BAB III

METODE DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Metode penelitian yang digunakan dalam implementasi algoritma RSA 1024 dan Salsa20 dalam aplikasi berbasis VoIP memiliki beberapa tahap yang terdiri dari :

1. Studi Literatur

   Pada tahap ini, akan dilakukan pembelajaran mengenai teori yang berhubungan dengan implementasi algoritma. Pembelajaran dapat dilakukan dengan cara membaca buku, jurnal dan media *online* seperti *website.*

2. Analisis Kebutuhan Sistem

   Setelah melewati proses telaah literatur selanjutnya adalah melakukan analisis sistem yang akan digunakan apakah *support* atau tidak dengan aplikasi yang ingin dibangun dan diimplementasikan algoritmanya. Aplikasi VoIP membutuhkan perangkat keras seperti *microphone* untuk merekam suara, dan *speaker / headphone* untuk memainkan suara serta jaringan internet untuk melakukan komunikasi data. Perangkat lunak pendukung yang dibutuhkan oleh sistem adalah Visual Studio yang digunakan sebagai IDE (*Integrated Development Environment*) dalam perancangan sistem, MariaDB yang berfungsi sebagai *database* untuk menyimpan data pengguna, dan beberapa *file* DLL (*Dynamic-link library*) untuk membantu sistem agar berjalan.

3. Perancangan Sistem dan Implementasi Algoritma

   Perancangan sistem dan aplikasi terdiri dari *data flow diagram* yang berfungsi untuk mengetahui aliran data dalam aplikasi dan *flowchart* yang merupakan

alur proses dari sistem dan implementasi. Implementasi algoritma digunakan bahasa pemrograman C#.

4. Pengujian Sistem

   Tahap ini merupakan tahap pengujian terhadap sistem yang telah dirancang sebelumnya. Dalam tahap pengujian menggunakan beberapa *tools* yang diantaranya adalah, TCPdump, Wireshark, dan Codetrack. Dalam tahap ini juga terdapat dua uji coba yaitu *functional test* dan *performance test*. *Functional test* dilakukan untuk mengetahui apakah algoritma sudah berhasil diimplementasikan atau belum. *Performance test* bertujuan untuk mengetahui dan membandingkan performa aplikasi saat menggunakan algoritma enkripsi atau tidak dengan melakukan uji coba perbandingan *jitter*, *processing delay*, dan rata – rata *throughput*.

5. Analisa Hasil Uji Coba

   Setelah proses pengujian selesai, maka tahap selanjutnya adalah melakukan analisa data yang diperoleh dari pengujian, dan melakukan perbandingan data aplikasi yang menggunakan enkripsi dengan data yang terdapat pada landasan teori.

6. Penulisan Laporan

   Tahap terakhir dalam penelitian ini adalah dengan menulis laporan yang terdiri dari latar belakang hingga hasil akhir yang didapatkan dari penelitian.

3.2 Perancangan Aplikasi

Perancangan aplikasi dalam penelitian ini meliputi *Data Flow Diagram*, *Flowchart* dan struktur tabel dari *database*. Penjelasan perancangan aplikasi akan dibahas pada subbab ini. Pada tahap perancangan aplikasi *client*, aplikasi VoIP

menggunakan program yang tersedia di codeproject.com karya Hitesh Sharma (Sharma, 2007) dan dimodifikasi sehingga memiliki koneksi secara *hybrid* dengan *authentication* menggunakan TCP, (*Transport Control Protocol*) dan *voice* menggunakan RTP rancangan sendiri yang sebelumnya hanya menggunakan UDP biasa. Aplikasi ini awalnya juga merupakan murni *peer-to-peer application* yang juga dimodifikasi sehingga, aplikasi ini berjalan secara *hybrid* dengan encoding hanya menggunakan A-law atau PCMA (*Pulse-code Modulation A-law*).

Dalam perancangannya algoritma enkripsi Salsa20 dirancang berdasarkan teori yang berasal dari cr.yp.to, dan crypto-it.net. Algoritma enkripsi RSA 1024 menggunakan landasan teori yang terdapat pada Bab 2.

Tahap perancangan server menggunakan koneksi TCP untuk berkomunikasi kepada *client. Server* bekerja untuk menyimpan daftar pengguna beserta alamat IP pengguna dan berfungsi sebagai jembatan untuk *authentication* antara *client* dengan *client* yang lainnya. Dalam implementasinya, algoritma RSA 1024 digunakan untuk enkripsi pengiriman kunci Salsa20 dan algoritma Salsa20 digunakan untuk enkripsi pengiriman pesan suara.

3.2.1 Analisis Kebutuhan Sistem

Pada bagian ini akan diberikan penjelasan mengenai kebutuhan perangkat keras dan perangkat lunak dalam merancang sistem, maupun menggunakan sistem. Dalam tahap perancangan dan penggunaan sistem perangkat keras yang dibutuhkan adalah PC (*Personal Computer*) yang digunakan sebagai sarana untuk melakukan *coding*, membangun aplikasi, dan menjalankan aplikasi, *headphone / speaker* untuk memainkan suara, dan *microphone* untuk merekam suara. Selain perangkat

keras, terdapat perangkat lunak yang digunakan dalam tahap perancangan aplikasi antara lain :

1. Visual Studio Enterprise 2015

   Menurut Microsoft (2017), Visual Studi (Microsoft.com, 2017) merupakan *tools* yang lengkap untuk membangun aplikasi *web* ASP.NET, XML *web services*, aplikasi *desktop*, dan *mobile application*. Visual Studio juga mendukung bahasa pemrograman seperti Visual Basic, Visual C#, dan Visual C++ (Microsoft.com, 2008). Dalam perancangan sistem Visual Studio digunakan sebagai *Integrated Development Environment* dalam perancangan aplikasi VoIP dan versi Visual Studio yang digunakan adalah versi Enterprise 2015.

2. XAMPP

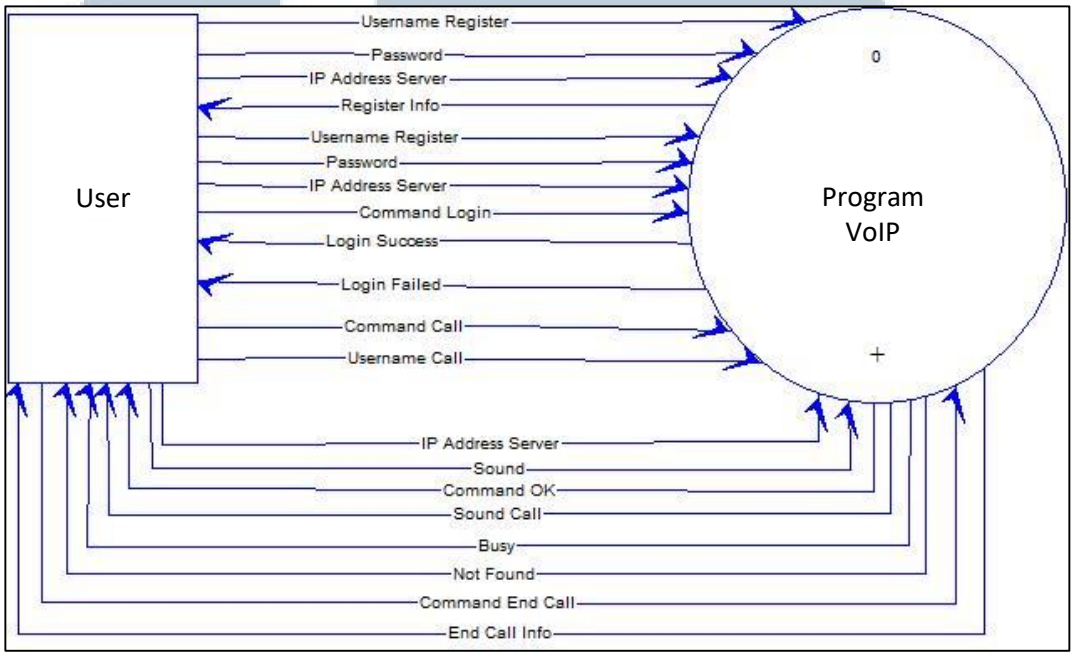
   XAMPP merupakan perangkat lunak yang dirancang oleh tim dari apachefriends.org yang berfungsi untuk memudahkan *developer* untuk melakukan instalasi *web server*, *database* (MariaDB), PHP dan Perl. Dalam perancangan aplikasi yang digunakan untuk aplikasi adalah *database* bernama MariaDB yang disediakan oleh XAMPP. Versi yang digunakan dalam perancangan adalah XAMPP v3.2.2 dan MariaDB versi 10.1.21.

