# Bab 2

# Landasan Teori

Bab ini menjabarkan teori-teori yang kami gunakan , sebagai landasan untuk membuat aplikasi ini, dan juga sejarah dibalik perkembangannya. Dimulai dari kriptografi, sebagai landasan untuk enkripsi, algoritma enkripsi *Blowfish*, sejarah pengenalan suara, sinyal analog, sinyal digital, dan pengenalan pembicara (*speaker recognition*).

## 2.1 Kriptografi

Kriptografi menurut Burton (2002, p144) adalah suatu teknik atau metode yang bertujuan untuk melindungi informasi yang dikirimkan melalui jaringan komunikasi publik.

Proses kriptografi dilakukan dengan memberikan suatu nilai untuk setiap satuan informasi terkecil, lalu juga membutuhkan variabel lain yang disebut kunci enkripsi (*encryption key*) untuk melakukan transformasi pergeseran terhadap informasi yang akan dienkripsi (Denning, 1983, p1).

Kriptanalisis (*cryptanalysis*) merupakan ilmu dan seni pembongkaran data, informasi atau pesan. Kriptologi adalah paduan dari kriptografi dan kriptanalisis. Fungsi-fungsi yang mendasar dalam kriptografi adalah enkripsi dan dekripsi. Enkripsi adalah suatu proses mengubah pesan asli (*plain text*) menjadi suatu pesan dalam bahasa sandi (*cypher text*). Sedangkan dekripsi adalah proses mengubah pesan dalam suatu

bahasa sandi menjadi bahasa asli kembali. *Cryptanalyst* adalah seorang pakar kriptografi yang mempelajari atau mengamati suatu *cyphertext* dan pada umumnya bermaksud untuk membongkar *ciphertext* tersebut.

### 2.1.1 Sejarah Singkat

Enkripsi bukanlah barang baru dalam memproteksi suatu informasi. Enkripsi sudah dimulai digunakan sejak tahun 1900 SM, yaitu berupa surat rahasia dimana hanya orang tertentu yang dituju mengetahui cara membacanya.

Salah satu yang dikenal pada saat itu adalah 'Julius Cipher' atau 'Caesar Cipher' (Stallings, 1995, p29), enkripsi sederhana yang dibuat oleh Julius Caesar untuk mengirimkan surat rahasianya, yang nantinya dikenal juga dengan *shift transformation cryptosystem*, yaitu sistem enkripsi dengan cara menggeser huruf-huruf dalam setiap pesannya sejauh tiga huruf.

### 2.1.2 *Symmetric dan Asymmetric Cryptosystem*

*Cryptography* dibagi menjadi dua jenis berdasarkan jumlah kunci yang dipakai, yaitu *symmetric cryptosystem* dan *asymmetric cryptosystem*. *Symmetric cryptosystem* adalah *cryptosystem* yang menggunakan satu kunci enkripsi yang sama untuk proses enkripsi maupun dekripsi (Columbitect, 2001). Contohnya adalah *Shift Transformation Cryptosystem*.

*Asymmetic cryptosystem* adalah *cryptosystem* yang menggunakan dua kunci enkripsi berbeda yang dikenal dengan *public key* dan *private key* untuk melakukan enkripsi dan dekripsi (Columbitech, 2001).

Jenis enkripsi ini pertama kali diperkenalkan oleh Whitfield Diffie (1975) yang dikenal sebagai *Public-Key Cryptography* (Stallings, 1995, p164-165). Jadi kita dapat memberikan *private key* kepada orang lain yang ingin mengakses informasi yang kita enkripsi tanpa diketahui kunci enkripsinya.

### 2.1.3 Serangan *Cryptanalyst*

*Cryptanalysis* adalah ilmu pembongkaran suatu *plaintext* tanpa menggunakan *key*. *Cryptanalysis* yang berhasil dapat membongkar *plaintext* atau *key* dari suatu *cryptosystem* dan dapat menemukan kelemahan dalam suatu *cryptosystem* sehingga pada akhirnya memudahkannya membongkar *plaintext* atau *key* tersebut.

Suatu percobaan untuk membongkar suatu *cryptosystem* disebut *attack* (serangan). Terdapat enam jenis serangan *cryptanalysis*, diurutkan berdasarkan kekuatan. Setiap serangan berasumsi bahwa seorang *cryptanalyst* mempunyai pengetahuan lengkap mengenai algoritma enkripsi yang digunakan.

a. *Ciphertext-only attack*

Pada serangan ini *cryptanalyst* mempunyai *ciphertext* dari beberapa pesan, yang kesemuanya telah di-*encrypt* menggunakan algoritma enkripsi yang sama.

b. *Known-plaintext attack*

*Cryptanalyst* tidak hanya mempunyai *ciphertext* dari beberapa pesan, tetapi juga mempunyai *plaintext* dari pesan-pesan tersebut.

c. *Chosen-plaintext attack*

*Cryptanalyst* tidak hanyak mempunyai pasangan *ciphertext* dan *plaintext* dari beberapa pesan tetapi juga memilih *plaintext* yang telah di-*encrypt*.

Serangan jeins ini lebih kuat dari *known-plaintext attack* karena *cryptanalyst* dapat memilih blok *plaintext* tertentu yang mungkin menghasilkan lebih banyak informasi mengenai *key*-nya.

d. *Adaptive-chosen plaintext attack*

Ini adalah kondisi khusus dari *chosen-plaintext attack*. *Cryptanalyst* tidak hanyak dapat memilih *plaintext* untuk di-*encrypt* tetapi juga dapat memodifikasi pilihannya berdasarkan hasil enkripsi sebelumnya. Dalam *chosen-plaintext attack*, *cryptanalyst* mungkin hanya dapat memilih satu blok besar *plaintext* untuk di-*encrypt*, sedangkan pada *adaptive-chosen-plaintext attact cryptanalyst* dapat memilih suatu blok *plaintext* yang lebih kecil dan kemudian memilih blok lainnya berdasarkan hasil enkripsi sebelumnya.

e. *Chosen-ciphertext attack*

*Cryptanalyst* dapat memilih *ciphertext* yang berlainan untuk di-*decrypt* dan mempunyai akses terhadap *plaintext* hasil dekripsi.

f. *Chosen-key attack*

Ini bukanlah suatu attack jika *key*-nya diberikan.

### 2.1.4 *Password* dan Jenis Kunci Enkripsi Lainnya

*Cryptosystem* yang dibuat saat ini bisa menerima kunci enkripsi berupa *password* yaitu sederetan huruf dan atau angka (*pass-phrase*) membentuk satu kata atau lebih (Denning, 1983, p162).

Menurut Stallings (1995, p213), *password* sebagai kunci enkripsi merupakan jenis enkripsi yang paling sering digunakan orang dalam melakukan pengamanan atau pembatasan akses informasi pribadi.

