Studi Analasis Pengiriman Suara Menggunakan Algoritma Serpent

Anastasia Tiwa [(1)], Arie S.M. Lumenta, ST,MT.[(2)], Arthur M. Rumagit, ST, MT. [(3)], Aneke P.R. Wowor, ST.[(4)]
(1)Mahasiswa (2)Pembimbing 1 (3)Pembimbing 2 (4)Pembimbing 3

anastasiatiwa@gmail.com[(1)] arie.lumenta@gmail.com [(2)]arthurrumagit@rocketmail.com[(3)] anneke.wowor@yahoo.com [(4)]

Jurusan Teknik Elektro-FT, UNSRAT, Manado-95115

Abstract - Along with current technological developments in voice communication has not been followed in the presence of a security standard. From the data there is useful information that not all parties are entitled to know what information is confidential. This paper, giving out a solution for voice messaging security by using cryptography is to encrypt voice messages by using Algorithm Serpent. Algorithm is one of the Serpent encryption algorithm chiperblock. This algorithm is a runner-up in the competition of the Advanced Encryption Standard (AES). Number of repeated rounds used in this algorithms is the most amongst other AES candidate algorithms, namely 32. Process is performed 32 times consist of key mixing operation, the value of the S-boxes and linear transformations. Implementation of this research in the form of encrypted voice transmission applications using the Serpent algorithm is implemented in Java using NetBeans IDE (Intergrated Development Environment) and runs on the Windows operating system. This application is used to encrypt voice messages in real-time voice communication. The application comes with the chat feature that can be used by user. In the process of sound transmission delay time (delay) produced large

Keywords: Cipher block, Encryption, Java, Serpent

Abstrak - Seiring dengan perkembangan teknologi saat ini dalam komunikasi suara belum diikuti dengan adanya suatu standart keamanan. Dari data ada informasi yang berguna yang tidak semua pihak berhak untuk mengetahui informasi yang bersifat rahasia. Makalah ini, memberikan satu solusi untuk keamanan pengiriman pesan suara dengan menggunakan kriptografi yaitu dengan mengenkripsi pesan suara tersebut dengan menggunakan Algoritma Serpent. Algoritma Serpent merupakan salah satu algoritma enkripsi chiperblock. Algoritma ini merupakan runner-up dalam kompetisi Advanced Encryption Standard (AES). Jumlah putaran berulang yang digunakan dalam algoritma ini paling banyak dari antara algoritma-algoritma kandidat AES yang lain, yaitu 32. Proses dilakukan 32 kali tersebut terdiri atas operasi pencampuran kunci, proses nilai S-boxes, dan transformasi linear. Implementasi dalam penelitian ini berupa aplikasi pengiriman suara terenkripsi menggunakan Algoritma Serpent yang diimplementasikan pada Java dengan menggunakan IDE (*Integrated Development Environment*) NetBeans dan berjalan pada sistem operasi Windows. Aplikasi ini digunakan untuk mengenkripsikan pesan suara dalam komunikasi suara real-time. Aplikasi ini dilengkapi dengan fitur *chatting* yang dapat digunakan oleh pengguna. Pada proses pengiriman suara waktu tunda (*delay*) yang dihasilkan besar.

Kata kunci: Cipher Blok, Enkripsi, Java, Serpent

I. PENDAHULUAN

Perkembangan teknologi dalam komunikasi data ini menawarkan berbagai jenis kemuduhan bagi kehidupan sehari-hari berbagai lapisan masyarakat. Salah satu komunikasi suara yang dikenal saat ini adalah VoIP (*Voice Over Internet Protocol)*, yaitu komunikasi suara yang dapat dilakukan melaluo jaringan untuk pertukaran informasi.

Namun berbagai peluang dari pencurian dan pengubahan data dari pihak yang tidak berkempentingan yang merugikan merupakan suatu isu yang sangat penting , oleh karena itu untuk melindungi data terhadap akses, pengubahan dan penghalangan yang akan dilakukan dari pihak yang tidak berkempentingan, peranti keamanan data yang melintas di jaringan komputer harus disediakan.

Pada komunikasi *VoIP* terdapat 2 jenis solusi pengamanan yang dapat dilakukan. Solusi pertama adalah *Voice Scrambling* dan solusi lainnya Enkripsi. *Voice Scrambling* yaitu perubahan pada sinyal telekomunikasi untuk membuatnya menjadi tidak dapat diketahui oleh siapapun kecuali pihak yang memiliki alat penerima khusus, Enkripsi dilakukan pada data suara tersebut sebelum data suara dikirimkan, sehingga pihak lain yang tidak berhak tidak dapat memahami data suara yang dikirimkan tersebut meskipun data suara berhasil diakses. Biasanya enkripsi dilakukan oleh suatu alat atau aplikasi pengenkripsi.

Pada suatu sistem enkripsi algoritma yang digunakan untuk enkripsi suara ada bermacam-macam , masing-masing memiliki karakteristik sendiri. Karena masih belum ada satu algoritma tertentu yang menjadi standart enkripsi komunikasi suara, maka perlu ada usaha untuk menerapkan algoritma enkripsi lain untuk mengetahui sebaik apa algoritma tersebut mengenkripsi suara.

Pada tugas akhir ini penulis menggunakan Algoritma Serpent untuk diterapkan pada aplikasi pengiriman suara ini. Algoritma serpent merupakan Algoritma kuat saat ini yang sampai saat ini dinyatakan aman karena masih belum ada serangan kriptanalisis yang benar-benar dapat mematahakan algoritma ini.

Serpent adalah sebuah algoritma cipher blok yang merupakan hambatan jika diterapkan pada enkripsi suara. Pada bentuk algoritma ini jika di implemetasikan pada pengiriman suara akan menimbulkan *delay* yang cukup besar karena algoritma ini beroperasi dalam bentuk blok bit, yang panjangnya sudah ditentukan

sebelumnya. Untuk memperkecil *delay*-nya, harus dulakukan peyusuaian pada algoritma ini yaitu dengan menyesuaikan mode operasinya.

Algoritma menggunakan operasi *Cipher block Channing* (CBC) yang melakukan enkripsi secara sekuensial yang membuat kecepatan enkripsi tidak meningkat, oleh karena itu digunakan mode operasi menyerupai chiper aliran yaitu mode operasi *counter* yang dapat digunakan untuk mengubah kecepatan enkripsi.

II. LANDASAN TEORI

A. Keamanan Jaringan

Layanan-layanan keamanan jaringan didefinisikan berdasarkan kebutuhan yang harus disediakan untuk memenuhi permintaan terhadap keamanan jaringan. Layanan keamanan jaringan bedasarkan rekomendasi ITU-T yaitu Otentikasi, kendali akses, kerahasian data, keutuhan data, *Non-Repudution* dan ketersediaan. Untuk mewujudkan layanan keamanan jaringan pengembang sistem dapat menggunakan mekanisme keamanan jaringan. Mekanisme keamanan antara lain *Encipherment* yang merupakan keamanan yang digunakan untuk menyembunyikan data, mekanisme ini juga menyediakan layanan kerahasian data.

