-bit data *hidden* yang masih acak, maka bit-bit data *hidden* tersebut harus diurutkan kembali dengan menggunakan pola *Pseudo Random* yang sama pada saat proses *Encode*.

d. *Konvert dari bit ke byte*

Setelah membentuk bit-bit data *hidden* urut maka bit-bit data *hidden* tersebut harus diubah lagi kedalam bentuk byte supaya data *hidden* dapat dibaca kembali oleh user.



Gambar 2.2. Proses Decode

2.5. Attack (Serangan)

Attack/serangan merupakan cara-cara untuk menghilangkan *watermark* yang telah disisipkan. *Attack* berhubungan erat dengan *robutness*, jika semakin tahan terhadap *attack* maka *robutness* dianggap tinggi. Ada beberapa jenis serangan terhadap *watermarking*, yaitu :

1. Active Attack

Merupakan serangan dimana seseorang berusaha untuk menghilangkan *watermark* yang terdapat didalam file *ter-watermark*.

2. Passive Attack

Berbeda dengan *active attack*, pada *passive attack* hanya untuk mengetahui apa isi *watermark* tersebut, jika memang ada didalam data *ter-watermark*.

3. Forgery Attack

Serangan ini tidak hanya bertujuan untuk membaca atau menghilangkan *watermark* yang ada, tetapi juga menanamkan suatu *watermark* yang baru kedalam suatu

file host. Serangan ini cukup menjadi perhatian yang serius terutama untuk aplikasi bukti kepemilikan.

2.6. Binary Digit

Binary digit merupakan unit informasi terkecil yang ditangani oleh sebuah komputer. Satu bit diwujudkan sebagai 1 atau 0 dalam mode numeral, atau perwujudan dari true atau false sebagai kondisi logika. Sebuah grup dengan 8 bit akan disebut dengan satu byte. Dalam bentuk byte akan memberikan banyak tipe informasi, sebagai contoh surat yang ditulis dengan alphabet, digit desimal atau karakter lain. Binary digit sering disingkat dengan bit (Help and Support, 2001).