3. Dynamic-link Library

   Menurut Microsoft (2017), *file* DLL adalah sebuah *library* yang mengandung kode dan data yang dapat digunakan oleh lebih dari satu program dalam waktu yang bersamaan. *File* DLL yang digunakan dalam sistem ini adalah MySql.Data.dll yang berfungsi sebagai *connector* antara program dengan *database* MariaDB, dan Microsoft.DirectX.DirectSound.dll yang berfungsi untuk pengolahan suara di dalam program.

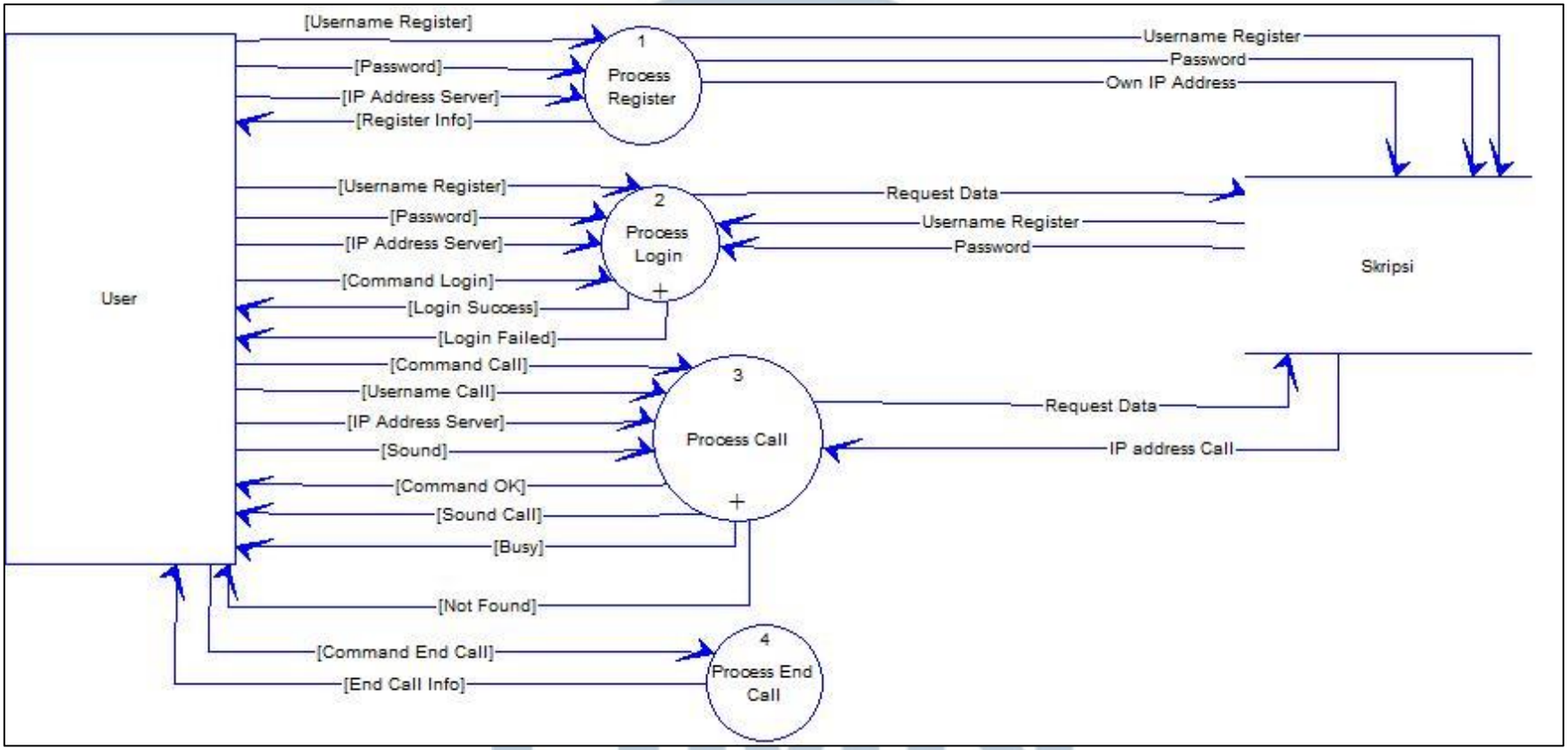
3.2.2 Data Flow Diagram

Pada tahap awal perancangan aplikasi, pertama dilakukan perancangan DFD untuk mengetahui aliran data di dalam program. DFD dalam aplikasi sendiri terdiri sampai level 2.



Gambar 3.1 Context Diagram Program Utama

Pada Gambar 3.1 merupakan *context diagram* pada sistem. *External entity* dalam program utama adalah *user* yang menggunakan program. Pengguna aplikasi hanya user tidak ada entitas lain. Pada *context diagram* yang terdapat pada Gambar 3.1 entitas *user* memberikan *input* yaitu *username register*, *password*, IP *address server*, *command login*, *command call*, *username call*, *sound*, dan *end call info*. Entitas *User* juga akan menerima *output* berupa *register info*, *login success*, *login failed*, *command OK*, *sound call*, *busy*, *not found*, *end call info*.



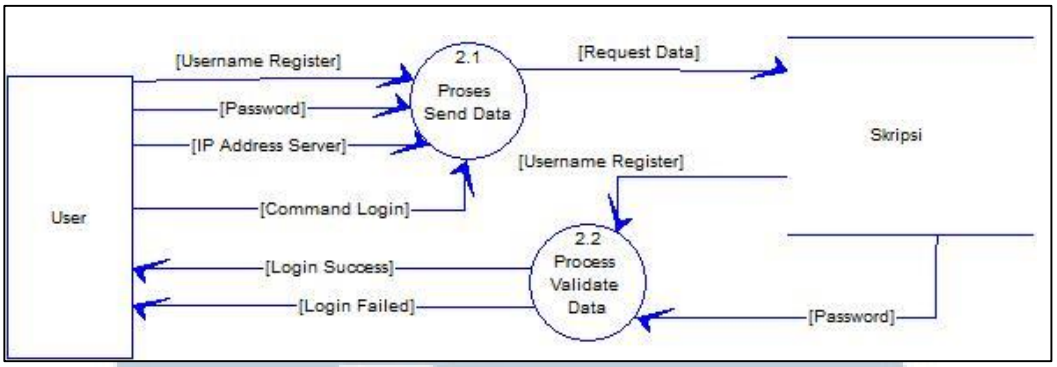
Gambar 3.2 DFD Level 1 Program

Pada DFD *level 1* program, terdapat 4 proses yaitu Proses Register, Proses Login, Proses Call, dan Proses End Call. Proses Register menerima *input* berupa *username register, password,* dan IP *address server*. Keluaran dari Proses Register adalah *username register, password, own* IP *address* yang akan dimasukkan ke *database* dan informasi *register* yang akan diberikan ke *user*.