Mekanisme Keutuhan data Digunakan untuk memastikan keutuhan data pada unit atau pada suatu aliran (*stream*) data unit. Cara lain yang digunakan adalah dengan menambahkan nilai penguji (*check value*) pada data asli, dimana pada proses pengiriman pihak penerima dapat menguji apakah data yang dikirim sama dengan data asli. Mekanisme Digital Signature Merupakan mekanisme keamanan jaringan yang menyediakan cara bagi pengirim data untuk "manandatangani" secara elektronik sebuah data dan penerima dapa memverifikasi "tanda tangan" itu secara elektronik. *Digital Signature* ditambahkan pada data unit dan digunakan sebagai bukti sumber pengirim dan menghindari pemalsuaan (*forgery*) tanda tangan. *Authentication Exchange* Mekanisme ini memberikan cara agar dua entitas dapat saling meng-ontentikasi dengan cara bertukar pesan untuk saling membuktikan identitas. Mekanisme *Traffic padding* menyediakan cara untuk pencegahan lalu lintas data pada jaringan yaitu dengan menambahkan data palsu pada lalu lintas data, sedangkan mekanisme *Routing Control* menyediakan cara untuk memilih dan secara terus menerus mengubah alur (*route*) pada jaringan komputer antara pengirim dan penerima. Mekanisme ini menghindarkan komunikasi dari penguping (*eavesdropper*), dan Mekanisme Kendali Akses Memberikan cara bagi pengguna untuk memperoleh hak akses sebuah data. Misalnya dengan table ralasi pengguna dan otoritasnya (kemampuan aksesnya)

B. Kriptografi

Kriptografi *(cryptography)* berasal dari bahasa yunani yang terdiri dari suku kata yaitu *kryptos* yang artinya tersembunyi dan *graphein* yang artinya tulisan. Jadi kata kriptografi dapat diartikan sebagai frase "tulisan tersembunyi". Pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana menyembunyikan pesan oleh karena itu kriptografi ini disebut juga sebagai kriptografi klasik. Kriptografi saat ini atau kriptografi moderent dikenal sebagai ilmu yang bersandarkan pada teknik matematika dalam mengamankan informasi seperti kerahasiaan., keutuhan data dan otentikasi entitas, oleh karena itu saat ini kriptografi tidak hanya menyembunyikan pesan namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi. *Cryptanalysis* adalah suatu ilmu dan seni membuka (*breaking*) *ciphertext* dan orang yang melakukan tindakan tersebut disebut sebagai cryptanalyst.
Sistem kriptografi terdiri dari 5 bagian (stinson,2002):

Plaintext

Pesan atau data dalam bentuk aslinya yang dapat terbaca. *Plaintext* adalah masukan bagi algoritma enkripsi. Untuk selanjutnya digunakan istilah teks asli sebagai padanan kata *plaintext*.

Secret Key

Secret key yang juga merupakan masukan bagi algoritma enkripsi merupakan nilai yang bebas terhadap teks asli dan menentukan hasil keluaran algoritma enkripsi. Untuk selanjutnya digunakan istilah kunci rahasia padanan kata *secret key*.

Ciphertext

Ciphertext adalah keluaran algoritma enkripsi. *Ciphertext* dapat dianggap sebagai pesan dalam bentuk tersembunyi. Algoritma enkripsi yang baik akan menghasilkan *ciphertext* yang terlihat acak. Untuk selanjutnya digunakan istilah teks sandi sebagai padanan kata *ciphertext.*

Algoritma Enkripsi

Algoritma enkripsi memiliki 2 buah masukan teks asli dan kunci rahasia. Algoritma enkripsi melakukan transformasi terhadap teks asli sehingga menghasilkan sandi teks.

Algoritma Deskripsi

Algoritma deskripsi memiliki 2 masukan yaitu teks sandi dan kunci rahasia. Algoritma ini memulihkan kembali teks sandi menjadi teks asli bila kunci rahasia yang dipakai algoritma deskripsi sama dengan kunci rahasi yang dipakai algoritma enkripsi.

Analisis sandi (*cryptanalysis*) adalah ilmu yang digunakan untuk memecah teks sandi sehingga terungkap teks asli dan kunci rahasianya. Sistem kriptografi yang aman haruslah tahan terhadap serangan analisis sandi.

C. Algoritma

Dalam ilmu matematika dan ilmu komputer, algoritma adalah kumpulan intruksi yang terdeskripsi dengan jelas isi dan urutan pengerjaannya, yang bertujuan untuk menyelesaikan suatu tugas, dengan mendefinisikan kondisi awal dan akan berakhir pada kondisi akhir yang telah ditentukan pula. Konsep dari algoritma, secara informasi dapat digambarkan sebagai

sebuah resep dimana setiap permasalahan akan dicampurkan dengan berbagai penyelesaian kemudin akan melihat mana yang terbaik.

Pada dasarnya terdapat tiga buah struktur dasar yang menyusun suatu algoritma yaitu struktur Sekuensial, Struktur Seleksi, dan Struktur Pengulangan.

Sejauh ini tidak ada standarisasi tentang bagaiman menyusun algoritma. Secara prinsip kita memiliki kebebasan untuk menyusun bentuk suatu algoritma. Walaupun begitu ada beberapa hal yang perlu diperhatikan dalam menyusun suatu algoritma menurut Knuth (1973) dan Horowitz (1999) ada lima ciri penting yang harus dimiliki sebuah Algoritma yaitu *Finiteness* yang menyatakan bahwa suatu algoritma harus berakhir untuk semua kondisi setelah memproses sejumlah langkah, d*efiniteness* yang menyatakan bahwa setiap langkah harus dinyatakan dengan jelas (tidak rancu atau mendua-arti), masukan dimana setiap algoritma bisa tidak memiliki masukan atau satu atau beberapa masukan. Masukan merupakan suatu besaran yang diberikan diawal sebelum algoritma diproses. Keluaran hal mana setiap algoritma memiliki keluaran, entah hanya sebuah atau banyak keluaran. Keluaran merupakan besaran yang mempunyai kaitan atau hubungan dengan masukan. Dan yang terakhir adalah e*fektivitas*, dimana setiap algoritma diharapkan bersifat efektif, dalam arti semua operasi yang dilaksanakan oleh algoritma haruslah sederhana dan dapat dikerjakan dalam waktu terbatas.

