BAB III

ANALISIS DAN PERANCANGAN SISTEM

III.1. Analisis Masalah

Perancangan aplikasi pengamanan data bertujuan mengakses komputer *server* untuk mengirimkan file gambar pada komputer *client* dengan menggunakan metode atau algoritma *RC4*, implementasi komputer *server* pada komputer *client* memudahkan pengguna dalam melakukan banyak memperoleh informasi. Pada perancangan ini sistem yang dibangun menggunakan perangkat *wifi* sebagai penghubung koneksi jaringan yang digunakan dan mengimplementasikan sistem dari transfer data file dengan algoritma *RC4*. Salah satu fungsi dari sistem jaringan komputer yang banyak digunakan adalah penerapan *file* keamanan transfer , dimana dengan penerapan *file* yang transfer ini setiap pengguna dapat saling mengirim data.

Perihal menyangkut pertukaran file menggunakan jaringan haruslah membutuhkan keamanan terhadap data yang akan dikirim. Hingga saat ini belum banyak pihak yang menerapkan konsep pengamanan tambahan sehingga file maupun data yang di bagikan dapat dengan mudah dipergunakan orang lain untuk kepentingan yang merugikan pihak tertentu.

RC4 merupakan enkripsi *stream simetrik proprietary* yang dibuat oleh *RSA Data Security Inc* (RSADSI). Penyebarannya diawali dari sebuah source code yang diyakini sebagai RC4 dan dipublikasikan secara '*anonymously*' pada tahun 1994. Algoritma yang dipublikasikan ini sangat identik dengan

implementasi RC4 pada produk resmi. RC4 digunakan secara luas pada beberapa aplikasi dan umumnya dinyatakan sangat aman. Sampai saat ini diketahui tidak ada yang dapat memecahkan/membongkarnya, hanya saja versi ekspor 40 bitnya dapat dibongkar dengan cara "*brute force*" (mencoba semua kunci yang mungkin). RC4 tidak dipatenkan oleh RSADSI, hanya saja tidak diperdagangkan secara bebas (*trade secret*).

Algoritma RC4 cukup mudah untuk dijelaskan. RC4 mempunyai sebuah *S-Box*, $S_0,S_1,...,S_{255}$, yang berisi permutasi dari bilangan 0 sampai 255, dan permutasi merupakan fungsi dari kunci dengan panjang yang variabel. Terdapat dua indeks yaitu i dan j, yang diinisialisasi dengan bilangan nol. Untuk menghasilkan random byte langkahnya adalah sebagai berikut :

$i = (i + 1) \bmod 256$

$j = (j + S_i) \bmod 256$

swap S_i dan S_j

$t = (S_i + S_j) \bmod 256$

$K = S_t$

Byte K di XOR dengan *plaintexs* untuk menghasilkan *cipherteks* atau di XOR dengan cipherteks untuk menghasilkan plainteks. Enkripsi sangat cepat kurang lebih 10 kali lebih cepat dari DES.

Inisialisasi S-Box juga sangat mudah. Pertama isi secara berurutan $S_0 = 0$, $S_1 = 1,...,S_{255} = 255$. Kemudian isi array 256 byte lainnya dengan kunci yang diulangi sampai seluruh array K_0, $K_1,...,K_{255}$ terisi seluruhnya. Set indeks j dengan nol, Kemudian lakukan langkah berikut :

for i = 0 to 255

$j = (j + S_i + Ki) \bmod 256$

swap S_i dan S_j

Salah satu kelemahan dari RC4 adalah terlalu tingginya kemungkinan terjadi tabel S-box yang sama, hal ini terjadi karena kunci user diulang-ulang untuk mengisi 256 bytes, sehingga 'aaaa' dan 'aaaaa' akan menghasilkan permutasi yang sama. Untuk mengatasi ini maka pada implementasinya nanti kita menggunakan hasil hash 160 bit SHA dari password kita untuk mencegah hal ini terjadi. Kekurangan lainnya ialah karena enkripsi RC4 adalah XOR antara data bytes dan *pseudo-random byte stream* yang dihasilkan dari kunci, maka penyerang akan mungkin untuk menentukan beberapa byte pesan orisinal dengan meng-XOR dua *set cipher byte*, bila beberapa dari pesan input diketahui (atau mudah untuk ditebak). Untuk mengatasinya pada aplikasinya kita menggunakan initialization vector (IV) yang berbeda-beda untuk setiap data, sehingga bahkan untuk file yang sama akan dihasilkan *ciphertext* yang berbeda.

Proses enkripsi deskripsi mempunyai proses yang sama sehingga hanya ada satu fungsi yang dijalankan untuk menjalankan kedua proses tersebut. RC4

mempunyai sebuah S-Box dan key dalam bentuk array 256 byte yaitu : S0, S1, ..., S255 yang berisi permutasi dari bilangan 0 sampai 255, K0, K1, …, K255 dengan kategori stream simetrik. Sedangkan untuk inisialisasi S-Box yaitu dengan mengisikan nilai 1 sampai dengan 255 dimulai dari S0 sampai dengan S255, isi S-Box secara berurutan, yaitu S0=0, S1=1, ... , S255=255 (Scheiner, 2001). Untuk inisialisasi key yaitu dengan mengisikan array K255 byte dengan kunci yang diulangi sampai seluruh array K0,K1, ... K255 terisi seluruhnya. Pseudocode yang terbentuk untuk menciptakan initialisasi key adalah sebagai berikut.