Proses selanjutnya adalah Proses Login. Proses Login merupakan proses menerima *input* berupa *username register, password,* IP *address server,* dan *command login.* Proses ini meminta data dari *database* dan *database* akan memberikan data berupa *username register, password* kepada proses, dan proses akan memberikan keluaran berupa informasi *login* apakah *login* benar atau salah.

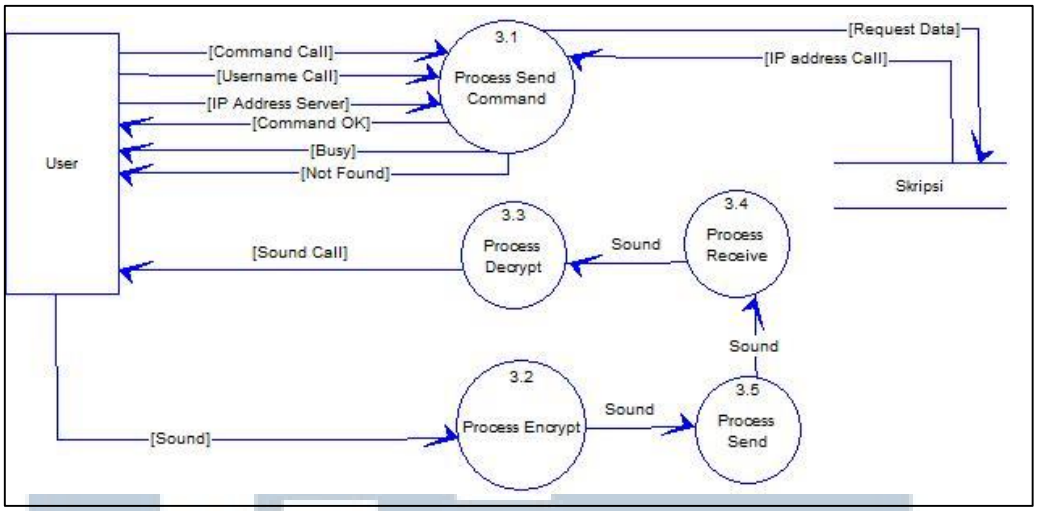
Proses selanjutnya pada DFD *level* 1 adalah Proses Call. Proses Call menerima masukan berupa *command call, username call,* IP *address server* dan *sound.* Selanjutnya Proses Call melakukan *request* data ke *database* dan *database* akan memberikan data IP *address call*. Kemudian Proses Call akan memberikan keluaran berupa *command OK, busy, not found,* dan *sound call* kepada *user*

Proses terakhir pada DFD *level* 1 adalah Proses End Call. Proses End Call menerima *input* berupa *command end call* dari *user*. Selanjutnya *input* dari *user* akan diproses, dan akan menghasilkan *end call info* yang diberikan kepada *user*.



Gambar 3.3 DFD Level 2 Proses Login

Gambar 3.3 merupakan turunan dari Proses Login. Subproses dari Proses Login memiliki dua proses lainnya yaitu Proses Send Data, dan Proses Validate Data. Proses Send Data menerima *input username register*, *password*, IP *address server*, dan *command login.* Selanjutnya, proses akan melakukan *request* ke *database*, dan dari *database* data berupa *username register*, dan *password* akan dikirm ke Proses Validate Data. Kemudian, Proses Validate Data akan mengirim keluaran berupa informasi *login* berhasil atau *login* gagal.

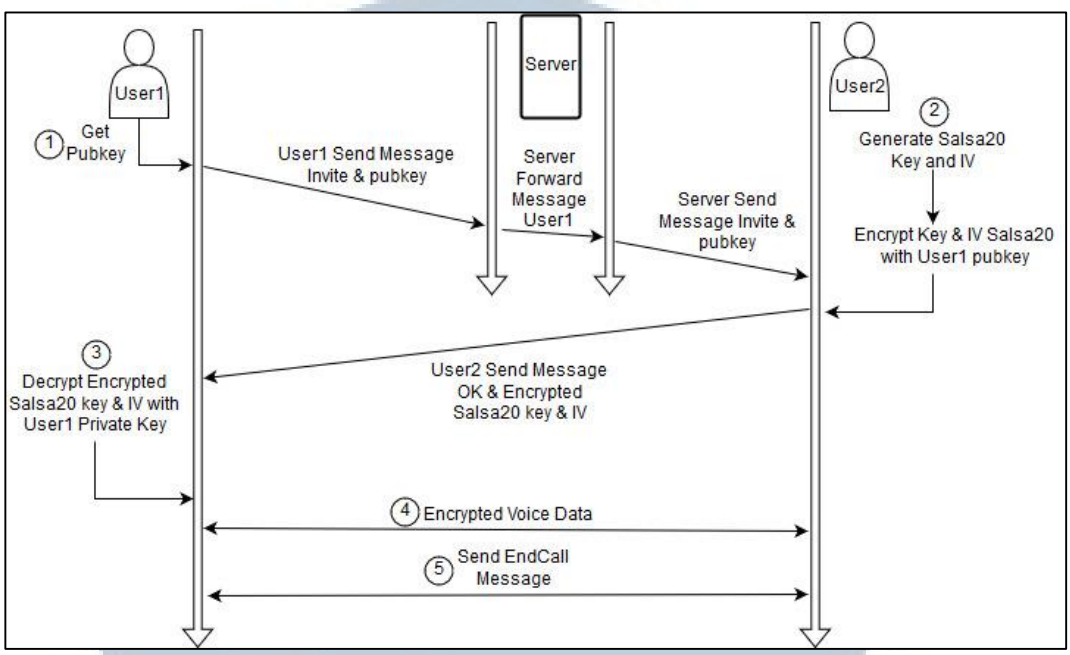


Gambar 3.4 DFD Level 2 Proses Call

Pada DFD level dua dari Proses Call terdapat lima proses yang terdiri dari, Proses Send Command, Proses Decrypt, Proses Encrypt, Proses Receive, dan Proses Send. Proses Send Command menerima masukan berupa *command call*, *username call*, dan IP *address server*. Kemudian proses akan meminta data ke *database* dan *database* akan memberikan data berupa IP *address call*.

Kumudian pada Proses Encrypt menerima masukan berupa *sound* dari *user*, kemudian data *sound* akan dikirimkan ke Proses Send, lalu dari Proses Send, data diteruskan ke Proses Receive, kemudian Proses Decrypt menerima data dan terakhir data akan dikirimkan kembali ke *user* berupa *output sound call.*

3.2.3 Flowchart



Gambar 3.5 Flowchart Alur Program

Pada Gambar 3.5 merupakan alur program jika pengguna melakukan panggilan.