Sekarang dengan bantuan teknologi canggih dan *digital/abstract multimedia interfaces*, maka pola muka, sidik jari, sidik retina, pola suara, atau pola panas tubuh (atau gabungan keseluruhannya) bisa dijadikan kunci enkripsi yang paling baik.

### 2.1.5 Penggunaan Enkripsi

Miller (2000) membuat tiga kelompok yang membutuhkan *cryptography*, yaitu:

a. Pengguna Individual

*Cryptography* sebagai kebutuhan individual terlihat dengan dibuatnya *software* enkripsi yang mampu mengamankan direktori atau *software* khusus untuk pembuatan suatu dokumen pasti memiliki kemampuan untuk mengamankan informasi di dalamnya dengan suatu enkripsi. Di internet terdapat pula jasa *e-mail* yang memiliki fasilitas enkripsi.

b. Perdagangan

Sejak *booming*-nya perdagangan dan penggunaan internet maka peranan *cryptography* dalam dunia perdangan semakin diperlukan. Kebutuhan enkripsi ini dapat berupa kebutuhan pelanggan akan keamanan ketika melakukan transaksi *on-line*.

Misalnya ketika seorang nasabah melakukan *online-banking*, pembelian, penjualan atau transaksi bisnis lainnya, maka data rahasia nasabah berupa nomor rekening, PIN, dan lainnya perlu dijaga dari pihak lain. Dan untuk melakukannya, diperlukan teknik enkripsi yang baik. Hal ini juga mendorong diciptakannya berbagai macam algoritma dan teknik *crpytography*.

c. Pihak militer

Di bidang militer, *cryptography* digunakan untuk telekomunikasi melalui telepon, radio, visual, dan untuk pengamanan jaringan komputer dalam ruang militer.

Sebagai contoh, pengacakan sinyal sistem pelacakan melalui satelit pada *Global Positioning System* (GPS) ke alat telekomunikasi agar arah bicara dan transfer suara melalui alat tersebut tidak dapat disadap oleh pihak lain.

Kebalikan dari itu seorang *crpytanalyst* juga diperlukan untuk mendekripsi informasi dari seluruh musuh atau pihak lain yang ingin disadap balik, jadi ada sifat kebutuhan timbal-balik.

## 2.2 Teknik Enkripsi *Blowfish*

*Blowfish Encryption* adalah enkripsi 64-bit *block cipher* dengan menggunakan *variable-length key*, yang bisa mencapai 448 bit. Blowfish diciptakan oleh Bruce Schneier pada tahun 1993.

### 2.2.1 Latar Belakang

*Blowfish* muncul sebagai standar baru, setelah algoritma enkripsi DES, yang sudah dipakai selama 15 tahun dan menggunakan 56-bit *key*, sudah sangat rentan terhadap penyerangan.

Banyak algoritma enkripsi rahasia seperti *Khufu*, REDOC, atau IDEA. Namun algoritma-algoritma tersebut dilindungi oleh hak paten, sehingga masyarakat tidak dapat menggunakannya dengan bebas.

Dunia tentu saja memerlukan algoritma enkripsi yang aman, tidak dipatenkan, dan dapat digunakan secara gratis. Maka itu, dikembangkanlah algoritma yang dapat memenuhi tuntuan tersebut.

*Blowfish* muncul sebagai algoritma yang dianggap paling memenuhi tuntuan diatas, selain berbagai tuntutan teknis lainnya. Namun algoritma ini juga memiliki keterbatasan. *Blowfish* hanya cocok untuk aplikasi yang *key*-nya tidak terlalu sering berganti, seperti *communications link* atau *automatic file encryptor*. Namun *Blowfish* lebih cepat secara signifikan dibanding DES ketika di-implementasikan pada mikroprosesor 32-bit dengan data *caches* yang besar, seperti Pentium dan PowerPC.

### 2.2.2 Algoritma Blowfish

Algoritma *Blowfish* terdiri dari dua bagian, yaitu bagian *key-expansion*, dan bagian *data-encryption*. *Key-expansion* mengubah key 448 bit menjadi beberapa *array subkey* dengan total 4168 *byte*.

Enkripsi data berlangsung via *Fiestel network* sebanyak 16 *round*. Setiap *round* terdiri dari permutasi *key-dependent*, dan *key-and-data-dependent-substitution*. Semua operasi adalah operasi XOR dan penjumlahan 32 bit. Satu-satunya operasi tambahan adalah pencarian *array* ber-indeks 4 pada setiap *round*.

*Blowfish* menggunakan banyak *subkey*, yang harus dihitung sebelum melakukan enkripsi atau dekripsi. *P array* terdiri dari 32-bit *subkey* yang berjumlah 18 (*P1*, *P2*,..., *P18*) dan ada 4 buah 32-bit *S-box*, masing-masing dengan 256 *entry*, yaitu :

*S1,0, S1,1, ..., S1,255*;

*S2,0, S2,1, ..., S2,255*;

*S3,0, S3,1, ..., S3,255*;

*S4,0, S4,1, ..., S4,255*.

Berikut adalah langkah-langkah dalam membuat subkey:

1. Inisialisasikan terlebih dahulu *P-array*, lalu 4 *S-boxes*, secara berurutan dengan panjang *string* tetap. *String* ini terdiri dari digit heksadesimal dari *Pi*. Contoh:

   *P1* = 0x243f6a88

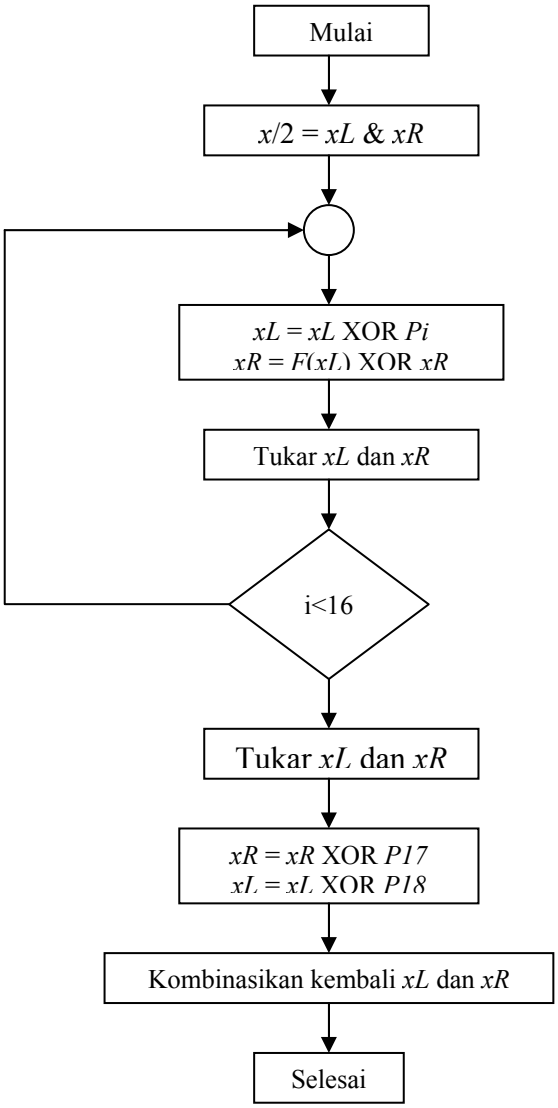
   *P2* = 0x85a308d3

   *P3* = 0x13198a2e

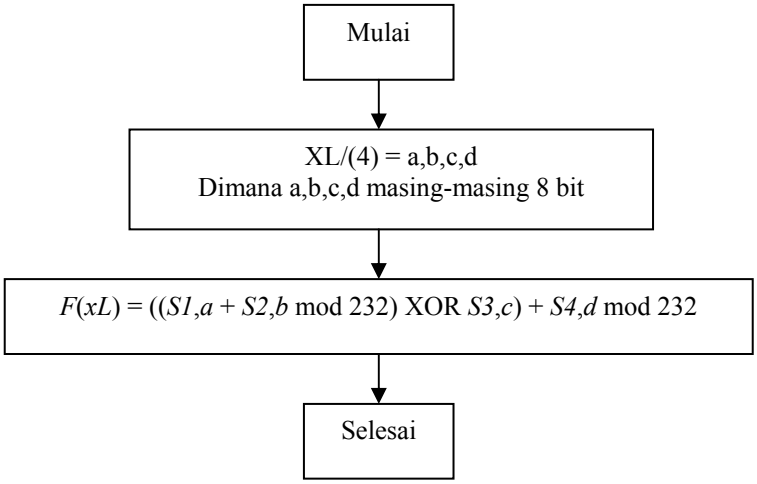
   *P4* = 0x03707344

2. XOR *P1* dengan 32-bit pertama dari *key*, XOR *P2* dengan 32-bit kedua dari *key*, dan selanjutnya untuk semua bit dari *key*. Lakukan perulangan sampai seluruh *P-array* sudah di-XOR dengan *key*.
3. Enkrip semua string nol dengan algoritma *Blowfish*, menggunakan *subkey* yang dipakai pada langkah (1) dan (2).
4. Ganti *P1* dan *P2* dengan output dari langkah (3).
5. Enkrip output dari langkah (3) dengan menggunakan algoritma *Blowfish* dengan menggunakan *subkey* yang sudah dimodifikasi.
6. Ganti *P3* dan *P4* dengan output dari langkah (5).
7. Lanjutkan proses, mengganti semua *entry* dari *P- array*, lalu ke-empat *S-box* secara berurutan, dengan output dari algoritma *Blowfish* yang berganti secara kontinu.

Secara total, dibutuhkan 521 iterasi untuk membuat semua *subkey*. Setelah *subkey* dibuat, enkripsi dapat dilakukan. Algoritma enkripsi Blowfish, dengan *x* sebagai 64-bit data input, adalah sebagai berikut:




**Gambar 2.1 Algoritma Enkripsi Blowfish**




**Gambar 2.2 Fungsi F dalam Algoritma Enkripsi Blowfish**

Input sebanyak 64 bit dibagi menjadi dua bagian, lalu dilakukan operasi XOR dengan ke-16 *subkey* pertama, dan operasi pertukaran antara kedua bagian tersebut dalam sebuah perulangan sebanyak 16 kali. Setelah itu, kembali dilakukan pertukaran, dan dilanjutkan dengan operasi XOR dengan *subkey* ke-17 dan ke-18. Akhirnya, kedua bagian tersebut digabungkan kembali, dan proses enkripsi telah selesai.

Untuk melakukan dekripsi, prosesnya sama persis dengan enkripsi, hanya saja, *P1*, *P2*, ... , *P18* digunakan dalam urutan terbalik. Filosofi dari Blowfish adalah kesederhanaan *design* membuat algoritma lebih mudah untuk dimengerti dan diimplementasi.

Algoritma Blowfish tidak dipatenkan, dan algoritma tersebut dapat diakses oleh publik dan dapat dipergunakan dengan bebas. Sehingga, orang-orang yang membutuhkan algoritma enkripsi yang aman dapat mempergunakannya tanpa memerlukan ijin atau hak cipta apapun.

## 2.3 Sejarah Pengenalan Suara

Penelitian di bidang pengenalan suara (*speech recognition*) oleh mesin telah dilakukan selama hampir lima dekade. Percobaan yang paling awal untuk mengembangkan *system speech recognition* oleh mesin dibuat pada tahun 1950-an yakni pada saat berbagai peneliti mencoba untuk mengeksploitasi ide fundamental dari *acoustic-phonetics*.

Tahun 1952, di laboratorium Bell, Davis, Biddulph dan Balashek membuat suatu sistem pengenalan digit terisolasi untuk seorang pembicara. Sistem tersebut sangat bergantung pada pengukuran resonansi spektral di daerah vokal dari setiap digit.

Pada tahun 1956 sebuah usaha independen pada laboratorium RCA, Olson dan Belar berusaha untuk mengenali 10 suku kata berbeda dari seorang pembicara yang juga bergantung pada pengukuran spectral pada area vokal.

Tahun 1959, Universitas College di Inggris, Fry dan Denes mencoba untuk membuat sebuah pengenalan fonem untuk mengenali 4 vokal dan 9 konsonan. Mereka menggunakan sebuah analisis spektrum dan pencocok pola untuk menghasilkan keputusan dari pengenalan.

Usaha lain pada periode ini adalah pengenalan vokal dari Forgie dan Forgie, dikonstruksikan di laboratorium Lincoln MIT pada tahun 1959, dimana 10 vokal disisipkan dalam format a/b/-vocal-/t dapat dikenali. Sekali lagi sebuah *filter bank analyzer* digunakan untuk menyediakan informasi spektral dan pengukuran waktu yang dipakai untuk memperkirakan resonansi jejak vokal yang menentukan vokal mana yang diucapkan.

Pada tahun 1960-an jumlah ide fundamental dalam *speech recognition* muncul ke permukaan dan diterbitkan. Dekade ini dimulai dengan beberapa laboratorium Jepang yang memasuki arena pengenalan dan membangun *special-purpose hardware* sebagai bagian dari sistem mereka. Awal dari system Jepang dimulai oleh Suzuki dan Nakata dari laboratorium penelitian radio di Tokyo yang merupakan sebuah *hardware* pengenalan huruf vokal.

Usaha lainnya dilakukan oleh Sakae dan Doshita dari Universitas Kyoto tahun 1962 yang membangun sebuah hardware pengenalan fonem. Usaha yang ketiga merupakan *hardware* pengenalan digit oleh Nagata dan rekan kerjanya di laboratorium NEC pada tahun 1963. Usaha ini mungkin merupakan yang paling dikenali sebagai percobaan perdana dari *speech recognition* pada NEC dan menjadi awal bagi sebuah program penelitian yang produktif.