For I = 0 to 255

Ki = I mod length (key)

Pseudorandom adalah nilai yang dibangkitkan dari nilai S-Box dan Key yang telah diinisialisasi. Caranya set indeks j dengan nol, dan melakukan penukaran nilai S-Box yang sudah diinisialisasi sebelumnya dengan nilai perulangan ditambah dengan S-Box awal ditambah dengan nilai dari array K yang dapat digambarkan dan dijelaskan dalam Pseudocode sebagai berikut.

j = 0

for i = 0 to 255

j = (j + Si + Ki) mod 256

swap Si dan Sj

Fungsi *swap* merupakan fungsi yang menukarkan nilai S ke-i dengan nilai S ke-j. Kemudian membangkitkan nilai pseudorandom key berdasarkan indeks dan nilai S-Box. Terdapat 2 indeks yaitu i dan j, yang diinisialisasi dengan

bilangan nol. Untuk menghasilkan random byte langkahnya adalah sebagai berikut.

i = 0

j = 0

for idx = 0 to len-1

i = (i + 1) mod 256

j = (j + Si) mod 256

swap Si dan Sj

t = (Si + Sj) mod 256

k = St

buffidx = k XOR buffidx

Adapun keterangan dari rumus diatas sebagai berikut :

1. buff merupakan pesan yang akan dienkripsi atau dekripsi
2. len merupakan panjang dari buff

Nilai pseudorandom key inilah yang akan di XOR dengan plainteks untuk menghasilkan cipherteks atau XOR dengan cihperteks untuk menghasilkan plainteks. Untuk menghasilkancipherteks yaitu dengan rumus ”Cipherteks = Plainteks XOR K” sedangkan untuk menghasilkan plainteks yaitu dengan rumus “Plainteks = Cipherteks XOR K”.

Contoh lain dari implementasi algoritma RC4 adalah saat proses *key scheduling algorithm* didapatkan S-Box terakhir 54, 157, 62, 162, 25, 135, 195, 103, 208, 8, 188, 42, 165, 13, 141, 253, 35, 231, 108, 134, 93, 82, 49, 9, 83, 139, 147, 38, 87, 193, 22, 219, 113, 248, 155, 117, 64, 123, 154, 67, 53, 94, 46, 102,

133, 170, 106, 194, 24, 246, dan 45 maka penyelesaiannya terdapat di dalam tabel berikut :

Tabel III.1 Pengimplementasian Algoritma RC4

Iterasi	Pesan	Ascii	i=(i+1)mod 256	j=(j+SBox[i]) mod 256	Pertukaran	SBox[(Sbox[i]+ Sbox[j])mod 256]	Pesan ^ K	Ascii
			i	j				
			Nilai awal 0	Nilai awal 0		K	Hasil	
0	i	105	(0+1)mod 256= 1	(0+157)mod 256=157	sbox ke-1 dengan Sbox ke - 157	SBOX[144]+157 mod 256] = SBOX[301%256] = SBOX[45] = 170	105 ^ 170=195	Ã
1	b	98	(1+1) mod 256 = 2	(157+62) mod 256 = 219	sbox ke-2 dengan sbox ke-219	SBOX[15+62 mod 256]=SBOX[77]=236	98 ^ 236 = 142	ž
2	u	117	(2+1) mod 256 = 3	(219+162) mod 256 = 125	sbox ke-3 dengan sbox ke-125	SBOX[47+162 mod 256]=SBOX[209]=158	117 ^ 158 = 235	ë

Desain dan implementasi rancangan meliputi desain interface, gambaran proses sistem, implementasi desain, dan semua yang diperlukan dalam aplikasi yang akan dibangun.

Algoritma RC4 memiliki dua fase yatitu setup kunci dan pengenkripsian. Setup untuk kunci adalah fase pertama dan yang paling sulit dalam algoritma ini. Dalam setup N-bit kunci (N merupakan panjang dari kunci). Kunci enkripsi digunakan untuk menghasilkan variabel enkripsi yang menggunakan dua buah array yaitu array state dan array kunci, dan sejumlah-N hasil dari operasi penggabungan.

Operasi penggabungan ini terdiri dari (untuk meregenerate keystream, keystream = Pembangkit aliran-bit-kunci) yaitu pemindahan (swapping) byte, operasi modulo, dan rumus lainnya. Dalam pemindahan (swapping) byte, operasi modulo, adalah bagian dari KSA (key-scheduling algorithm) yang berisi secret

internal state. KSA ini nanti nya akan menghasilkan (A permutation of all 256 possible bytes (denoted “S” below)) permutasi dari 256 kemungkinan bytes yang ada pada array S, permutasi diinisialisasi dengan panjang kunci dan melakukan swaping kemudian “rumus lain” yang dimaksud adalah *Pseudo-Random Generation Algorithm* (PRGA) sesuai nama nya PRGA ini digunakan untuk mendapatkan *byte* acak untuk enkripsi.

Contoh cara kerja algoritma RC4 dengan menggunakan 4 (empat) bit kunci Buat array state Si berukuran 4 byte, yang memiliki nilai 0 sampai dengan 3

Si = 0 1 2 3

S0 = 0

S1 = 1

S2 = 2

S3 = 3

Array state Si ini digunakan untuk menghasilkan variabel enkripsi, kunci enkripsi. Buat array kunci Ki berukuran 4 byte, yang memiliki nilai pengulangan dari kunci untuk memuat keseluruhan isi array.

Ki = 1 7 1 7

K0 = 1

K1 = 7

K2 = 1

K3 = 7

Array kunci ki ini digunakan untuk menghasilkan variabel enkripsi. Kunci enkripsi 2 array ini Si dan Ki digunakan untuk menghasilkan variabel

enkripsi langkah selanjut nya adalah menggabungkan kedua array. Tetapi dalam pemrograman akan menggunakan perulangan, sehinggga butuh variabel yang digunakan untuk meng-index array Si dan Ki, sehingga dimungkinkan penggabungan array dengan perulangan. Variabel i dan f, kemudian inisialisasi keduanya dengan nilai nol berikut rumus iterasi ny…(f + Si + Ki) mod 4.