Algoritma kriptografi adalah metode yang digunakan dalam kriptografi untuk mengubah data berupa *plaintext* menjadi *ciphertext* sehingga dapat membuat ata yang ingin dikirimkan menjadi rahasia dan hanya dapat diakses oleh *user-user* yang mengetahui *key* untuk mendeskripsikan *ciphertext* tersebut.

Ariyus (2008) Algoritma kriptografi terdiri dari 3 fungsi dasar yaitu enkripsi, deskripsi dan kunci.

Enkripsi merupakan hal yang sangat penting dalam kriptografi, merupakan pengamanan data yang dikirim agar terjaga kerahasiannya. Pesan asli disebut sebagai *plaintext*, yang dirubah menjadi kode-kode yang tidak dimengerti. Enkripsi bisa diartikan sebagai cipher atau kode. Sama halnya dengan kita tidak mengerti akan sebuah kata maka kita akan melihatnya didalam kamus atau daftar istilah. Beda halnya dengan enkripsi, untuk mengubah teks asli ke bentuk teks-kode kita menggunakan algoritma yang dapat mengkodekan data yang kita ingini.

Deskripsi merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks-asli), disebut dengan enkripsi pesan. Algoritma yang digunakan untuk eskripsi tentu berbeda dengan algoritma yang digunakan untuk enkripsi.

Kunci yang dimaksud disini adalah kunci yang dipakai untuk melakukan enkripsi dan deskripsi. Kunci terbagi menjadi 2 bagian, kunci rahasia (*private key*) dan kunci publik (*public key*).

Algoritma Kriptografi dibagi menjadi tiga karakteristik yaitu tipe operasi yang dipakai dalam proses enkripsi dan deskripsi adalah tipe subsitusi, transposisi, dan karakteristik lainnya yaitu tipe operasi pengolahan kunci yang dipakai kriptografi kunci simetris atau yang disebut *Public Key* dan kriptografi kunci Asimetri atau yang dikenal sebagai kriptografi *Private Key* dan tipe pengolahan pesan yaitu bagaimana *plaintext* diproses dibagi menjadi *block cipher* dan *stream cipher.*

D. Block Cipher

Pada *Block Cipher*, rangkaian bit-bit *plaintext* dibagi menjadi blok-blok bit dengan panjang yang sama, biasanya 64 bit (tapi adakalanya lebih). Algoritma enkripsi menghasilkan blok *ciphertext* yang pada kebanyakan sistem kriptografi simetri berukuran sama dengan blok *plaintext*. Dengan *blockcipher*, *block plaintext* yang sama akan dienkripsi menjadi *block ciphertext* yang sama bila digunakan kunci yang sama pula. Ini berbeda dengan cipher aliran dimana bit-bit *plaintext* yang berbeda setiap kali dienkripsi.

Misalkan blok *plaintext* (P) yang berukuran m bit dinyatakan sebagai vektor

$$P = (p_{1,} p_{2,\ldots} p_m) \quad (1)$$

Yang dalam hal ini p_1 *adalah bit 0* atau bit 1 untuk i = 1,2,…m dan *blokciphertext* (C) adalah

$$C = (c_{1,} c_{2,\ldots} c_m) \quad (2)$$

Yang dalam hal ini c_1 adalah bit 0 atau bit 1 untuk i = 1,2,…,m. bila *plaintext* dibagi menjadi m buah blok, barisan blok-blok *plaintext* dinyatakan sebagai berikut

$$P = (P_{1,} P_{2,\ldots} P_m) \quad (3)$$

Untuk setiap blok *plaintext* P_1 penyusunnya dapat dinyatakan sebagai vektor

$$P = (P_{1,} P_{2,\ldots} P_{im}) \quad (4)$$

Enkripsi dengan kunci *K* dinyatakan dengan persamaan

$$Ek(P) = C \quad (5)$$

Sedangkan deskripsi dengan kunci *K* dinyatakan dengan persamaan

$$Dk\ (C) = P \quad (6)$$

Fungsi E haruslah fungsi yang berkoresponden satu-ke-satu sehingga

$$E^{-1} = D \quad (7)$$

.

Mode operasi *Cipher block Chaining* (CBC) bekerja dengan menggunakan larik khusus yang disebut IV seukuran dengan ukuran blok (n bit). Untuk blok-1, hitung hasil teks sandinya sebagai berikut :

Enkripsi Mode Opersi CBC

Input: IV, $\{P_1,..,P_N\}$, *K*
Output: $\{ C_{1,}\ldots,C_N\}$
C_0 = IV
for I = 1 → *N* do
$C_i = \text{enc}_K (C_{i-1} \oplus P_i)$
End for (8)

Deskripsi Mode Operasi CBC

Input: IV, $\{ C_{1,}\ldots,C_N \}$, *K*
Output: $\{ P_1,..,P_N \}$
C_0 = IV
for I = 1 → *N* do
$P_i = C_{i-1} \oplus \text{dec}_K (C_i)$
end for (9)

Mode Opereasi *counter* adalah Mode operasi yang digunakan untuk mngubah nilai IV.

Input : IV, { P_1,…..P_N}, K, r
Output : {C_1,….,C_N}
CTR = IV
For *i* = 1-*N* do
$T = enc_k$ (CTR)
$C_i = T\ P_i \bigoplus$
CTR + +
end for (10)

Mode Oprasi CTR (*Counter*) pada pemilihan nilai awalnya direkomendasikan berbeda untuk setiap penyandian pesan. Meskipun menggunkan sebuah *counter* dapat dibuktikan bahwa mode operasi CTR (*counter*) setidaknya memiliki keamanan yang sama dengan mode operasi lainnya. Mode operasi CTR (*counter*) dapat diimplementasikan secara efisien diperangkat keras. Selain itu, mode operasi CTR (*counter*) memiliki struktur enkripsi, dan deskripsi yang sama. Mode operasi ini sering direkomendasikan pada standar ATM (*Asynchoronous Transfer Mode)* dan IPSec (*IP Security*).

E. Algoritma Serpent

Serpent adalah Algoritma Kriptografi yang bersifat *block cipher symmetric cryptography* yang merupakan AES finalis pada kontes AES. Algoritma Serpent ditemukan oleh Ross Anderson, Eli Biham dan Lars Knudsen.

Seperti halnya AES yang lain, serpent mempunyai ukuran blok sebesar 128 bit dan dapat mendukung ke dengan ukuran 128 bit, 192 bit, ataupun 256 bit. Serpent mengimplementasikan kriptosistem 32 tahap *Subsitution-Permutation Network* (SP Network), dimana tahap-tahap tersebut mengoperasikan empat buah variable dengan ukuran 32 bit. Masing-masing tahap menggunakan 32 buah 1 *bitslice*. Hal ini memaksimalkan pararelisme, tetapi juga memungkinkan adanya kriptonalisis yang luas.