Di era 60 an ini juga terdapat proyek penelitian kunci. Yang pertama dari ketiga proyek itu adalah usaha yang dilakukan Martin dan rekannya pada laboratorium RCA untuk membangun solusi realistik bagi problem yang berkaitan dengan ketidak seragaman dalam interval waktu dari *speech event*. Martin membangun sistem dengan dasar pada kemampuan untuk mendeteksi awal dan akhir dari suatu *speech*. Martin membangun metodenya dan mendirikan suatu perusahaan Treshold Technology yang membuat, memasarkan, dan menjual produk-produk *speech recognition*.

Sementara pada waktu yang bersamaan, di Uni Sovyet, Vintsyuk mengajukan penggunaan dari metode *dynamic programming* untuk penyamaan waktu dari sepasang pengutaraan *speech*.

Pencapain yang terakhir di tahun 60-an merupakan penelitian *pioneer* dari Reddy dalam bidang *continous speech recognition* dengan *dynamic tracking* dari fonem.

Pada tahun 1970-an penelitian *speech recognition* meraih sejumlah batu pijakan yang signifikan. Pertama, area dari kata terisolasi atau pengenalan ucapan diskrit menjadi suatu teknologi yang mungkin dan berguna berdasar pada pembelajaran fundamental oleh Velicho dan Zagoruyko di Rusia, Sakoe dan Ciba di Jepang, dan Itakura di Amerika. Pembelajaran Rusia membantu memajukan penggunaan dari ide *pattern recognition* dalam *speech recognition*. Penelitian Jepang menunjukkan bagaimana metode *dynamic programming* dapat diterapkan dengan sukses dan penelitian Itakura menunjukan bagaimana ide dari *Linear Predictive Coding* (LPC) yang mana telah digunakan dengan suskses pada pencodean *speech* ber-bit rendah.

Penelitian *speech* dalam tahun 1980-an bercirikan pada pergeseran teknologi dari pendekatan berbasis *template* menjadi metode *statistic modeling* terutama pendekatan model *hidden* Markov. Ide lain yang diperkenalkan pada akhir tahun 1980-an adalah penerapan *neural network* pada *speech recognition*. *Neural network* pertama kali dikenalkan pada tahun 1950 tapi tidak terbukti berguna pada awalnya karena terlalu banyaknya masalah praktikal.

## 2.4 Sinyal Analog

Sebuah sinyal analog merupakan sinyal variabel kontinu. Sinyal analog biasanya dimasukkan ke dalam konteks listrik. Bagaimanapun, mekanik, *pneumatic*, hidrolik dan sistem yang lain juga menggunakan sinyal analog.

Kata analog menyatakan sebuah analogi antara sebab dan akibat yakni tegangan masuk dan tegangan keluar, arus listrik masuk dan arus listrik keluar, suara yang masuk dan frekuensi yang keluar.

Sebuah sinyal analog menggunakan beberapa properti dari sebuah medium untuk menyampaikan informasi sinyal. Sebagai contoh sebuah *aneroid* barometer menggunakan putaran posisi sebagai sinyal untuk menyampaikan informasi. Secara elektris, properti yang biasanya digunakan adalah tegangan, diikuti oleh frekuensi, arus listrik, dan beban listrik.

Berbagai informasi dapat diteruskan oleh sebuah sinyal analog. Sering terjadi sebuah sinyal analog merupakan sebuah pengukuran respon untuk perubahan fenomena, seperti suara, cahaya, temperatur, posisi, atau tekanan.

Sebagai contoh, dalam sebuah perekaman suara analog, variasi dalam tekanan suara yang masuk melalui mikrofon menciptakan sebuah variasi korespondensi dalam penyampaiannya. Peningkatan volume suara mengakibatkan terjadinya fluktuasi peningkatan dari arus listrik selama menjaga ritem agar tetap sama.

Kerugian utama dari sebuah sinyal analog adalah bahwa banyak sistem memiliki gangguan (*noise*). Ketika sinyal di-*copy* dan di-*copy* ulang atau ditransmisikan untuk jarak jauh, variasi acak menjadi dominan. Secara elektrik kehilangan ini dikurangi dengan memberi pelindung, koneksi yang baik, dan beberapa tipe kabel seperti kabel *coaxial* dan *twisted pair*.

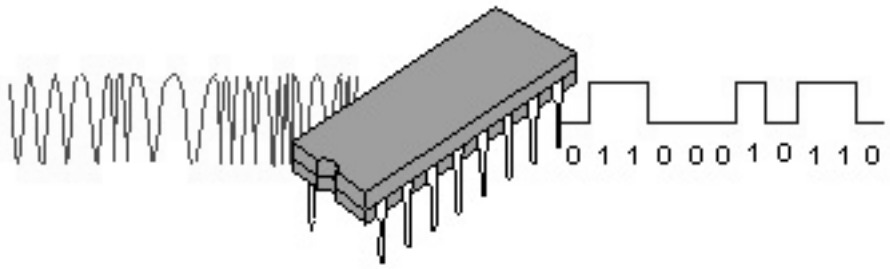
Efek dari gangguan (*noise*) membuat sinyal menjadi hilang dan distorsi menjadi tidak mungkin untuk dipulihkan.

## 2.5 Sinyal Digital

Sinyal digital merupakan sinyal elektronik yang ditansmisikan sebagai kode *binary* yang dapat menggambarkan ada atau tidaknya arus listrik, tinggi atau rendahnya tegangan, atau *pulse* yang pendek pada frekuensi tertentu.

Sinyal digital adalah sebuah sinyal yang telah dikuantisasikan dari sebuah sinyal kontinu. Dengan mengkuantisasikan sinyal, nilai dari sinyal diskrit tidak lagi kontinu, tetapi menjadi diskrit.

Dalam kebanyakan aplikasi, pengkuatisasian sinyal digital diukur dalam bits. Sebagai contoh, *compact disc audio* menggunakan 16 *bit stereo audio sample* pada 44.1 *kHz.* Yang berarti, setiap detik, CD audio membutuhkan 16*2*44,100 = 1,411,200 *bits*.




**Gambar 2.3 Sinyal Analog Diubah Menjadi Sinyal Digital**

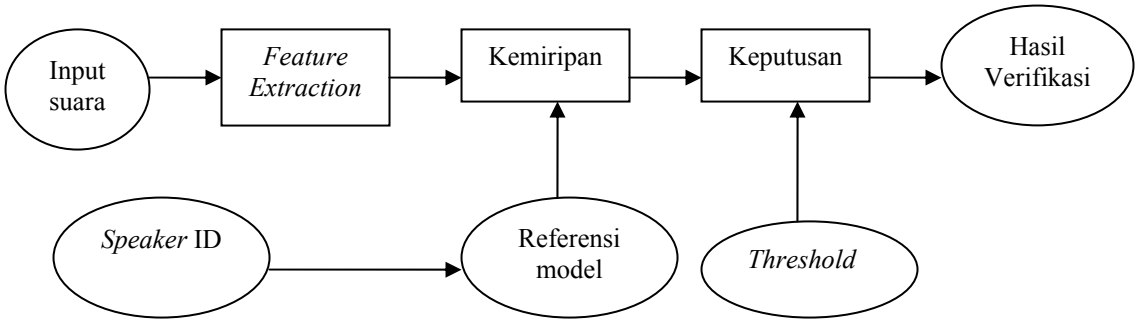
## 2.6 *Speaker Recognition* (Pengenalan Pembicara)

*Speaker recognition* merupakan sebuah proses pengenalan secara otomatis siapa yang berbicara berdasarkan pada informasi yang ada di dalam gelombang suara. Teknik ini memungkinkan untuk menggunakan suara pembicara untuk menggambarkan identitas mereka, dan mengontrol akses pelayanan seperti *voice dialing*, pelayanan

perbankan melalui telepon, belanja lewat telepon, akses *database*, informasi pelayanan, *voice mail*, pengendalian keamanan area informasi rahasia, dan untuk akses komputer jarak jauh.