(f + Si + Ki) mod 4

f = var_index

Si = array_state

Ki = array_kunci

berikut ini langkah iterasi nya :

iterasi kesatu

for i = 0 (0 + 0 + 1) mod 4 = 1 = f

f = 0

S0 = 0

K0 = 1

Swap S0 dengan S1

Si = 1 0 2 3

S0 = 1

S1 = 0

S2 = 2

S3 = 3

Dilakukan swap S0 dengan S1, karena pada iterasi ke satu hasil dr operasi modulus didapat hasil = 1 (S1) dan ini adalah perulangan ke satu, i = 0

(dalam indeks array adalah nol, sehingga yang ditukar adalah S0) sehingga tukar isi array S0 dengan S1.

Iterasi kedua

for i = 1(1+0+7) mod 4 = 0 = f

f = nilai_var_index = 1

S1 = nilai_array_state1 = 0

K1 = nilai_array_kunci1 = 7

- perhatikan ini adalah perulangan/iterasi ke dua, dalam array adalah indeks ke 1sehingga i = 1.
- perhatikan juga nilai f adalah 1 (f = 1) nilai ini didapat dari hasil operasi modulus sebelumnya.
- perhatikan juga nilai dari indeks array ke satu (S1) pada mula2 dia mempunyai nilai 1 tetapi setelah dilakukan swap pada iterasi kesatu nilainy telah tertukar dengan S0, sehingga nilai S1 yang dipakai adalah 0, S1 = 0.

Swap S1 dengan S0

Si = 0 1 2 3

S0 = 0

S1 = 1

S2 = 2

S3 = 3

Swap yang dilakukan adalah array S1 dengan array S0, karena ini perulangan ke 2 (dalam array adalah indeks ke 1, sehingga i= 1, S1) hasil modulus

adalah 0, sehingga terpilih array yang akan di swap adalah S0 jadi hasil akhirnya, swap S1 dengan S0.

Iterasi ketiga

for i = 2 (0 + 2 + 1) mod 4 = 3 = f

f = nilai_var_index =0

S2 = nilai_array_state1 = 0

K2 = nilai_array_kunci1 = 1

Swap S2 dengan S3

Si = 0 1 3 2

S0 = 0

S1 = 1

S2 = 3

S3 = 2

Dalam swap S2 dengan S3 dilakukan untuk mengikuti lagi iterasi kesatu dan iterasi kedua.

Iterasi keempat

for i = 3 (3 + 0 + 7) mod 4 = 2= f

f = nilai_var_index =3

S3 = nilai_array_state1 = 0

K3 = nilai_array_kunci1 = 7

Swap S3 dengan S2

Si = 0 1 2 3

S0 = 0

S1 = 1

S2 = 2

S3 = 3

Iterasi berhenti pada i = 3, karena pada array nilai dimulai dengan nilai 0. Tentukan nilai byte acak untuk enkripsi (PRGA). Pertama-tama Inisialisasi ulang i dan f menjadi 0, set i menjadi (i + 1) mod 4 dan set f menjadi (f + Si) mod 4. Lalu swap Si dan Sf.

i = (i + 1) mod 4

f = (f + Si) mod 4

Kemudian Set t menjadi (Si + Sf) mod 4, nilai acak untuk enkripsi adalah St.

t = (Si + Sf) mod 4(0 + 1) mod 4 = 1 = i

i bernilai nol, karena inisialisai dengan nilai nol.

(0 + 2) mod 4 = 2 = f

f = 0

Si = 2

Si bernilai 2

Iterasi keempat

for i = 3 (3 + 0 + 7) mod 4 = 2= f

f = nilai_var_index =3

S3 = nilai_array_state1 = 0

K3 = nilai_array_kunci1 = 7

Swap S3 dengan S2

Si = 0 1 2 3

S0 = 0

S1 = 1

S2 = 2

S3 = 3

Dari swap terakhir adalah S3 dengan S2, sehingga kemungkinan nilai Si diambil dari nilai S2 yaitu 2. Karena i = 1, dan f = 2, didapat Si =S1 dan Sf = S2, jadi swap S1 dengan S2.

Swap S1 dengan S2

Si = 1 0 2 3

S0 = 1

S1 = 0

S2 = 2

S3 = 3

Kemudian cari t, nilai acak enkripsi yang akan menjadi kunci untuk menghasilkan ciphertext.

t = (0 + 2) mod 4 = 2= f

S1 = 0

S2 = 2

St = 2

Hasil akhir menemukan nilai dari St akan dipakai untuk menXOR kan plainteks.

St = 2, (nilai dr 2 biner adalah 00000010), variabel enkripsi ini lalu di XOR-kan

dengan plaintext untuk menghasilkan ciphertext. Sebagai contoh akan digunakan pesan “HI”.

H I

0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1

0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 XOR

0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1

Setelah penerima mendapatkan pesan tersebut, penerima harus XOR pesan yang encrypted dengan kunci decrypt-nya untuk mendekripsi pesan tersebut.