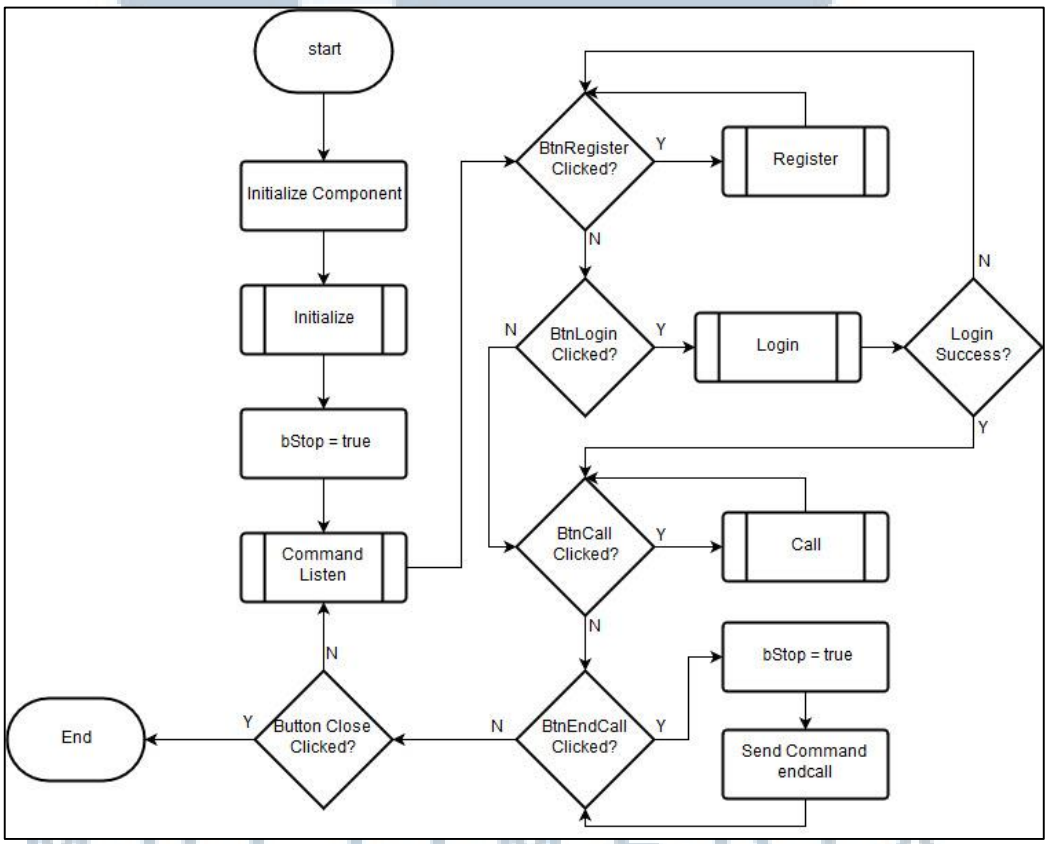
1. User1 mendapatkan PubKey yang merupakan *public key* dari user1. Kemudian PubKey dan Pesan Invite dikirimkan menuju *server* dan *server* akan melakukan *forward* ke User2.
2. User2 menerima pesan dari *server*, kemudian User2 akan membangkitkan kunci dan IV dari Salsa20. Selanjutnya kunci dan IV Salsa20 dienkripsi menggunakan *public key* dari User1 dan mengirimkan kunci dan IV yang telah dienkripsi beserta Pesan Ok kepada User1.
3. User1 menerima kunci, dan IV Salsa20 yang telah dienkripsi serta pesan dari User2. Kemudian User1 melakukan dekripsi kunci dan IV tersebut menggunakan *private key* dari User1.

4. Pada tahap ini kedua pengguna sudah dapat melakukan komunikasi dengan data suara yang telah dienkripsi menggunakan Salsa20.
5. Jika salah satu pengguna ingin mengakhiri panggilan, maka salah satu pengguna akan mengirimkan Pesan Endcall dan komunikasi berakhir.

Selanjutnya adalah perancangan aplikasi yang merupakan lanjutan dari perancangan sistem. Dalam subbab ini akan dijelaskan perancangan *flowchart* dari program *client* dan *server*.

A. Flowchart Aplikasi Client

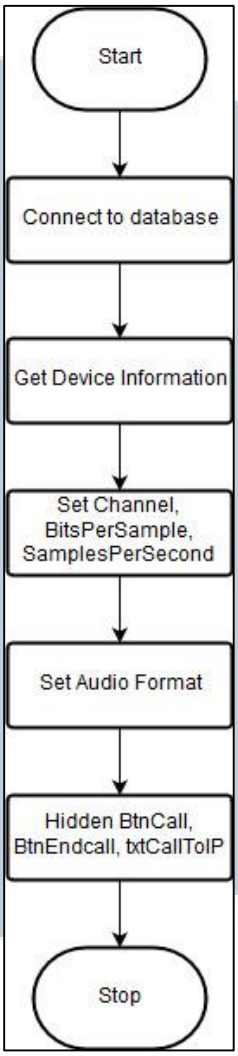
Pada subbab ini akan dibahas *Flowchart* Aplikasi yang digunakan *Client*. *Flowchart Main Client* akan menjelaskan bagaimana proses di dalam aplikasi.



Gambar 3.6 Flowchart Main Client

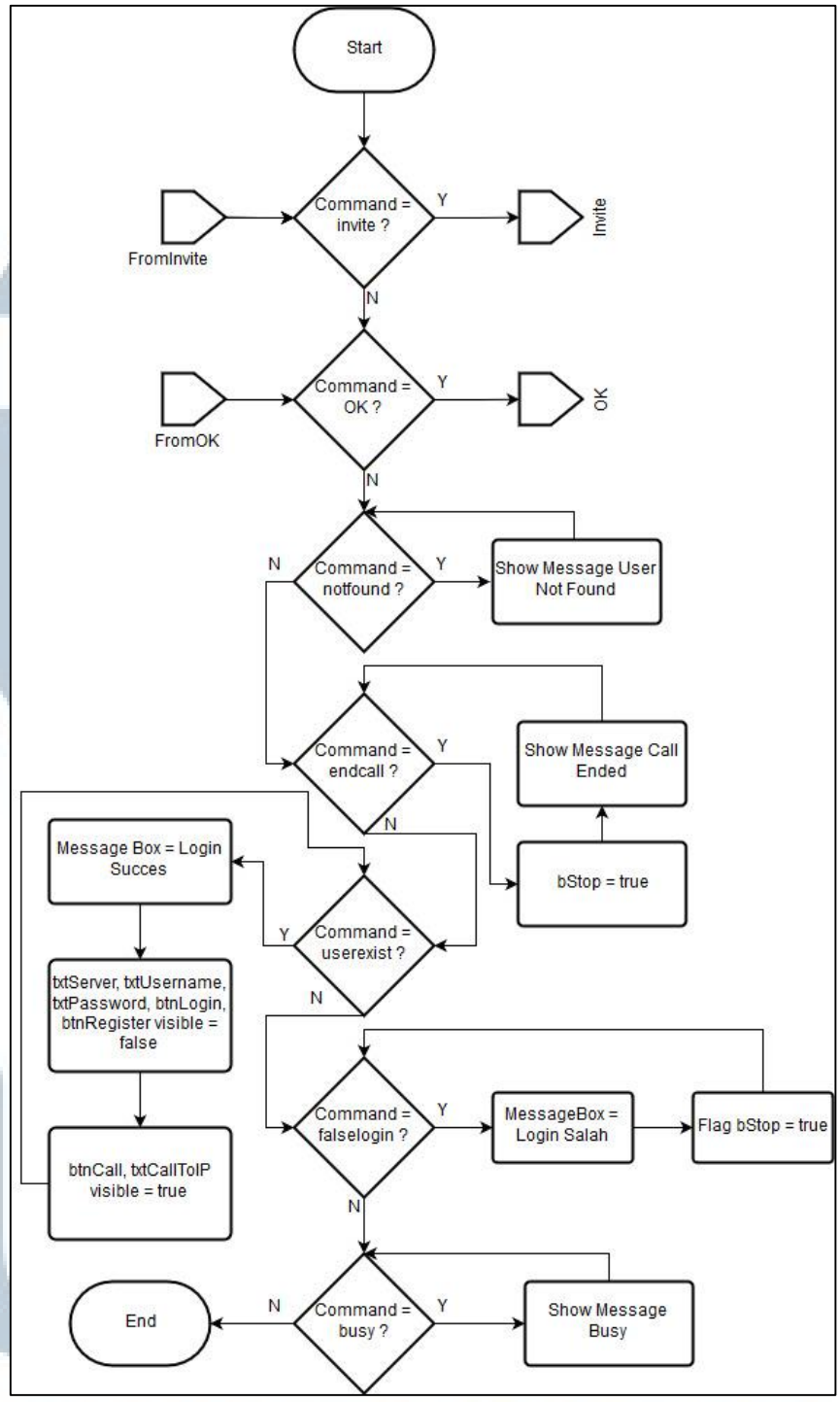
Gambar 3.6 merupakan *Flowchart Main Client*. Tahap awal program adalah Proses Initialize Component. Proses Initialize Component merupakan tahap awal program untuk membentuk tombol atau kotak teks yang terlihat oleh pengguna dan dapat berinteraksi langsung dengan pengguna. Selanjutnya terdapat Proses Initialize yang merupakan proses inisialisasi suara dan pengaturan kontrol visual. Pada inisialisasi suara terdiri dari inisialisasi *channel, bits per sample, samples per second*, dan format suara. Dalam kasus ini format suara yang ditentukan adalah wav *audio*. Pada pengaturan kontrol visual, saat awal dibuka aplikasi, terdapat beberapa kontrol visual yang disembunyikan agar tidak terjadi *bug*.