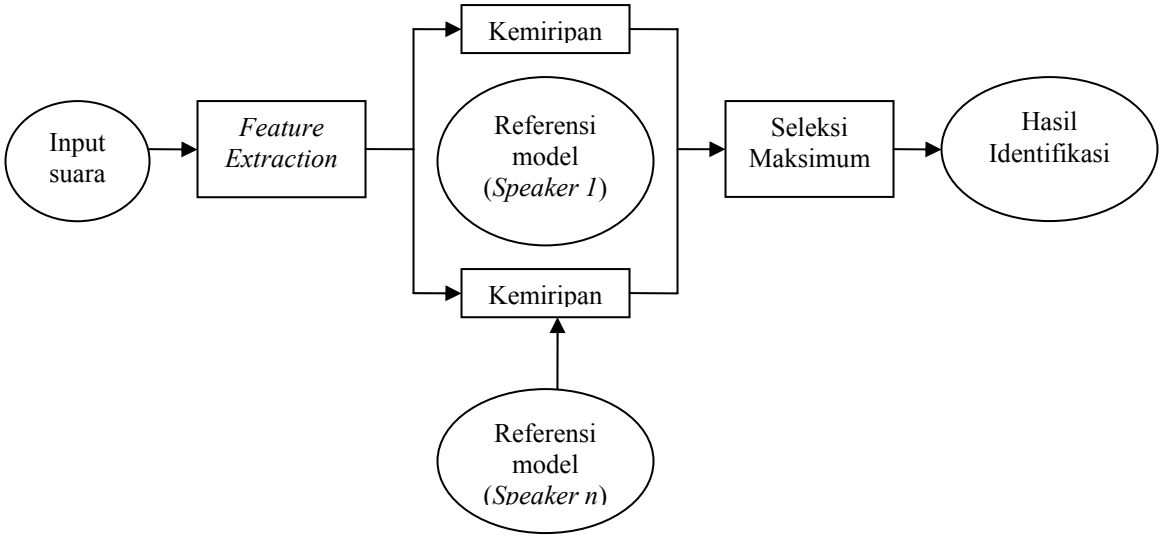
### 2.6.1 Prinsip *Speaker Recognition*

Pengenalan pembicara dapat diklasifikasikan menjadi dua macam, yaitu identifikasi dan verifikasi. Identifikasi pembicara adalah proses untuk menentukan siapa pembicara tersebut, berdasar data yang telah didapat sebelumnya. Verifikasi pembicara adalah suatu proses menerima atau menolak identitas yang diklaim sebagai pembicara. Struktur dari identifikasi pembicara dan sistem verifikasi digambarkan seperti dibawah ini.




**Gambar 2.4 *Speaker Verification***

Setiap sistem pengenalan pembicara (*speaker recognition*) harus melewati dua tahap penting. Tahap yang pertama adalah fase pelatihan, dan fase yang kedua adalah fase *testing*. Dalam fase pelatihan, setiap pembicara yang telah tercatat harus menyediakan sampel dari suara mereka sehingga sistem dapat menciptakan model referensi untuk pembicara tertentu. Dalam kasus sistem verifikasi pembicara, *threshold*




**Gambar 2.5*Speaker Identification***

pembicara yang spesifik juga dihitung dari sampel pelatihan. Dalam fase *testing*, input suara dicocokan dengan model referensi yang telah disimpan sebelumnya dan selanjutnya dibuat keputusan pengenalan.

*Speaker recognition* adalah suatu sistem yang sulit dan masih menjadi hal yang aktif dikembangkan. *Speaker recognition* bekerja berdasarkan keyakinan bahwa karakteristik seseorang adalah unik untuk setiap pembicara. Bagaimanapun, penelitian ini telah ditantang dengan banyaknya variasi *in-out* dari sinyal suara. Dasar sumber dari variasi suara berasal dari pembicara sendiri. Sinyal suara dalam fase pelatihan dan testing menjadi berbeda sama sekali berdasar banyak fakta bahwa suara manusia berubah dengan berjalannya waktu, kondisi kesehatan, dan lain sebagainya. Ada faktor lain diluar faktor pembicara sendiri, yaitu suara dan variasi akustik dalam perekaman suara.

### 2.6.2 Ekstraksi Ciri (*Feature Extraction*) Suara

Tujuan dari ekstraksi ciri suara ini adalah untuk mengubah gelombang suara menjadi beberapa tipe representasi parametrik yang dapat diproses. Hal ini sering dikaitkan dengan pengolahan sinyal.

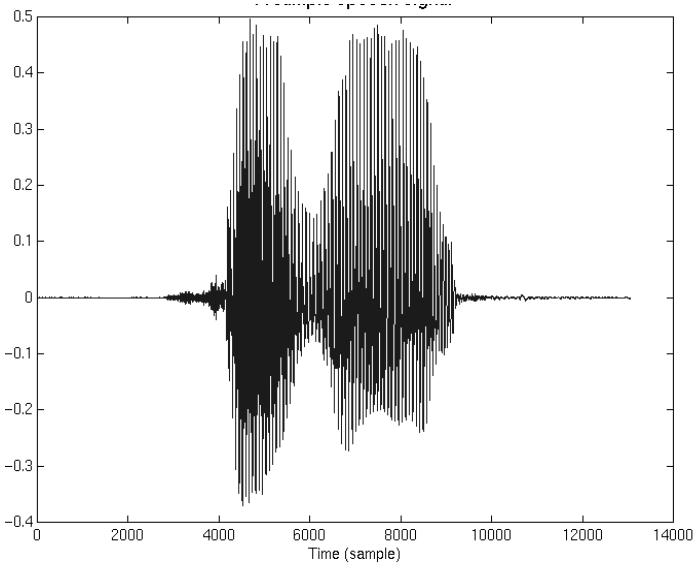
Sinyal suara adalah sebuah sinyal beragam yang diperlambat. Ketika di-tes dalam sebuah periode waktu yang singkat (antara 5 dan 100 *msec*), karakteristiknya menjadi tetap dan tanpa bias. Bagaimanapun, dalam jangka waktu yang lama (antara 0,2 detik atau lebih) karakteristik dari sinyal berubah untuk merefleksikan suara yang diucapkan. Untuk itu analisis spektral jangka waktu singkat adalah merupakan cara yang paling umum untuk mengkarakterisasi sinyal suara.