0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 1

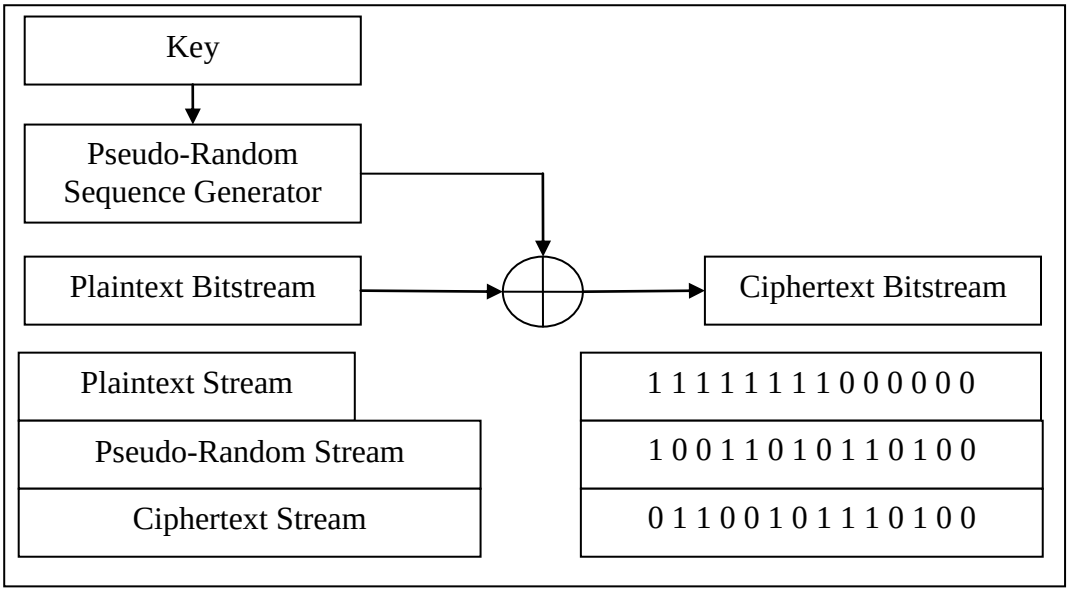
0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 XOR

0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1

H I

Dapat dengan jelas untuk mengenkripsi data yang akan dikirimkan dan hasil yang diterima oleh penerima.

Berikut ini merupakan blok diagram algoritma RC4 yang sering terjadi. Adapun gambar tersebut dapat dilihat pada gambar III.1 berikut.



Gambar III.1. Blok Diagram Algoritma RC4 Secara Umum

III. 1.1. Spesifikasi *Hardware* dan *Software*

Dalam perancangan aplikasi transfer file ini terdapat beberapa perangkat yang penulis gunakan agar aplikasi berjalan sebagaimana mestinya, yaitu sebagai berikut :

1. Perangkat Lunak (*Software*)
    a. *Operating System*, OS yang digunakan dalam perancangan dan tes untuk program aplikasi yang dirancang adalah *Windows 7*.
    b. Visual Studio 2010, sebagai bahasa pemograman yang digunakan
2. Perangkat Keras (*Hardware*)
    a. Komputer yang setara dengan *Core I 3*.
    b. *Mouse*, *keyboard*, dan *Monitor*.

III. 2. Strategi Pemecahan Masalah

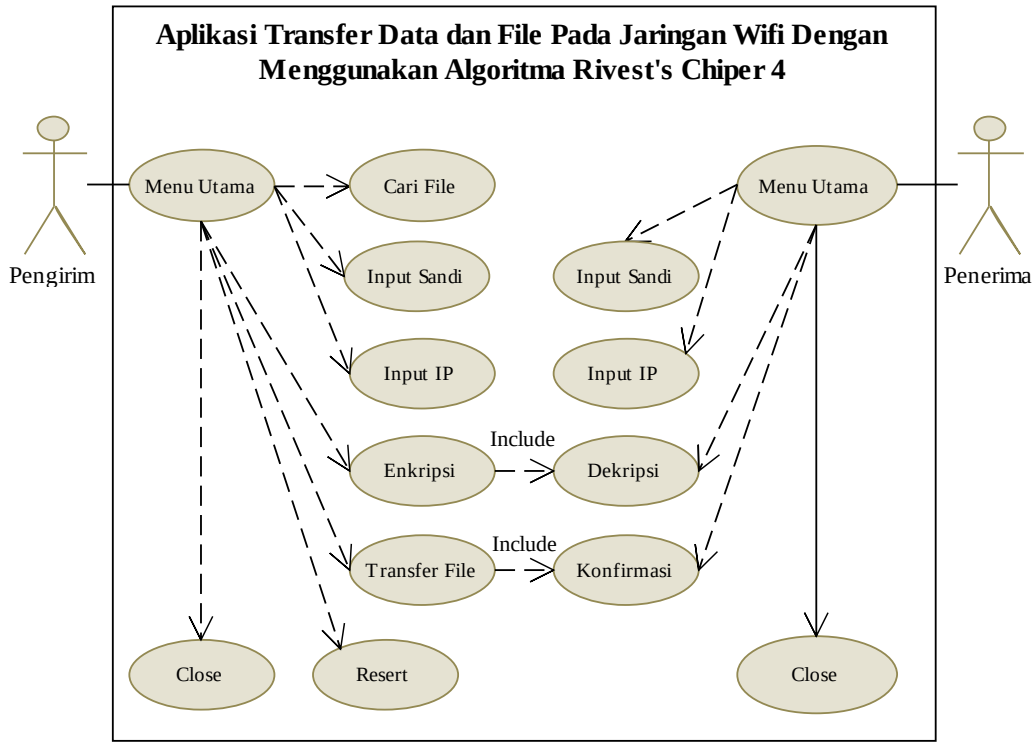
Adapun strategi pemecahan masalah terhadap pemasalahan yang ada adalah dengan melakukan penerapan konsep terhadap file yang akan di transfer. Dalam hal ini penulis akan merancang sebuah aplikasi enkripsi dan enkripsi file dengan menggunakan algoritma *RC4*, dan pengguna dapat menginputkan ataupun menyimpan file yang akan di transfer. Kemudian untuk proses transfer menggunakan koneksi *wifi* agar komputer client dan server dapat terhubung.

III.3. Perancangan Sistem

Pada perancangan sistem ini akan menjelaskan mengenai sistem yang berjalan dan tampilan rancangan aplikasi transfer file berbasis client-server yang akan dibangun.

III.3.1. *Use Case Diagram*

Pada perancangan sistem ini akan menjelaskan mengenai algoritma rancangan dan tampilan rancangan aplikasi *remote desktop* komputer yang akan dibangun. *Use case* diagram berfungsi untuk menggambarkan kegiatan aktor atau pengguna aplikasi. Adapun *use case* diagram aplikasi yang dirancang dapat dilihat pada gambar III.2 berikut.