Serpent dapat digambarkan sebagai *Subsitution-permutation Network* sebanyak 32 tahap, yang mengoperasikan 4 buah variabel 32 bit, sehingga ukuran *block plaintext* adalah 128 bit. Algoritma Serpent sendiri terdiri dari Permutasi awal 32 tahap *Subsitutition Permutation Network*, terdiri dari operasi pencampuran *key*, melalui *S-Box* dan transformasi linear kecuali pada tahap terakhir. Pada tahap terakhir, transformasi linear diganti menjadi operasi pencampuran *key*. Dan Permutasi akhir. Algoritma kriptografi dapat digambarkan sebagai berikut :

$B_0 = IP\ (P)$
$B_{1+1}=R_i(B_i)$
$C=FP(B_{32})$

Dimana

$R_i(x) = S_i(X \oplus K_i) \oplus K_{32}$ (11)

S-Box pada serpent adalah permutasi 4 bit dengan sifat-sifat, Masing-masing karakteristik diferensial memiliki probabilitas paling besar $^1/_4$ dan sebuah perbedaan input 1 bit akan menghasilkan perbedaan yang besar. Masing-masing karakteristik linear memiliki probabilitas $^1/_2 \pm {}^1/_4$ dan hubungan linear antara sebuah bit input dan sebuah bit output mempunyai probabilitas diantara $^1/_2 \pm {}^1/_8$. Dan Deretan non-linear bit output merupakan fungsi dari input bit adalah maksimum bernilai 3.

Algoritma kriptografi ini, dibutuhkan 132 kunci variabel berukuran 32 bit. Pada awalanya *user* menyediakan *key* dengan 256 bit, kemudian diperluas *key* tersebut hingga menjadi 33 buah *subkey* ($K_0, K_1, K_2, \ldots K_{32}$) dengan ukuran 128 bit. Kita menulis *key* K sebagai delapan buah variabel 32 bit w-8,w-7,w-6,….,w-1 dan memperluas variabel-variabel tersebut menjadi deretan panjang *key* (dimana kita menyebutnya *prekey*) $w_0, w_1, w_2, \ldots, w_{31}$ dengan rekusi dan transformasi *affine* sebagai berikut :

$$wi = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus 1) \lll 11 \quad (12)$$

Dimana PHI adalah bagian *fractal* dengan perbandingan emas $(\sqrt{5}+1)/2$ atau 0x9E3779B9 pada *hexadecimal*. Polinom yang mendasari, $x^8 + x^5 + x^3 + 1$ bersifat primitive, dimana bersama dengan *round key* dilakukan untuk memastikan distribusi genap dari *key* pada tahap-tahap, dan untuk menghilangkan *key-key* lemah dan *key* yang saling berhubungan.

Round key dikalkulasi dengan *prekey* dari S-Box. Kita menggunakan *S-Box*. kita menggunakan *S-Box* untuk mentransformasikan *prekey* w*i* dan k_i dengan cara berikut :

{k0, k1, k2, k3} = S3 (w0, w1, w2, w3)
{k4, k5, k6, k7} = S2 (w4, w5, w6, w7)

{k8, k9, k10, k11} = S1 (w8, w8, w10, w11)
{k12, k13, k14, k15} = S0 (w12, w13, w14, w15)
. . .
{k124, k125, k126, k127} = S4 (w124, w125, w126, w127)
{k128, k129, k130, k131} = S3 (w128, w129, w130, w131)

Kemudian kita menomori angka-angka 32 bit *kj* sebagai sub*key* K_i sebagai berikut :

$$Ki = \{k4i, k4i+1, k4i+2, k4i+3\} \quad (13)$$

Saat kita mengimplementasikan algoritma ini, awalnya sesuai dengan yang disebutkan diatas, daripada menggunakan operasi bitslice, sekarang kita menggunakan permutasi awal daripada *roundkey* untuk mendapat posisi bit yang benar pada kolom yang benar. $K_i = IP\ (K_i)$.

F. Socket Programming

Socket adalah sebuah abstraksi perangkat lunak yang digunakan sebagai suatu "terminal" dari suatu hubungan antara dua mesin atau proses yang saling berinterkoneksi (Wiharto,2010).

Socket dapat melakukan beberapa operasi yakni: koneksi ke mesin *remote*, mengirim data (*write*), menerima data (*read*), menutup koneksi (*close*), *bind to port, listen* pada data yang masuk dan menerima koneksi

dari mesin *remote* pada *port* tertentu. *Protocol* yang digunakan dalam *socket* dapat menggunakan *Transmission Control Protocol* (TCP) ataupun *User Datagram Protocol*(UDP).

Dalam *protocol* jaringan, TCP/IP, sebuah *port* adalah mekanisme yang mengizinkan sebuah komputer untuk mendukung beberapa sesi koneksi dengan komputer lainnya dan program didalam jaringan. *Port* dapat mengidentifikasi aplikasi dan layanan yang menggunakan koneksi di dalam jaringan TCP/IP. Sehingga, *port* juga mengidentifikasi sebuah proses tertentu dimana sebuah server dapat memberikan sebuah layanan kepada klien atau bagaimana sebuah klien dapat mengakses sebuah layanan yang ada dalam *server*. *Port* dapat dikenali dengan angka 16 bit(dua *byte*) yang disebut *port* number dan diklasifikasikan dengan jenis protokol trans*port* apa yang digunakan, ke dalam server. *Port* dapat dikenali dengan angka 16 bit (dua byte) yang disebut dengan *port* number dan diklasifikasikan dengan jenis *protocol* trans*port* apa yang digunakan, ke dalam *port* TCP dan *port* UDP. Karena memiliki 16-bit, maka total maksimum jumlah *port* untuk setiap protokol trans*port* yang digunakan adalah 65536 buah. Pada penomoron *port* TCP dan UDP dibagi menjadi tiga jenis, yakni, *Well-known Port* yang pada awalnya berkisar antara 0 hingga 255 namun kemudian diperlebar untuk mendukung antara 0 – 1023. *Port* number yang termasuk ke dalam *well-known Port*, selalu merepresentasikan layanan jaringan yang sam, dan ditetapkan oleh *Internet Assigned Number Authority* (IANA). Beberapa diantara *port-port* yang berada didalam rang *well-known Port* masih belum ditetapkan dan direservasikan untuk digunakan oleh layanan yang bakal ada dimasa depan. Sebagai contoh www. Atau http ada di*port* 80 , sedangkan email di*port* 25. *Registered Port* yang ada pada *Port-port* yang digunakan oleh vendor-vendor komputer atau jaringan berbeda untuk mendukung aplikasi dan sistem operasi yang mereka buat. *Registered port* juga diketahui dan didaftarkan oleh IANA tapi tidak dialokasikan secara permanen, sehingga vendor lainnya dapat menggunakan *port number* yang sama. *Range Registered port* berkisar dari 1024 hingga 49151 dan beberapa *port* antaranya adalah *Dynamically Assigned Port. Dynamically Assigned Port* Merupakan *port-port* yang ditetapkan oleh sistem operasi atau aplikasi yang digunakan untuk melayani request dari pengguna sesuai dengan kebutuhan. *Dynamically Assigned Port* berkisar dari 1024 hingga 65536 dapat digunakan atau dilepas sesuai kebutuhan.