Ada banyak cara untuk merepresentasikan suara secara parametris sehingga dapat diproses lebih lanjut, antara lain seperti *Linier Predictive Coding*, *Mel Frequency Cepstrum Coefissients* (MFCC) dan masih ada banyak lagi. MFCC mungkin merupakan yang paling dikenal dan paling populer, dan MFCC juga merupakan metode yang akan digunakan dalam aplikasi ini.

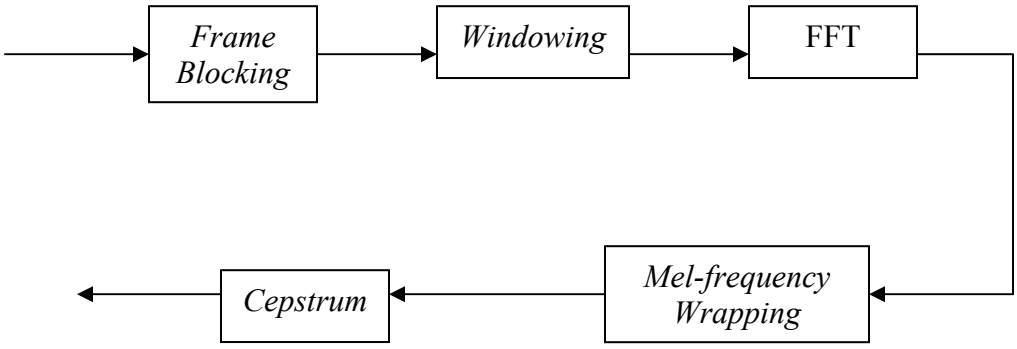
MFCC didasarkan pada variasi *bandwith* frekuensi dari telinga manusia, atau bagaimana manusia mendengar. Linear pada frekuensi rendah, dan logaritmik pada frekuensi tinggi. Teknik ini baik digunakan untuk menangkap karakteristik dari suara manusia. Hal ini diekspresikan dalam skala *mel-frequency*, yang mana adalah linear untuk frekuensi di bawah 1000 Hz, dan logaritmik untuk frekuensi diatas 1000 Hz.

### 2.6.3 Prosesor *Mel-frequency Cepstrum Coefficients* (MFCC)

Input suara biasanya direkam pada *sampling rate* diatas 8000 Hz. Frekuensi *sampling* ini dipilih untuk meminimalisasi efek dari *aliasing* dalam konversi analog ke digital. Sinyal yang telah di-*sampling* ini dapat menangkap semua frekuensi sampai dengan 5 kHz, yang meliputi kebanyakan energi suara yang dihasilkan oleh manusia. Seperti telah disebutkan pada bagian sebelumnya, tujuan utama dari prosesor MFCC adalah untuk menirukan perilaku telinga manusia.




**Gambar 2.6 Input Suara**

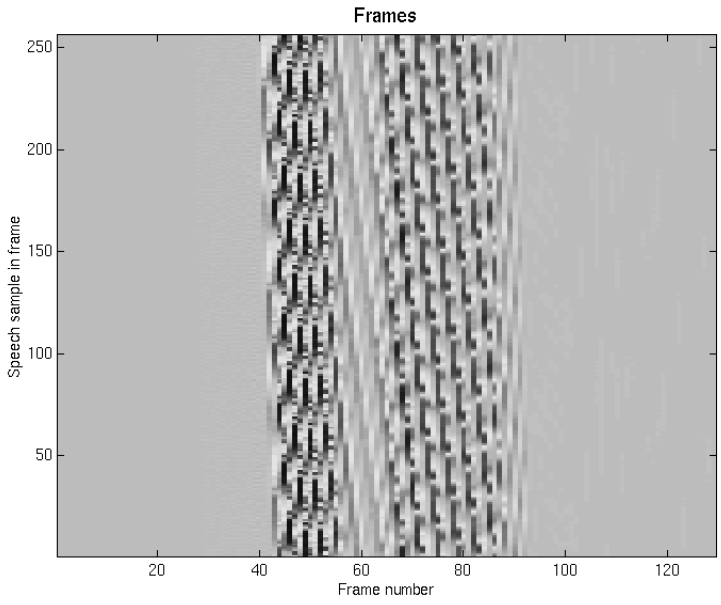



**Gambar 2.7 Prosesor *Mel-frequency Cepstrum Coefficients***

Seperti terlihat pada gambar, prosesor MFCC mempunyai lima tahap, yaitu *Frame Blocking, Windowing, FFT, Mel-frequency Wrapping*, dan *Cepstrum*. Berikut adalah penjelasan dari masing-masing bagian tersebut.

a. *Frame Blocking*

Dalam tahap ini sinyal suara diblok menjadi banyak *frame* dengan *N* sampel, dengan *frame-frame* selanjutnya dipisahkan oleh *M* (*M*<*N*). *Frame* yang pertama terdiri dari N sampel yang pertama. *Frame* yang kedua mulai *M* sampel setelah *frame* yang pertama dan di-*overlap* oleh *N*-*M* sampel. *Frame* yang ketiga mulai dari 2*M* sampel setelah *frame* yang pertama atau *M* sampel setelah *frame* yang kedua dan di-*overlap* oleh *N*-2*M* sampel. Proses ini berlangsung kontinu hingga seluruh suara atau ucapan dihitung sebagai satu atau lebih *frame*. Nilai untuk *N* dan *M* yang umum adalah *N*=256 (yang ekuivalen dengan 30 msec *windowing* dan memfasilitasi radix-2 FFT) dan *M*=100.




**Gambar 2.8 Input Suara Setelah Melalui Tahap *Frame Blocking***

b. *Windowing*

Tahap berikutnya adalah untuk membingkai setiap *frame* individu untuk meminimalisasikan sinyal yang diskontinu pada setiap awal dan akhir setiap *frame*. Konsepnya adalah untuk meminimalisasikan distorsi spektral dengan menggunakan bingkai untuk membuat sinyal menjadi nol pada awal dan akhir setiap frame. Jika kita mendefinisikan bingkai sebagai *w(n)*, $0 \le n \le N$-1 dimana *N* adalah jumlah sampel di setiap *frame*, maka hasil dari proses *windowing* adalah sinyal

$$y_l(n) = x_l(n)w(n),\ 0 \le n \le N\text{-}1$$

Biasanya teknik *windowing* yang digunakan adalah *Hamming window*, yaitu:

$$w_n = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1$$

c. *Fast Fourier Transform* (FFT)