Selanjutnya adalah aplikasi akan selalu menjalankan Proses Command Listen. Aplikasi akan selalu mendengar *command* melalui *port* yang telah ditentukan. Di dalam *flowchart* juga terdapat simbol *decision* yang merupakan gambaran bahwa dalam aplikasi terdapat *button* yang memiliki fungsi tersendiri.



Gambar 3.7 Flowchart Initialize

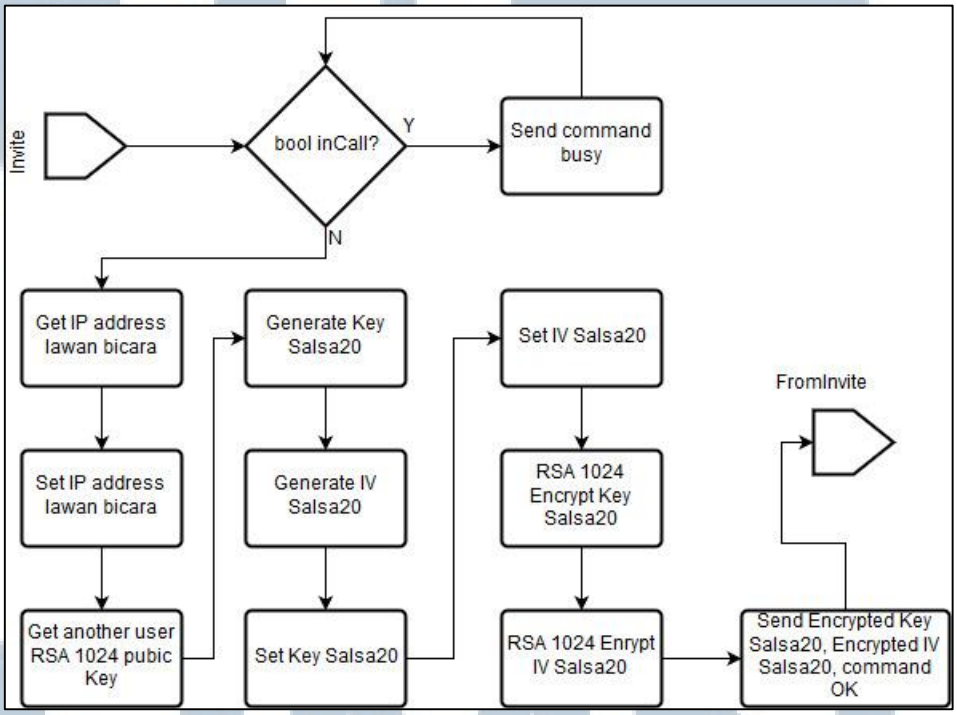
Gambar 3.7 merupakan *Flowchart* Initialize yang berfungsi sebagai inisialisasi awal program yang terdiri dari proses koneksi ke *database*, mendapatkan informasi *device*, mengatur *channel*, *bit per sample*, dan *samples per second*, proses mengatur *format audio*, dan menyembunyikan kontrol visual karena kontrol visual tersebut baru akan berfungsi setelah pengguna melakukan Proses Login.



Gambar 3.8 Flowchart Command Listen

Gambar 3.8 merupakan *Flowchart* Listen Command yang memiliki lanjutan berupa *Flowchart* Command OK dan *Flowchart* Command Invite. Pada *flowchart* ini aplikasi akan mendengarkan perintah yang dikirimkan baik melalui *server*

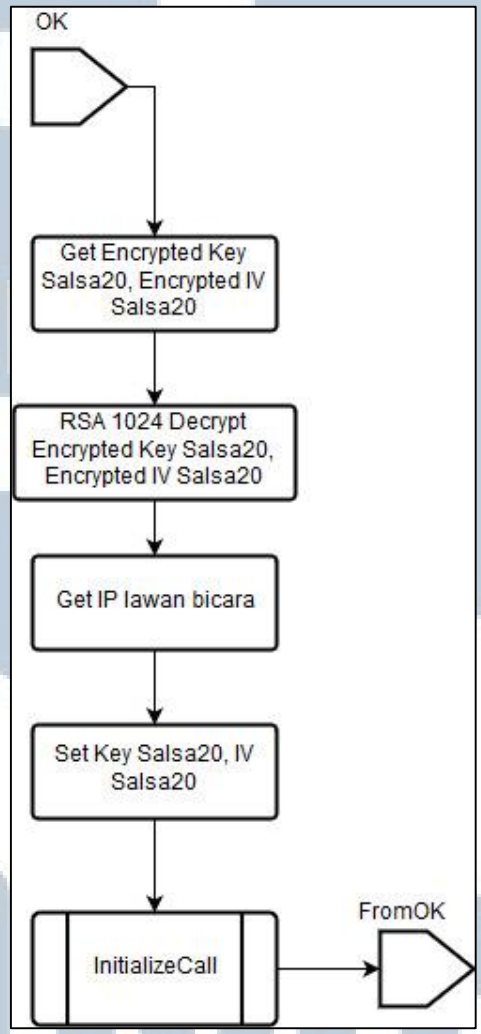
maupun *client.* Setiap *command* yang diterima maka program akan melakukan tugas yang berbeda-beda seperti contoh pada *flowchart* Gambar 3.8 ketika menerima *command false login* maka aplikasi akan memunculkan *message box* yang berbunyi bahwa *login* salah. Ketika ada *command userexist* maka program akan mengganti *user interface* untuk berada dalam posisi siap merima dan melakukan panggilan.