Gambar III.2. *Use Case* Diagram

Dari gambar diatas dijelaskan tentang *server* berfungsi sebagai pengeksekusi perintah yang mengirim file gambar dengan mengenkripsi file tersebut, perintah yang diterima dari komputer *client* yang kemudian didekripsi file menjadi gambar yang aslinya.

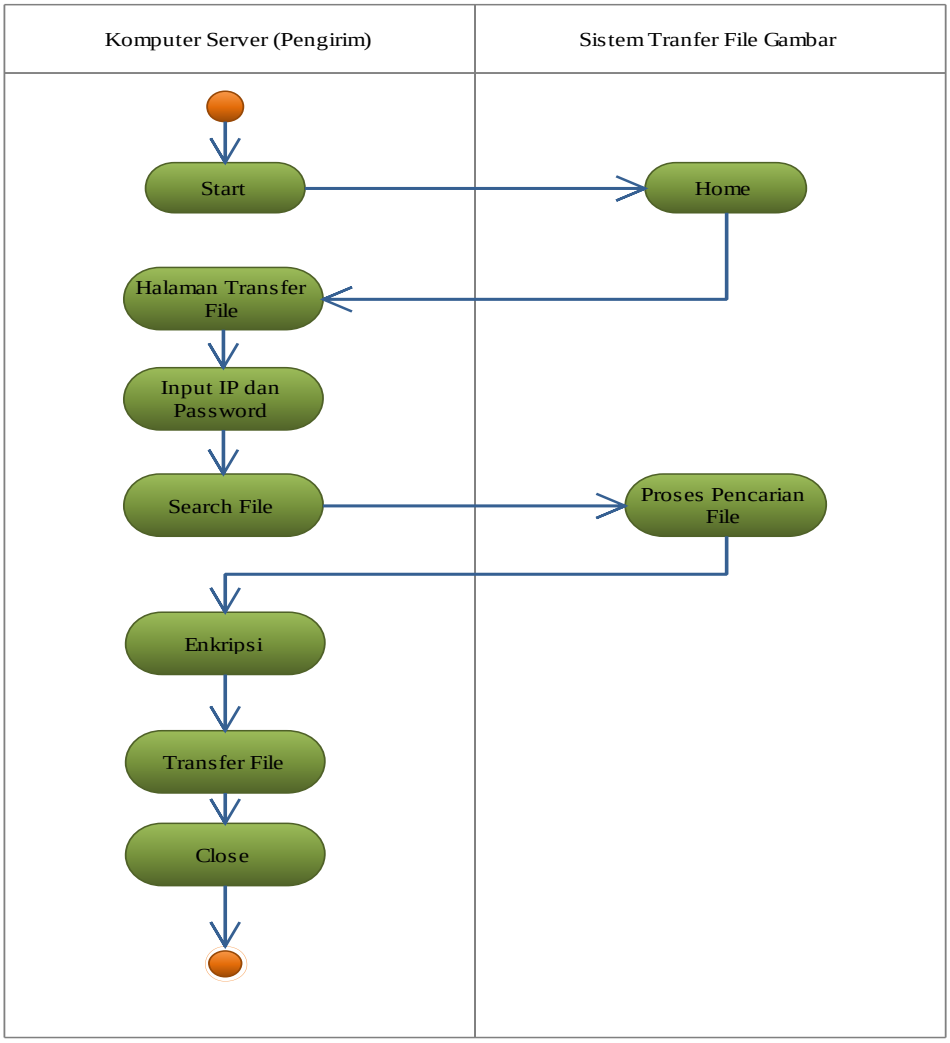
III.3.2 *Activity* Diagram

Activity diagram adalah teknik untuk mendiskusikan logika *prosedural*, Berikut ini adalah *activity* diagram aplikasi kompresi yang dirancang.

1. *Activity* Diagram Aplikasi Komputer *Server*

Aktivitas pengguna pada sisi komputer server (pengirim) terlebih dahulu mencari perangkat *wifi* yang aktif pada area jangkauan, begitu juga pada computer client (penerima). Setelah ditemukan, Pengguna dapat menghubungkan setelah

terhubung pengguna dapat memilih menginputkan IP dan sandi sama hal yang dilakukan oleh *client*, dapat dilihat pada gambar III.3 berikut.