Socket merupakan fasilitas IPC (*Inter Proses Communication*) untuk aplikasi jaringan. Agar suatu *socket* dapat berkomunikasi dengan *socket* lainnya maka *socket* butuh suatu alamat unik sebagai identifikasi. *Socket* terdiri atas alamat IP dam Nomor *Port* yang di gunakan untuk komunikasi antara *Client* dan *Server*.

III. METODOLOGI PENELITIAN

A. Tempat dan Waktu Penelitian

Dalam pelaksanaan tugas akhir ini penulis mengambil tempat penelitian pada Ruang Laboratorium Sistem Komputer (LSK), Jurusan Teknik Elektro, Fakultas Teknik Universitas Sam Ratulangi (UNSRAT), dan rumah penulis.

B. Bahan dan Peralatan

Dalam mengerjakan tugas akhir ini mulai dari mendesain sampai tahap pemrograman penulis menggunakan perlengkapan komputer sebagai media untuk menjalankan program. Secara lebih spesifik perlengkapan komputer beserta pendukung yang digunakan yaitu Intel Core Intel Core i5-450M 1066MHz, Memory RAM DDR 2 GB, Harddisk 500Gb HDD, sistem operasi Windows 7, NetBeans 6.8.

C. Prosedur Penelitian

Prosedur yang dilakukan dalam pembuatan Aplikasi pengiriman suara terenkripsi ini adalah sebagai berikut :

Sebelum melakukan penelitian, penulis terlebih dahulu melakukan studi literatur. Penulis mencari materi-materi yang berhubungan dengan pembuatan Aplikasi pengiriman suara terenkripsi dengan menggunakan algoritma serpent dan mencari materi bahasa pemrograman java. .

Setelah mendapatkan informasi yang dibutuhkan, maka penulis mencari program-program pendukung dalam pembuatan tugas akhir.

Penulis menggunakan bahasa pemrograman java dengan IDE NetBeans 6.8 dan melakukan instalasi java dan NetBeans. Setelah itu penulis melakukan konfigurasi Algoritma Serpent pada java untuk merancang suatu aplikasi pengiriman suara terenkripsi.Kemudian penulis mengimplentasikan konfigurasi Algoritma serpent pada java dengan menggunakan NetBeans sebagai IDE.

Penulis menguji Aplikasi ini dengan menghubungkan 2 buah perangkat laptop pada jaringan komputer yang sama untuk membuktikan pada komunikasi 2 arah ini ada proses enkripsi dalam pengirimannya.

D. Perancangan sistem

Dalam pembuatan tugas akhir ini, program Aplikasi pengiriman suara ini akan di ujicobakan menggunkan 2 buah perangkat laptop dan direncanakan dibuat bersifat real time, dimana proses enkripsi dan deskripsi dilakukan melalui suatu jaringan yang sama dengan pengiriman suara bersifat dua arah. Aplikasi ini dirancang menggunakan menggunkan Algoritma Serpent yang di implementasikan pada *java.* Aplikasi ini menggunakan *NetBeans* sebagai IDE (*Integrated Development Environment*) . Aplikasi ini memberikan kemampuan untuk pengguna untuk mendengarkan, berbicara dan menggunakan fitur *chat* yang ada.

Alur aplikasi ini dimulai dengan pengguna mengakses menu awal untuk masuk dalam menu utama aplikasi. Kemudian aplikasi akan menampilkan menu utama yang diikuti dengan memasukan alamat IP dan nama pengguna. Setelah alamat IP dan nama pengguna dimasukan aplikasi ini akan menampilkan daftar IP yang menggunakan aplikasi ini pada jaringan yang sama. Kemudian pilih IP yang akan dihubungi. Setelah terhubung aplikasi ini akan memunculkan pilihan tombol *Talk* atau sebuah text box yang disediakan untuk

*chat*ting. Alur aplikasi ini akan berhenti ketika *user* menekan tombol Stop *Talk* untuk memutuskan hubungan . Sistem aplikasi ini memberikan kemampuan bagi pengguna untuk melakukan 3 hal yaitu mengirimkan suara, mendengarkan suara dan menggunkan fitur *Chat*.

Pada Diagram *Use Case* ini dapat kita lihat pengguna bukan saja hanya memiliki kemampuan melakukan pengiriman suara saja namun dapat melakukan hal lainnya yaitu mendengarkan suara dari pengguna lainnya. Selain itu pengguna dapat menggunakan fitur yang lainnya yaitu fitur *Chat*.

Diagram Use Case Komunikasi Suara terenkripsi

Use Case Diagram gambar 2 menunjukan proses komunikasi suara yang dilakukan oleh dua buah pengguna. Pada *Use Case* diagram dapat dilihat bahwa proses ini dilakukan dengan menggunakan *Microphone* sebagai input suara dan *Headset* sebagai Output suara. Saat proses komunikasi suara dilakukan sebuah proses enkripsi terjadi ketika suara dari pengguna dikirimkan pada pengguna lainnya melalui *microphone*. Dan ketika pengguna yang lainnya akan mendengarkan suara tersebut operasi deskripsi akan berlangsung. Setelah proses deskripsi selesai pengguna yang akan menerima informasi tersebut dapat mendengarkan dan mengerti maksud informasi yang diterimanya.

Diagram Use case Melakukan Chat

Use case diagram gambar 3 memperlihatkan bagaimana pengguna menggunakan fitur *Chat* pada aplikasi ini. Fitur in digunakan ketika pengguna ingin menggunakannya dan seperti yang terlihat pada sistem ini berbeda saat melakukan pengiriman dan mendengarkan suara . Pada sistem ini tidak terjadi proses enkripsi saat pengiriman maupun deskripsi saat penerimaan informasi.

Diagram Aktivitas

Pada diagram diatas kita dapat melihat bagaimana alir aktivitas dari sistem yang akan dirancang, bagaimana masing-masing alir berawal dan keputusan yang mungkin terjadi, dan bagaimana sistem ini berakhir. Seperti yang dapat dilihat pada diagram ini alir aktivitas dimulai pada saat menu utama ditampilkan.

Pada menu utama ini ada beberapa pilihan yang dapat ditentukan oleh pengguna. Untuk menggunkan fitur yang berada pada aplikasi ini pengguna diharukan menghubungkan aplikasi dengan server , itu dilakukan dengan cara mengisi nomor IP server dan menghubungkannya dengan menekan tombol *Connect* kemudian pengguna perlu memasukan data yaitu Nama pengguna. Pilihan lainnya yaitu pilihan untuk berkomunikasi dengan pengguna lainnya.