Gambar 3.9 Flowchart Command Invite

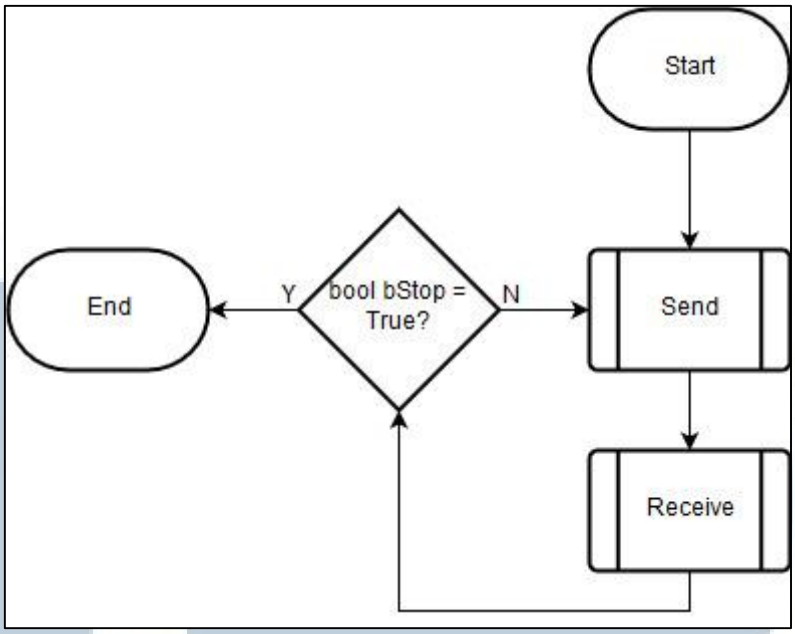
Gambar 3.9 merupakan lanjutan dari *Flowchart* Command Invite yang berada pada Gambar 3.8. *Flowchart* menjelaskan apa yang dikerjakan aplikasi jika menerima perintah *invite* dari pengguna lain. Pertama aplikasi akan melakukan *check* apakah dalam *state* sedang menerima panggilan atau tidak, kemudian aplikasi akan menerima IP *address* dari lawan bicara dan memasukannya ke sebuah variabel, selanjutnya aplikasi akan menerima kunci publik RSA 1024 dari lawan bicara. Sebelum mengirimkan kunci Salsa20, aplikasi akan membangkitkan kunci

Salsa20 dan IV Salsa20 yang kemudian dienkripsi menggunakan kunci publik RSA 1024 dari lawan bicara. Kemudian, kunci Salsa20 dan IV Salsa20 yang sudah terenkripsi akan dikirimkan beserta *command OK*.



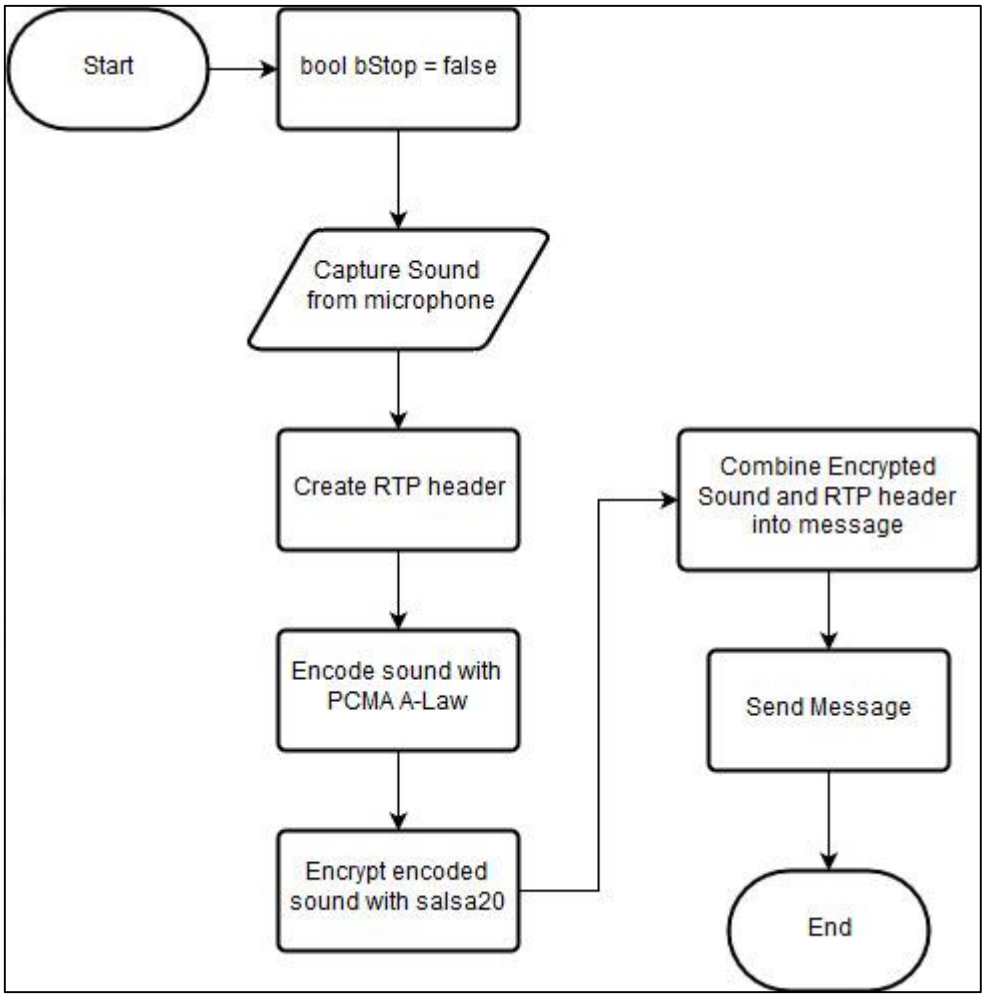
Gambar 3.10 Flowchart Command OK

Jika pesan yang diterima berupa *command OK*, maka aplikasi akan menerima kunci Salsa20 dan IV Salsa20 yang telah dienkripsi, kemudian kunci Salsa20 dan IV Salsa20 tersebut didekripsi menggunakan *private key* RSA 1024 dari aplikasi yang kemudian, hasil dekripsi kunci Salsa20 dan IV Salsa20 digunakan untuk berkomunikasi dan akan dilakukan Proses Inisialisasi Call.



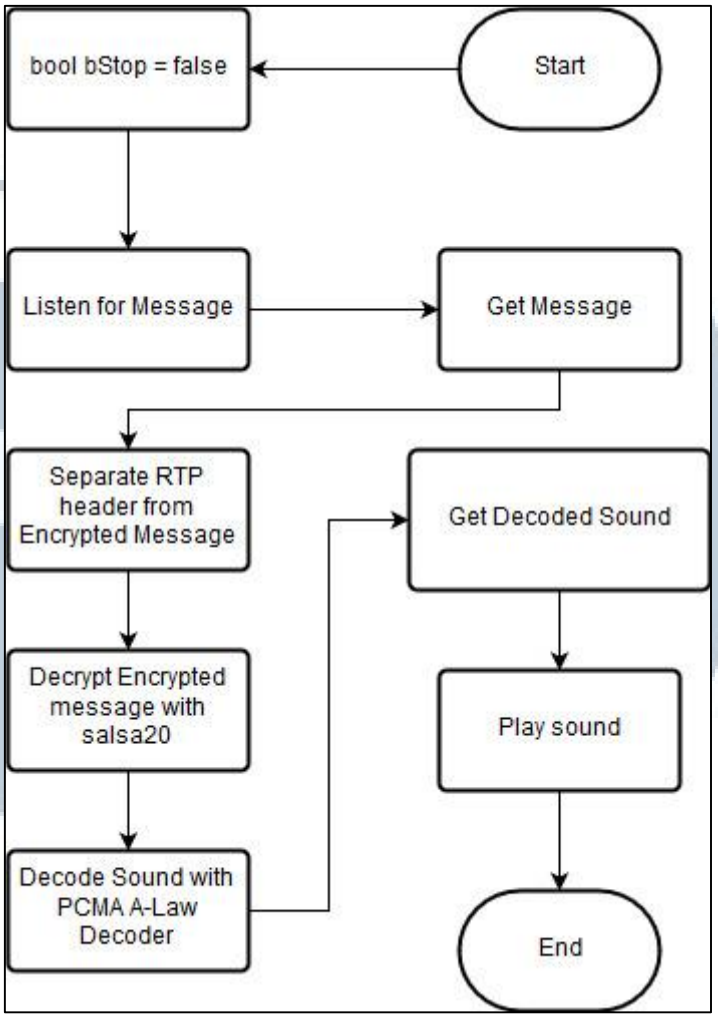
Gambar 3.11 Flowchart Initializecall

Gambar 3.11 merupakan *Flowchart* Initializecall. Dalam *flowchart,* aplikasi akan menjalankan Proses Send dan Proses Receive secara bersamaan dengan memulai Proses Send dan Proses Receive dengan *thread.* Proses Send dan Proses Receive akan dijelaskan pada Gambar 3.12 dan 3.13. Setelah Fungsi Send dan Fungsi Receive dijalankan, selanjutnya akan dilakukan penentuan *decision* apakah *state* dari varibel bStop *true* atau *false.* Jika *true* maka fungsi akan berakhir, jika *false* maka fungsi akan terus berjalan.