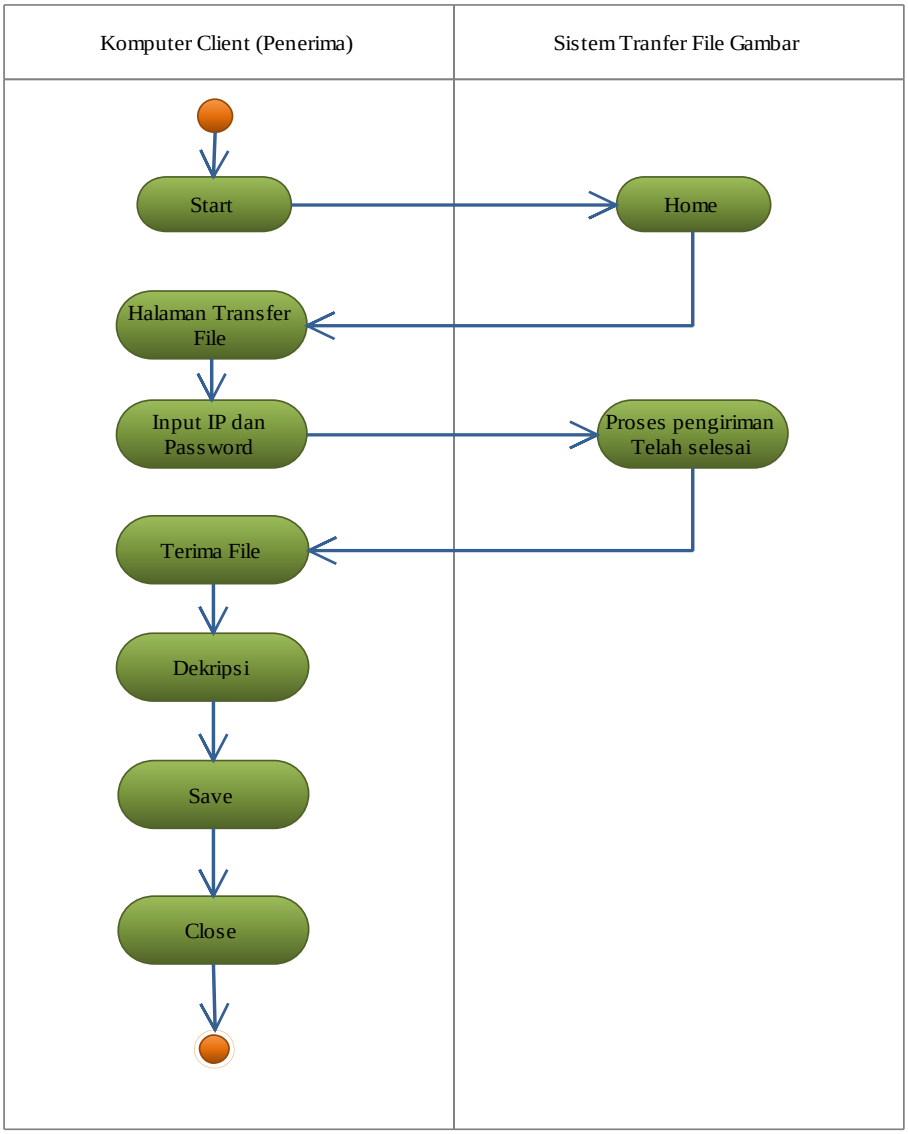


Gambar III.3. *Activity* Diagram Aplikasi Komputer *Server*

2. *Aktivity* Diagram Aplikasi Komputer *Client*

Sedangkan pada komputer penerima ataupun client untuk dapat menerima file yang telah ditransfer maka terlebih dahulu menghubungkan pada jaringan wifi, dan kemudian menginputkan IP dan Sandi pada komputer *server*. Lalu kemudian pengguna dapat mendekripsikan file tersebut dengan hasil file gambar yang sebenarnya dan kemudian dapat disimpan sesuai keinginan sipenerima.

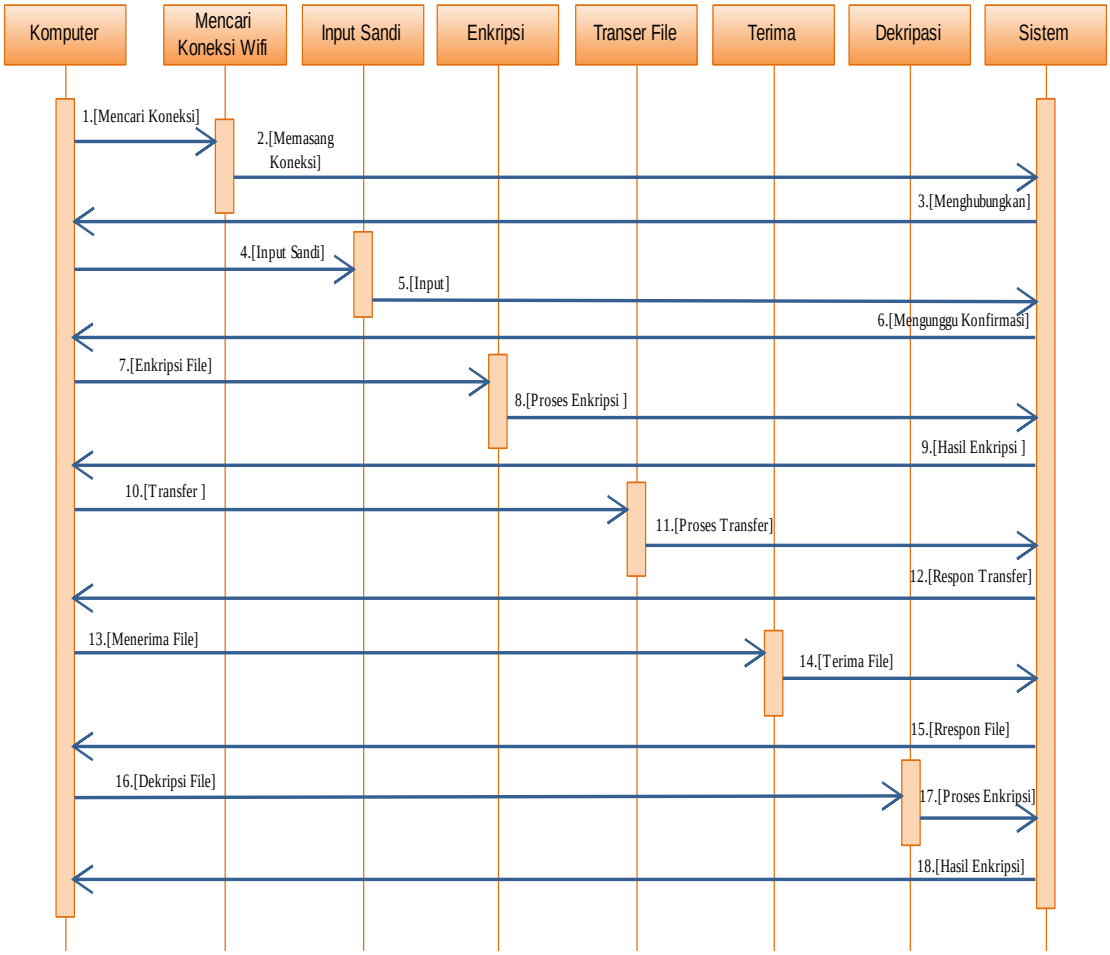
Penjelasan tersebut dapat digambarkan pada activity diagram pada gambar III.4 berikut:



Gambar III.4. *Activity* Diagram Aplikasi Komputer *Client*

III.3.3 *Squence* Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario proses penggunaan aplikasi. Berikut ini adalah *Sequence* diagram aplikasi yang dirancang yang dapat dilihat pada gambar III.5. berikut :

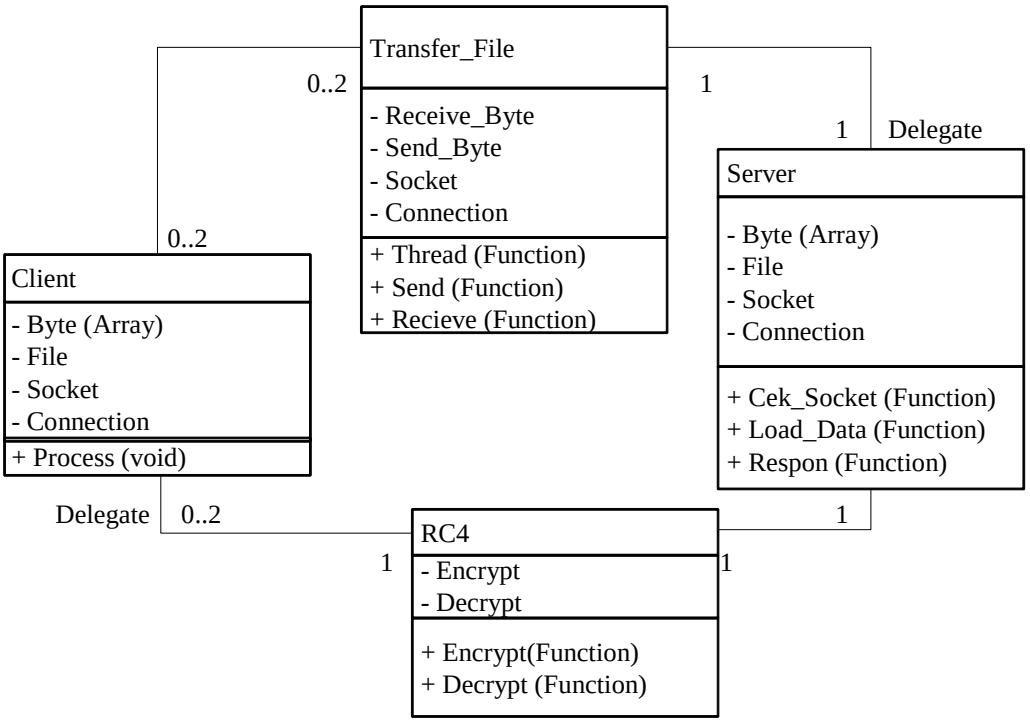


Gambar III.5. *Squence* Diagram

III.3.4 *Class* Diagram

Class diagram digunakan untuk menampilkan kelas-kelas dan paket-paket di dalam sistem. *Class* diagram memberikan gambaran system secara statis dan relasi antar mereka. Biasanya, dibuat beberapa class diagram untuk sistem

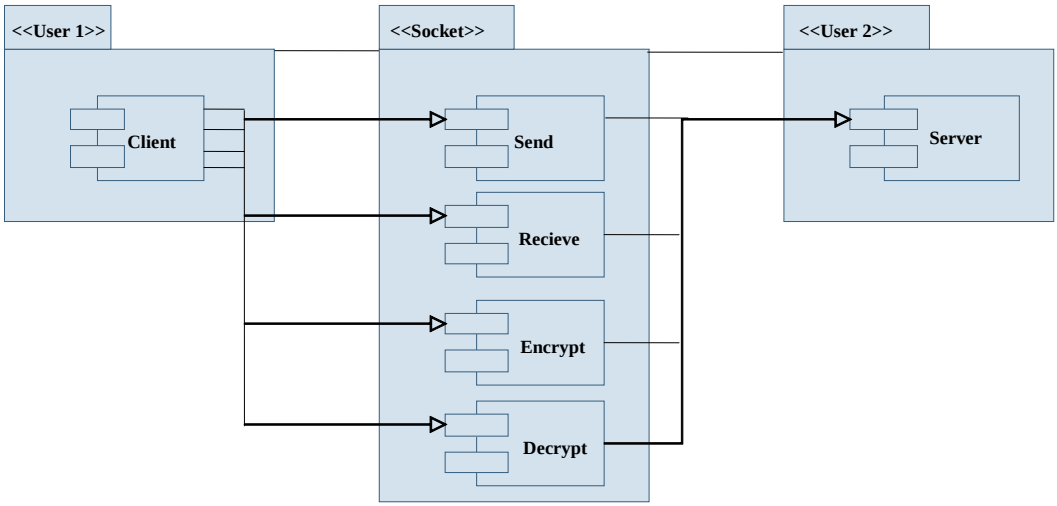
tunggal. Beberapa diagram akan menampilkan subset dari kelas-kelas dan relasinya. Dapat dibuat beberapa diagram sesuai dengan yang diinginkan untuk mendapatkan gambaran lengkap terhadap system yang dibangun.