Hal ini dilakukan dengan cara memilih pengguna lainnya pada daftar *User list* yang ditampilkan kemudian jika *user* memilih untuk melakukan komunikasi dengan pengguna lainnya, pengguna dapat memilih Tombol *Talk* atau mengetik pada *text box* yang disediakan untuk penggunaan fitur *chat*ting. Untuk keluar dari aplikasi ini pengguna dapat menekan tombol keluar kemudian aktivitas dari aplikasi ini akan selesai.

Pada gambar diagram *Use Case* kemampuan Aplikasi (Gambar 1) untuk pengguna dimana pengguna dapat melakukan 3 hal Mendengarkan, Berbicara , dan menggunakan fitur *chat*.

Pada gambar 2 diagram *use case,* menunjukkan komunikasi suara yang telah terenkripsi dan dapat kita lihat bagaimana proses terenkripsinya suara.

Pada gambar 3 diagram *Use Case*, menunjukkan bahwa antar *user* dapat melakukan *chat.* Pada gambar ini terdapat tahap – tahap dan fitur *chat* yang harus dilakukan agar dapat menggunakan fitur tersebut.

Pada Gambar berikutnya terdapat diagram *class* (Gambar 4). Pada diagram kelas ini diperlihatkan kelas apa saja yang diimplementasikan pada aplikasi ini. Pada aplikasi pada *project Chat_client* dignakan tujuh kelas yaitu *ChatClient, Recorder, Playback, CommonSound, Queue, Serpent dan StreamInputSerpent*.

E. Perancangan Antar Muka

Pada Tampilan pembuka (Gambar 5) ditampilkan menu awal dari program . Menu awal inilah yang akan muncul pertama kali ketika aplikasi ini akan dijalankan . Menu awal ini merupakan jembatan untuk dapat mengakses aplikasi ini ketika pengguna ingin menggunkannya. Pada menu ini terdapat pilihan enter yang dapat dipilih oleh pengguna sewaktu menjalankan aplikasi ini. Pada saat pengguna memilih tombol *Enter* maka menu utama pada aplikasi akan ditampilkan.

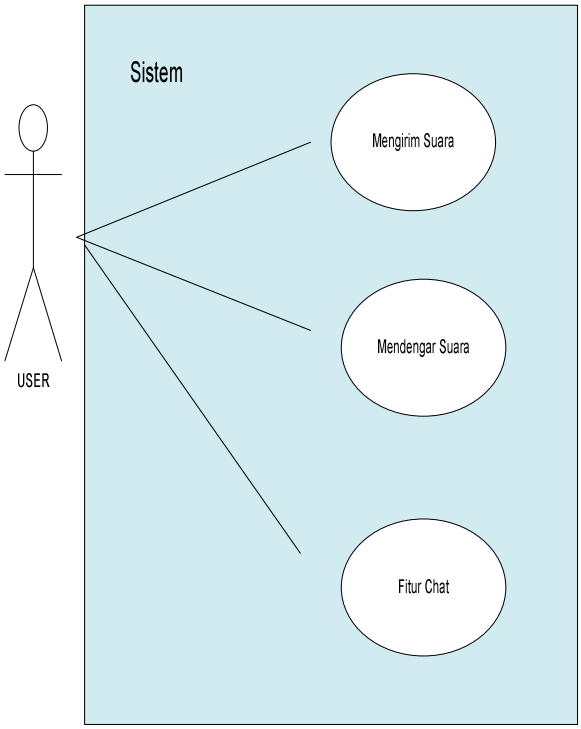
Pada tampilan menu ini (Gambar 6) yang merupakan Menu utama terdapat pilihan *Help , IP Server*, *Connect*, Masukan Nama, *User List*, *Talk* dan *Text box Chat*. Untuk menu *Help Contents* ini dimulai dulu dengan aplikasi menampilkan menu *Help*. Pengguna kemudian memilih tombol *Help* pada menu, setelah itu menu akan menampilkan pilihan pada menu *help*.

Kemudian *User* akan memilih pilihan *Help Contents*.Selain pilihan *Help Contents* pada menu *help* ada pilihan lainnya yaitu *About*. Menu ini dimulai setelah pengguna memilih tombol *Help* yang kemudian akan ditampilkan dua pilihan menu *help* yaitu, *Help contents* dan *About*.

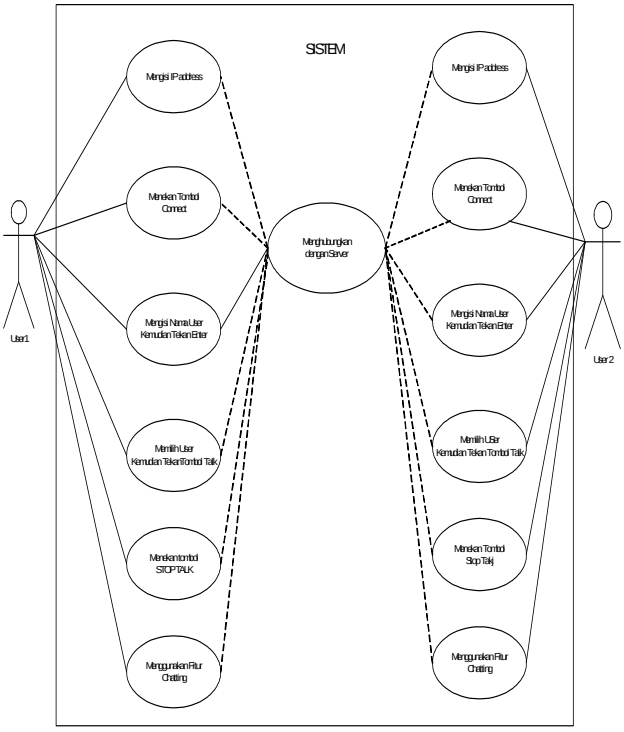
Kemudian ada pilihan IP Server. Pada pilihan IP server ini pengguna diharuskan untuk memasukan nomor . jaringan *server* agar dapat terhubung dengan *server*.

Aplikasi ini akan dimulai ketika pengguna memasukan *IP server* agar aplikasi terhubung dengan *server*. Setelah terhubung pada *server* perintah untuk memasukan data dari pengguna yaitu nama ditampilkan pada *user list*, maka pengguna melihat pengguna lainnya yang menggunkan aplikasi tersebut.