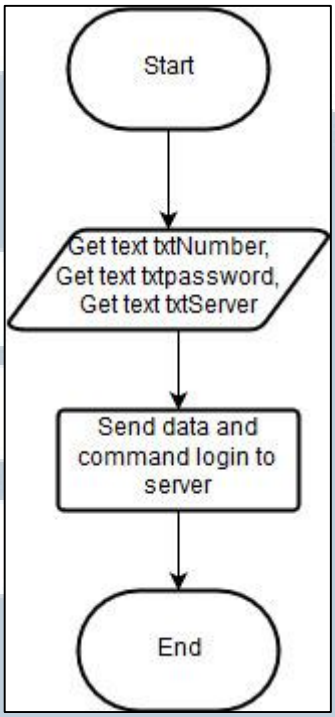
Gambar 3.12 Flowchart Send

Gambar 3.12 merupakan *flowchart* yang menjelaskan proses pengiriman data suara. Pertama proses akan melakukan *set* variabel bStop menjadi *false*, kemudian menerima masukan berupa suara yang didapat dari *microphone*, sebelum melakukan pengiriman data fungsi juga akan membuat RTP *header*. Suara yang bersumber dari *microphone* kemudian akan dilakukan *encode* dengan menggunakan A-Law *encoder*. Kemudian, hasil dari *encode* suara akan dienkripsi menggunakan Salsa20, dan terakhir RTP *header* akan digabungkan dengan pesan suara yang sudah dienkripsi menggunakan Salsa20, dan mengirimkannya melalui jaringan.



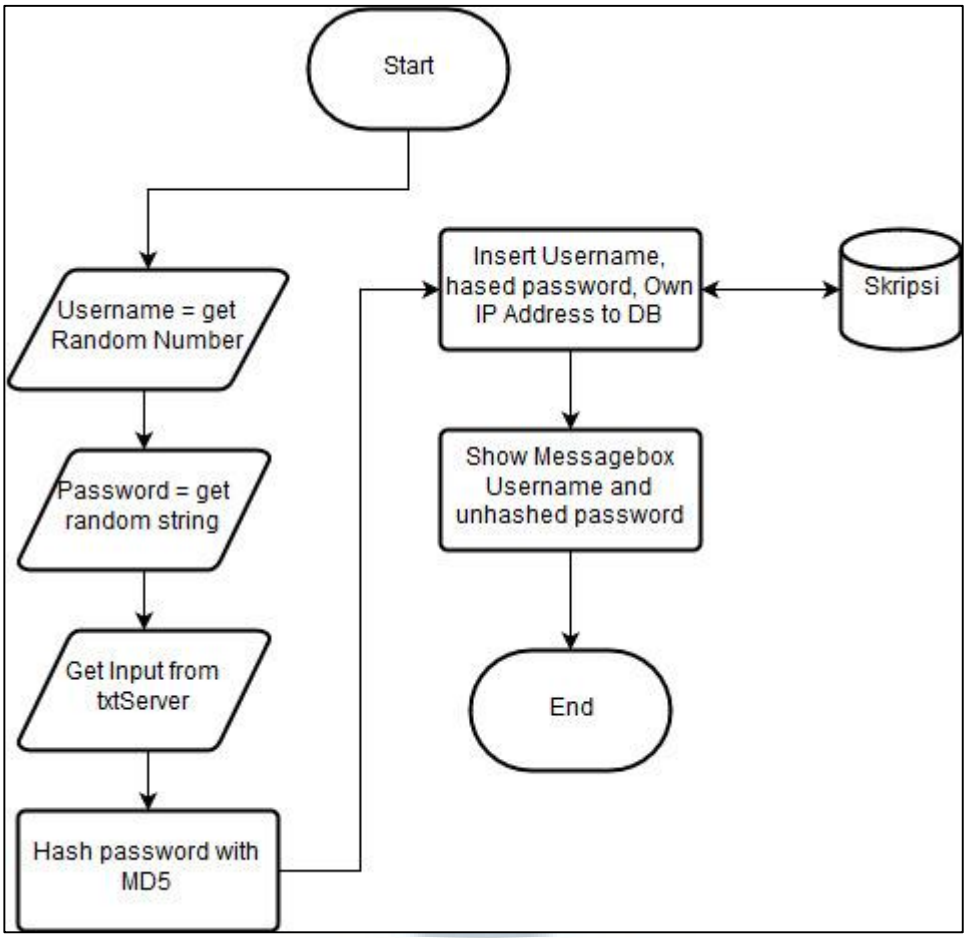
Gambar 3.13 Flowchart Receive

Agar dapat menerima pesan, Gambar 3.13 akan menjelaskan proses untuk menerima pesan. Proses pertama adalah dengan mengubah variabel bStop menjadi *false*. Proses Receive akan mendengarkan pesan yang dikirimkan melalui jaringan, dan mendapatkan pesan tersebut. Kemudian Fungsi Receive akan memisahkan badan pesan dengan RTP *header*. Agar suara dapat dimainkan maka pesan akan didekripsi terlebih dahulu, kemudian dilakukan *decode* dengan menggunakan A-Law *Decoder*. Pada tahap terakhir dari fungsi ini adalah memainkan suara yang diterima melalui pesan.



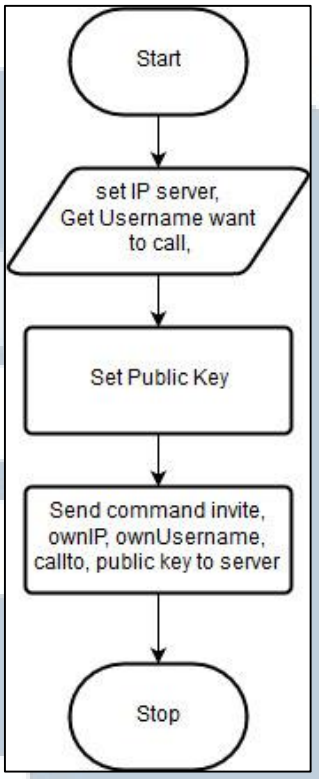
Gambar 3.14 Flowchart Login Aplikasi Client

Gambar 3.14 merupakan *Flowchart* Login yang berfungsi untuk melakukan *login* agar aplikasi dapat digunakan. Pertama dibutuhkan *input* berupa *number, password,* dan IP *address server*. Kemudian ketiga data tersebut dikirim ke *server* dan *server* yang akan mengkonfirmasi apakah informasi *login* sudah benar atau belum.



Gambar 3.15 Flowchart Register

Pada *Flowchart* Register, pertama aplikasi akan mendapatkan *number, password* yang dibangkitkan secara acak oleh aplikasi, dan IP *address server*. Kemudian *password* akan di-*hashing* menggunakan MD5. Setelah *password* berhasil di-*hash*, maka proses selanjutnya adalah mengirimkan data *number, password* yang telah di-*hash* , dan *own* IP *address* ke *database*. Terakhir, pengguna dapat melihat informasi *number* dan *password* dalam bentuk *message box*.