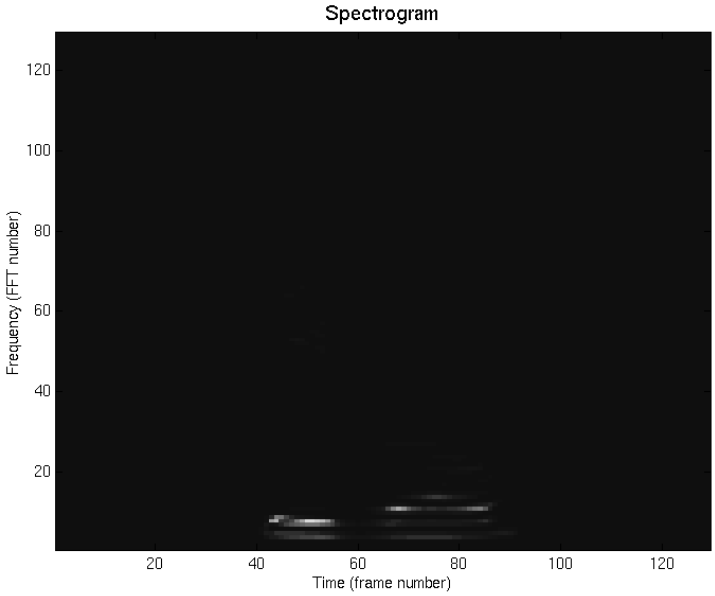
Proses berikutnya adalah *Fast Fourier Transform* yang mengkonversi setiap *frame N* sampel dari *domain* waktu menjadi *domain* frekuensi. FFT adalah algoritma cepat untuk mengimplementasikan *Discrete Fourier Transform* (DFT) yang didefinisikan dalam suatu set *N* sampel $\{x_n\}$ seperti berikut :

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N}, \qquad n = 0,1,2,...,N-1$$

Sebagai catatan bahwa kita menggunakan *j* disini untuk menyatakan unit imajiner. Secara umum *Xn* merupakan angka kompleks. Hasil dari *Xn* diinterpretasikan sebagai berikut : frekuensi 0 berkorespondensi ke *n*=0, frekuensi positif 2/0, *F f* < <

berkorespondesi ke 1 1 2 / - ≤≤ + *N n N*. Dalam hal ini *Fs* menyatakan frekuensi *sampling*.

Hasil yang didapatkan setelah melakukan proses ini biasanya direfrensikan sebagai sinyal spektrum atau *periodogram*.




**Gambar 2.9 Input suara Setelah Melalui FFT**

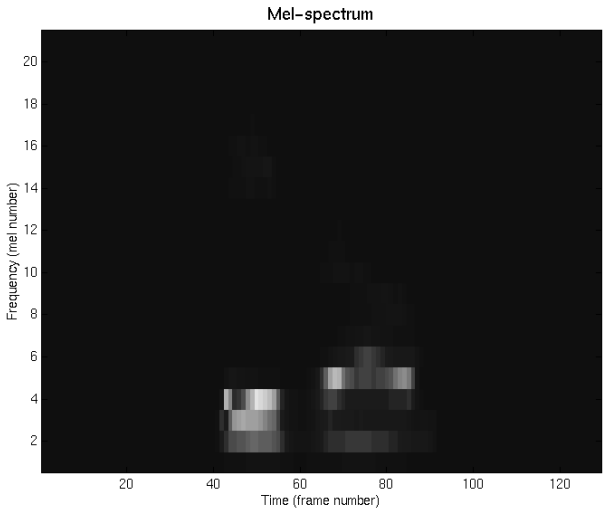
d. *Mel-frequency Wrapping*

Seperti telah dijelaskan sebelumnya , studi tentang *psychophysical* telah menunjukkan bahwa persepsi manusia tentang frekuensi berisi suara untuk sinyal suara tidak mengikuti sebuah skala linier. Jadi, untuk setiap nada dengan frekuensi *f*, diukur dalam satuan Hz, sebuah *subjective pitch* diukur pada skala yang dinamakan skala 'mel'. Skala *mel-frequency* terdiri dari frekuensi linear dibawah 1000 Hz dan frekuensi logaritmik diatas 1000 Hz. Sebagai referensi, *pitch* dari nada 1 kHz, 40 dB diatas nilai

*threshold* pendengaran, didefinisikan sebagai 1000 mels. Maka dari itu, kita dapat menggunakan formula yang tepat untuk menghitung mels untuk frekuensi yang diberikan dalam Hz:

$$mel(f) = 2595*\log_{10}(1 + f / 700)$$

Satu pendekatan untuk mensimulasi spektrum adalah dengan menggunakan *filter bank*, satu *filter* untuk setiap *mel-frequency* komponen. *Filter bank* memiliki respon *triangular bandpass*, dan jarak atau *bandwith* ditentukan oleh interval *mel-frequency* yang konstan. Spektrum yang telah dimodifikasi dari $S(\omega)$ jadi terdiri dari *output power* dari *filter* tersebut ketika $S(\omega)$ adalah inputnya. Angka dari koefisien mel spektrum, $K$, biasanya bernilai 20.




**Gambar 2.10 Input Suara Setelah Melalui Tahap Mel-frequency Wrapping**

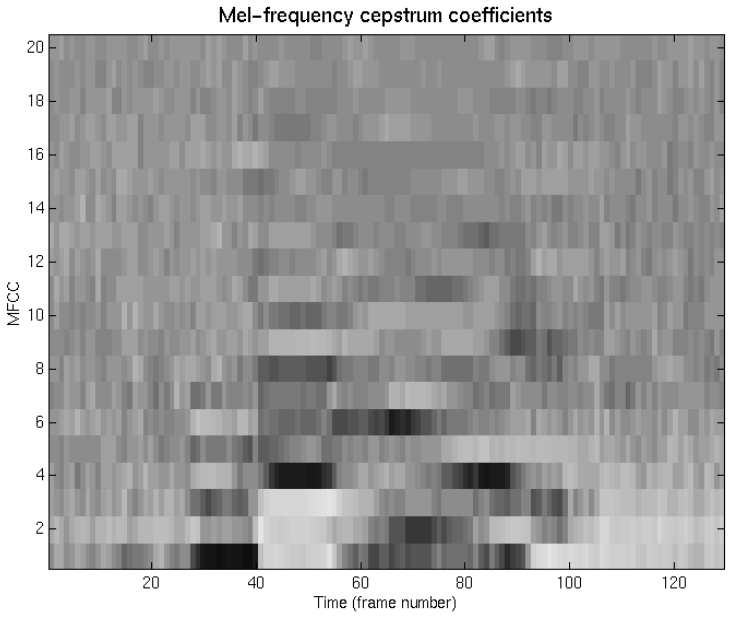
e. *Cepstrum*

Dalam tahap terkahir, kita mengkonversi *log mel spectrum* kembali ke waktu. Hasilnya disebut *Mel-frequency Cepstrum Coefficients* (MFCC). Representasi cepstral

dari spektrum suara menyediakan representasi yang baik bagi properti cepstral lokal dari sinyal untuk frame analisis yang diberikan. Karena mel spektrum koefisien adalah bilangan real, kita dapat mengkonversinya ke domain waktu menggunakan *Discrete Cosine Transform* (DCT). Untuk itu, jika kita menunjukkan *mel power spectrum coefficients* yang merupakan hasil dari tahap terkahir $\widetilde{S}_k$, $k$ =1,2,...,$K$ , kita dapat mengkalkulasi $\widetilde{C}_n$ sebagai

$$\widetilde{C}_n = \sum_{k=1}^{K} (\log \widetilde{S}_k) \cos\left[ n\left( k - \frac{1}{2} \right) \frac{\pi}{K} \right], \quad n = 1,2,...,K$$

Komponen $\widetilde{C}_0$ tidak termasuk dalam DCT karena $\widetilde{C}_0$ merepresentasikan nilai rata-rata dari sinyal, yang hanya membawa sedikit informasi spesifik tentang si pembicara.