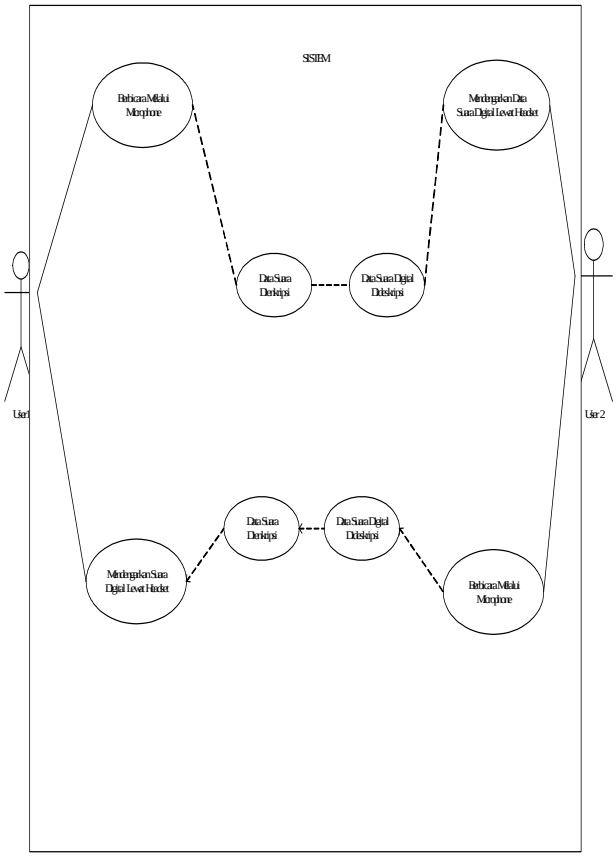
Setelah memilih pengguna lainnya yang akan dihubungi , pengguna dapat memilih tombol *Talk* agar dapat terhubung dengan *user* lainnya dalam komunikasi suara pada jaringan yang sama. Ketika terhubung maka tombol *talk* pada aplikasi ini akan menyala dan berwarna biru. Ketika tombol *Talk* telah dipilih oleh pengguna maka pengguna dapat terhubung dengan pengguna dapat



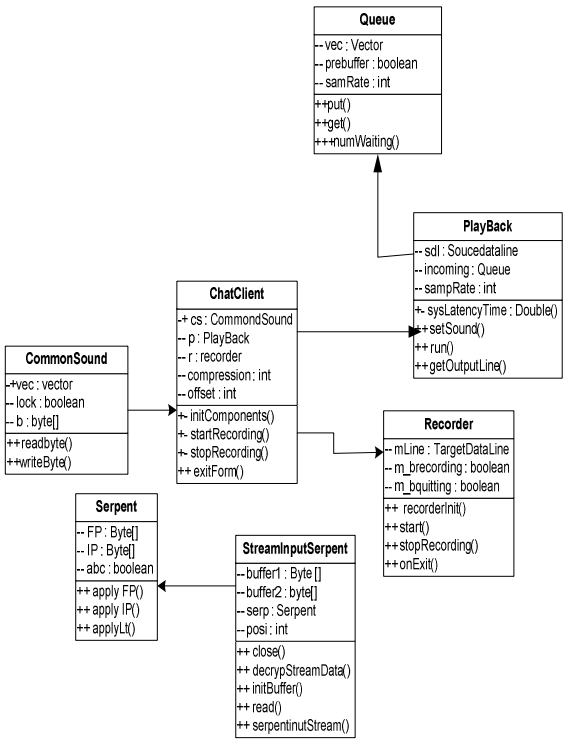
Gambar 1. Diagram *Use Case* Kemampuan Aplikasi.



Gambar 3. Diagram *Use Case* Penggunaan Fitur *chat*



Gambar 2. Diagram *Use Case* Komunikasi Suara yang Terenkripsi.

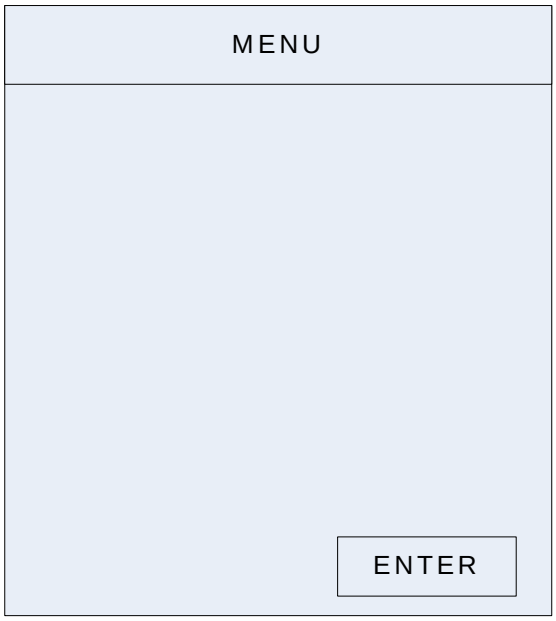


Gambar 4. Diagram Class

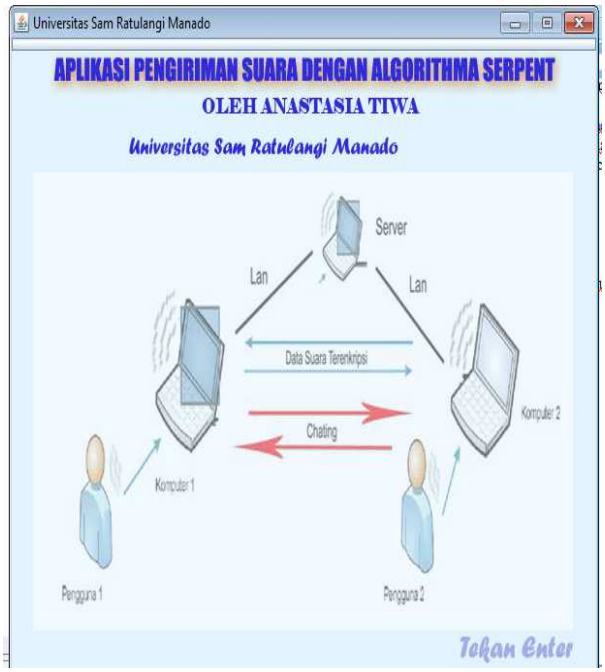
dengan pengguna lainnya dalam komunikasi suara pada satu jaringan yang sama dengan pengguna lainnya yang menggunkan aplikasi tersebut.

Setelah memilih pengguna lainnya yang akan dihubungi , pengguna dapat memilih tombol *Talk* agar dapat terhubung dengan *user* lainnya dalam komunikasi suara pada jaringan yang sama.