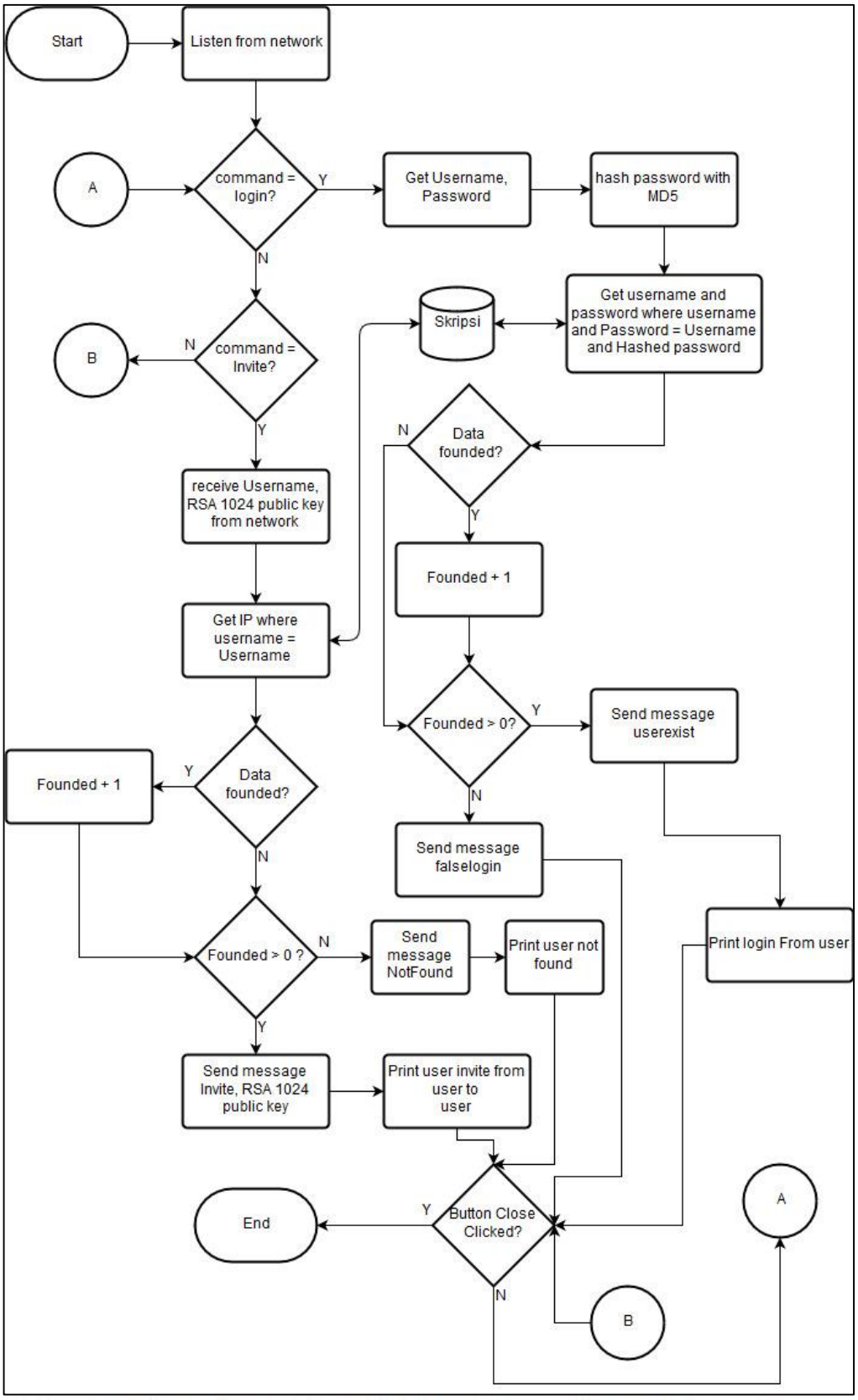


Gambar 3.16 Flowchart Call

Flowchart pada Gambar 3.16 merupakan *flowchart* saat tombol *call* ditekan. Pertama aplikasi akan menerima masukan berupa IP *address server*, dan *username* lawan bicara. Kemudian aplikasi akan membangkitkan *public key*, dan terakhir aplikasi akan mengirimkan data berupa *command invite, own* IP *address, own username, destination username* dan *public key* ke server.

B. Flowchart Aplikasi Server

Aplikasi *server* merupakan program terpisah yang bertujuan untuk perantara antara sesama pengguna. Tugas aplikasi *server* yaitu untuk melakukan verifikasi *login* pengguna, memberi informasi kepada sesama pengguna jika pengguna dalam kondisi sibuk, maupun pengguna tidak terdaftar. Dalam gambar 3.17 dijelaskan proses dari aplikasi *server*.



Gambar 3.17 Flowchart Command Listen Aplikasi Server

Gambar 3.17 merupakan *flowchart* pada Proses Command Listen Aplikasi Server. Pertama aplikasi akan *listening* pada *network*, kemudian aplikasi *server* akan melakukan *check* setiap *command* yang diterima. Jika *command* yang diterima adalah *login* maka aplikasi akan mendapatkan *username* dan *password* dari *client*, lalu aplikasi akan melakukan *request data* berdasarkan data *username* dan *password* yang diterima. Jika data tidak ditemukan maka, aplikasi tidak akan menambah nilai variabel *founded*, akan tetapi jika data ditemukan maka aplikasi akan menambahkan nilai dari variabel *founded* pada aplikasi, dan jika nilai *founded* melebihi 0 aplikasi akan mengirimkan *command* userexist ke *client*, dan mencetak informasi bahwa terdapat aktifitas *login* dari *client*, akan tetapi jika nilai *founded* tidak lebih dari 0 maka aplikasi akan langsung mengirim *command false login* ke *client*.

Jika *command* yang diterima adalah *command invite* maka, aplikasi akan menerima *username* dan RSA 1024 *public key* yang dikirimkan *client*. Kemudian program *server* akan meminta data IP *address* ke *database*. Jika data ditemukan maka program akan menambahkan 1 ke nilai variabel *founded*. Akan tetapi jika data tidak ditemukan maka proses akan langsung menuju proses *check* nilai variabel *founded*. Jika nilai *founded* melebihi 0, maka proses akan berlanjut ke pengiriman data *username* dan RSA 1024 *public key* ke *client* yang dituju dan mencetak informasi bahwa *client* satu menerima *invite* dari *client* yang lainnya. Akan tetapi jika nilai variabel *founded* sama dengan 0 maka, proses dilanjutkan dengan mengirimkan *command not found* ke *client* yang mengirim *command* invite dan proses akan mencetak informasi bahwa *user* tidak ditemukan. Proses terakhir adalah program akan melakukan *check* apakan tombol *close* ditekan atau tidak. Jika

ditekan maka program akan terhenti dan jika tidak maka program akan terus berjalan.

3.2.4 Struktur Tabel

Tabel 3.1 Struktur Tabel User

No.	Nama Kolom	Tipe Data	Keterangan
1	username	VARCHAR(100)	Untuk menampung ID / *number* dari pengguna
2	password	VARCHAR(50)	Untuk menampung *password dari pengguna*
3	ip	VARCHAR(20)	Untuk menampung alamat IP dari pengguna

Dalam subbab ini akan dijelaskan struktur tabel yang menyimpan informasi pengguna. Pada Tabel 3.1 terdapat tiga kolom yang masing – masing terdiri dari *username* dengan tipe data varchar dan panjang 100 yang berfungsi untuk menampung ID, *password* dengan tipe data varchar dan panjang 50 berfungsi untuk menampung *password* pengguna, dan terakhir *ip* dengan tipe data varchar dan panjang 20 berfungsi untuk menyimpan data IP *address* pengguna.