**Gambar 2.11 Input Suara Setelah Melalui Tahap Cepstrum**

Setelah sinyal suara melalui proses-proses dalam prosesor MFCC, sinyal tersebut telah berubah menjadi vektor suara yang ringkas dan hanya memuat informasi penting mengenai suara si pembicara. *Sampling rate* yang optimal untuk digunakan adalah 8000, sehingga komputasi tidak terlalu berat, tetapi masih menyimpan data penting suara.

### 2.6.4 *Vector Quantization*

Setelah suara diubah menjadi bentuk yang efisien dan hanya memuat informasi yang penting dari suara yang diucapkan, masalah yang muncul adalah: bagaimana melakukan pengenalan suara dari pembicara itu ketika yang bersangkutan melakukan dekripsi?

Ada banyak metode yang sudah dikembangkan untuk menjawab masalah ini, antara lain *Dynamic Time Warping* (DTW), *Hidden Markov Modeling* (HMM), *Neural Network*, dan *Vector Quantization*.

Pada awalnya, *Neural Network* dipakai untuk melakukan pengenalan suara. Namun, setelah diuji-coba, ternyata *Vector Quantization* menunjukkan hasil yang lebih akurat. Jadi, metode yang digunakan adalah *Vector Quantization*, dengan beberapa modifikasi.

*Vector Quantization* adalah proses untuk memetakan vektor dari ruang vektor yang sangat luas, menjadi *region-region* dengan jumlah terbatas di dalam ruang itu. Setiap *region* disebut *cluster*, dan dapat direpresentasikan dengan intinya, yang disebut *codeword*. Kumpulan dari *codeword* adalah *codebook*.

Jarak vektor ke *codeword* yang terdekat dalam suatu *codebook* disebut *VQ-distortion*. Dalam fase pengenalan, suara seorang pembicara yang tak dikenal di-"*vector*

*quantize*" menggunakan *codebook*, lalu total *VQ-distortion* dihitung. *VQ-distortion* dengan *codeword* yang terkecil diidentifikasi sebagai pembicaranya.

Namun, dalam aplikasi ini, permasalahannya sedikit berbeda. Algoritma *Vector Quantization* yang sebenarnya, digunakan untuk mengidentifikasi suara pembicara tidak dikenal dengan beberapa sample suara dari pembicara yang berbeda, yang sudah disimpan sebelumnya (dalam *codebook*). Namun dalam aplikasi ini, suara pembicara yang disimpan hanya satu. Yaitu si pelaku enkripsi.

Dalam *Vector Quantization*, *VQ-distortion* di-inisialisasikan sebagai *infinite*. Kemudian, jika ditemukan *VQ-distortion* yang lebih kecil dari sebelumnya dalam suatu pengulangan, *VQ-distortion* itu akan disimpan. Jadi, *VQ-distortion* yang terkecil yang akan muncul sebagai pembicara yang di-identifikasi.

Sedangkan pada aplikasi ini, algoritma tersebut diubah sedikit menjadi: Hitung nilai *VQ-distortion*. Jika *VQ-distortion*-nya cukup kecil, maka kembalikan nilai *true*.

Dalam mengaplikasikan metode *Vector Quantization* pada aplikasi pengenalan pembicara, jumlah *centroid* yang paling optimal adalah 16 buah untuk memudahkan dalam pencarian *Euclidean Distance*.

### 2.6.5 *Pairwise Euclidean Distance*

Perhitungan *VQ-distortion* dilakukan dengan menghitung *Pairwise Euclidean distance* antara dua buah vektor, dengan formula:

$$D = \text{sum}((x\text{-}y)^2)^{0.5}$$

dengan *D* untuk *distance* / *VQ-distortion*, dan *x* dan *y* adalah pasangan vektor suara pada saat enkripsi dan dekripsi.

Jika *VQ-distortion* sudah dihitung, langkah selanjutnya adalah menetapkan batas toleransi dalam melakukan pengenalan. Misalkan ditetapkan batasnya adalah $2 < D < 3$. Maka, bila $D$ bernilai 1.5 atau 3.7, *user* akan ditolak. Namun bila nilai $D$ masuk ke dalam *range*, misalnya 2.3, maka user akan diidentifikasi sebagai orang yang berhak membuka file.

### 2.6.6 Faktor yang Menentukan Akurasi Verfifikasi Pembicara

Ada beberapa faktor yang dapat mempengaruhi keberhasilan prosesor MFCC dan *Vector Quantization* dalam melakukan verifikasi pembicara yakni :

a) Frekuensi

Frekuensi merupakan pengukuran terhadap berapa banyak atau berapa kali suatu kejadian berlaku per satuan waktu. Untuk menghitung frekuensi suatu *event*, berapa banyak *event* terjadi dalam interval waktu tertentu dihitung dan dibagi dengan panjang waktu interval tersebut.

b) Intonasi

Intonasi merupakan suatu variasi nada yang digunakan ketika berbicara. Intonasi pembicara sangat menentukan tingkat keberhasilan dari proses verifikasi pembicara karena intonasi berbeda ketika mengucapkan suata kata dapat diartikan lain oleh komputer.

c) Tingkat kebisingan

Dalam pengertian umum, *noise* merupakan suara yang tidak diinginkan. Ketika berbicara *nois*e dalam relasinya dengan suara, *noise* merupakan suara yang lebih besar dari volume biasa. Pembicaraan orang lain dapat disebut *noise* bagi seseorang yang tidak terlibat dalam pembicaraan

tersebut, dan *noise* dapat berupa suara yang tidak diinginkan seperti suara kapal terbang, kebisingan jalan raya, dan sebagainya. *Noise* sangat berpengaruh sekali terhadap hasil verifikasi pembicara karena dengan banyaknya *noise* komputer tidak dapat membedakan suara asli dengan *noise* yang ada.

d) Volume suara

Volume berarti sebuah kuantifikasi dari seberapa banyak suatu objek menempati ruang kosong. Dalam hal ini volume berarti besar kecilnya suara yang dihasilkan.