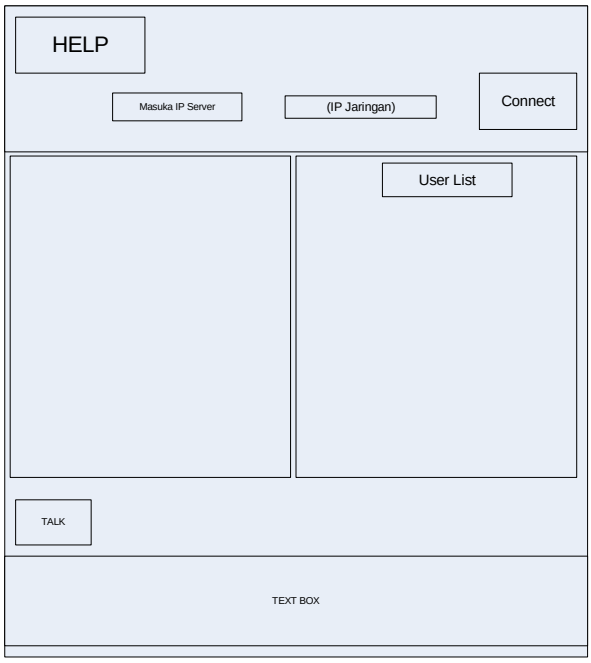
Ketika komunikasi suara ingin dihentikan maka pengguna dapat mengklik tombol *stop* yang berada pada tempat dimana tombol *talk* dipilih, karena pada saat tombol *talk* dipilih maka tombol *talk* akan diganti dengan tombol *stop*.



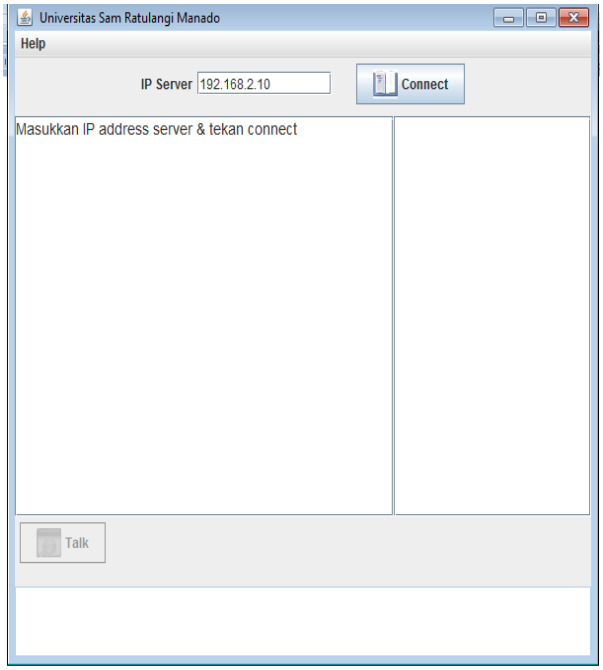
Gambar 5. Menu Awal Aplikasi



Gambar 6. Tampilan Antar Muka Menu Awal



Gambar 6. Menu Utama Aplikasi



Gambar 8. Tampilan Antar Muka Menu Utama

Tombol *Stop* ini berfungsi sebagai pemutus hubungan komunikasi suara antara *user*. Selain itu ketika sedang terhubung dengan server maka maka pengguna juga seperti yang dijelaskan pada Diagram *Use Case* Kemampuan Aplikasi lainnya ada fitur *chat* yang dapat digunakan oleh pengguna aplikasi selain dari pada penggunaan fitur *Talking*. Jadi ketika pengguna ingin menggunakan fitur ini caranya mudah saja dimana aplikasi ini menyediakan sebuah *text box* yang disediakan untuk fitur ini.

Jadi caranya sama saja dengan penggunaan fitur *Talking* dimana pengguna dapat memilih *User* yang akan dihubungi pada *UserList* yang terdapat pada aplikasi ini sehingga fitur *chat* ini dapat berjalan dengan sebagaimana mestinya aplikasi ini berjalan.

Setelah menggunakan fitur ini *user* dapat langsung menutupnya.

IV. PENGUJIAN

Pengujian Perangkat Lunak Aplikasi Pengiriman Suara Terenkripsi dilakukan untukmelihat bagaimana kinerja dari aplikasi ini. Selain itu, Pengujian ini dilakukan untuk melihat bagaimana Algoritma Serpent diimplementaikan pada bahasa pemrograman java. Pada

Aplikasi ini penulis menguji pengiriman suara yang berlangsung data suara yang dikirim dan diterima sama. Waktu yang dibutuhkan untuk menerima cukup lama karena terjadi *delay* akibat proses enkripsi dan deskripsi . *Delay* atau waktu tunda yang dihasilkan untuk proses ini adalah Latar suara yang terdengar sedikit mengganggu akibat proses enkripsi. Implentasi Algoritma Serpent untuk mengenkripsi suatu pesan dapat dibutikan dengan terjadinya pergantian *byte*.

V. KESIMPULAN

Berdasarkan implementasi dan uji coba yang dibangun, penulis dapat menarik beberapa kesimpulan, yaitu Dalam Implementasi Algoritma Serpent pada pengiriman suara cukup baik, namun masih ada beberapa hal perlu diperhatikan antara lain Data Suara harus diambil secara bertahap untuk menjaga *property real time*, Proses enkripsi dan deskripsi mempunyai proses yang sama. Kualitas Suara baik mengingat adanya proses enkripsi dan deskripsi. *Delay* yang cukup besar karena proses pengiriman dan penerimaan. komunikasi baik walaupun sering waktu tunda yang cukup besar terjadi. Suara tidak dipengaruhi besarnya data suara masukan.

DAFTAR PUSTAKA

[1] A. Kadir, "*Algoritma Dan Pemogramn Menggunakan Java*", Andi, Yogyakarta, 2012.

[2] Anderson, Ross, Eli Bilham, dan Lars Knudsen, "*Serpent : A Proposal for the Advance Encryption Standart*",2011

[3] Anggi. *Tugas akhir: Studi Dan Implementasi Enkripsi Pengiriman Suara Dengan Algoritma Serpent*. Jurusan Teknik Informatika, Fakultas Teknik Elektro dan Informatika, Instutut Teknologi Bandung.

[4] B. Hariyanto, Dr. , "*Esensi-esensi Bahasa Pemrograman Java Revisi 4*", Informatika, Bandung, 2011.

[5] Bora, Piotr. "*Tomascs czakka. Implementation of Serpent Algorithm Using Altera FPGA Device*". Military Communication Institute.

[6] M. Syarif, "*Bermacam-macam Project Java Dengan IDE Netbeans*", Andi, Yogyakarta, 2010.

[7] R. Dantas, " *Netbeans IDE 7 CookBook*", Packt, Mumbai, 2011

[8] R. Sadikin, "*Kriptografi Untuk Keamanan Jaringan*", Andi, Yogyakarta, 2012.

[9] S. Kromodimoeljo, "*Teori Dan Aplikasi Kriptografi. SPKTT Consultig*", 2009.

[10] S. Siallagan, "*Pemrograman Java Dasar-dasar Pengenalan Dan Pemahama*" Andi, Yogyakarta, 2009.