Gambar III.6. *Class* Diagram

III.3.5. *Component* Diagram

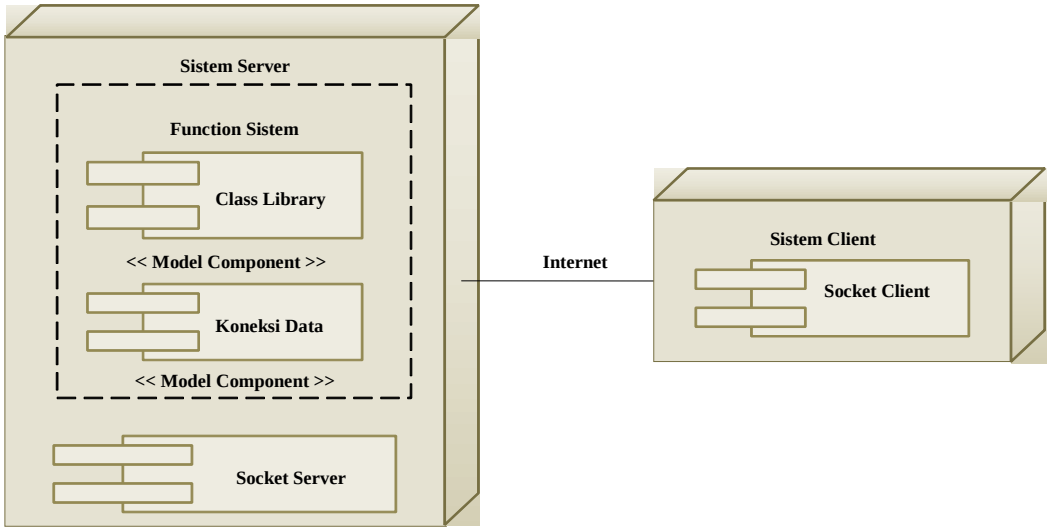
Component diagram menggambarkan struktur dan hubungan antar kompenen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Adapun gambaran *component* diagram dapat dilihat pada gambar dibawah ini.



Gambar III.7. *Component* Diagram

III.3.6. *Deployment* Diagram

Deployment diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur system, dimana komponen akan terletak (pada mesin, server atau piranti keras apa, bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal. Adapun gambaran *deployment* diagram dapat dilihat sebagai berikut.



Gambar III.8. *Deployment* Diagram

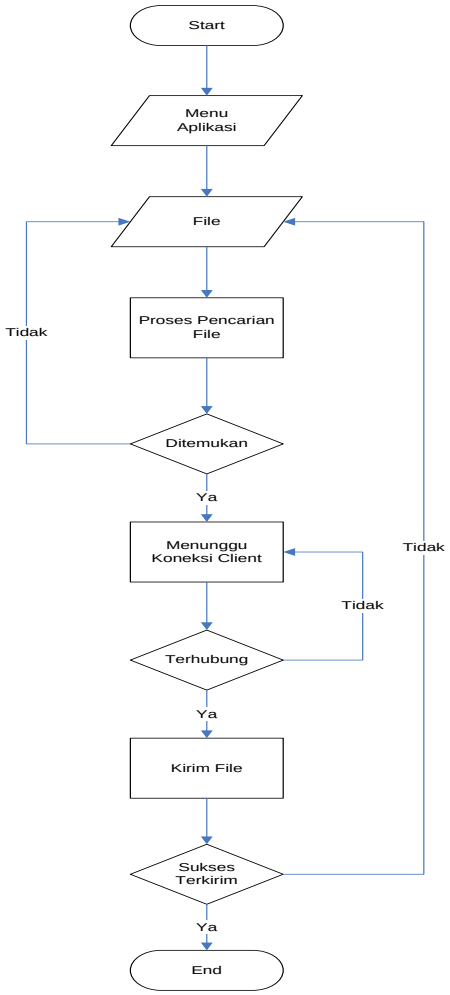
III.4 *Flowchart* Sistem

Secara umum aplikasi ini berjalan dengan konsep *client server*, sehingga terdapat dua buah *flowchart* yang akan dirancang. Adapun flowchart aplikasi yang dirancang dapat dilihat pada penjelasan dibawah ini

.

III.4.1 *Flowchart Server*

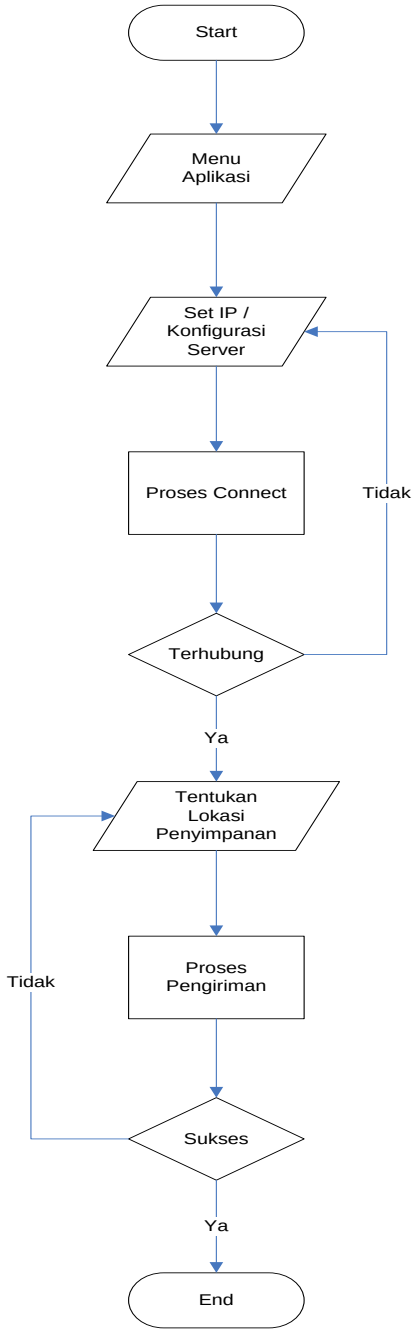
Adapun rancangan flowchart pada sisi server dapat dilihat pada gambar dibawah ini.



Gambar III.9. *Flowchart Server*

III.4.2 *Flowchart Client*

Adapun rancangan flowchart pada sisi client dapat dilihat pada gambar dibawah ini.



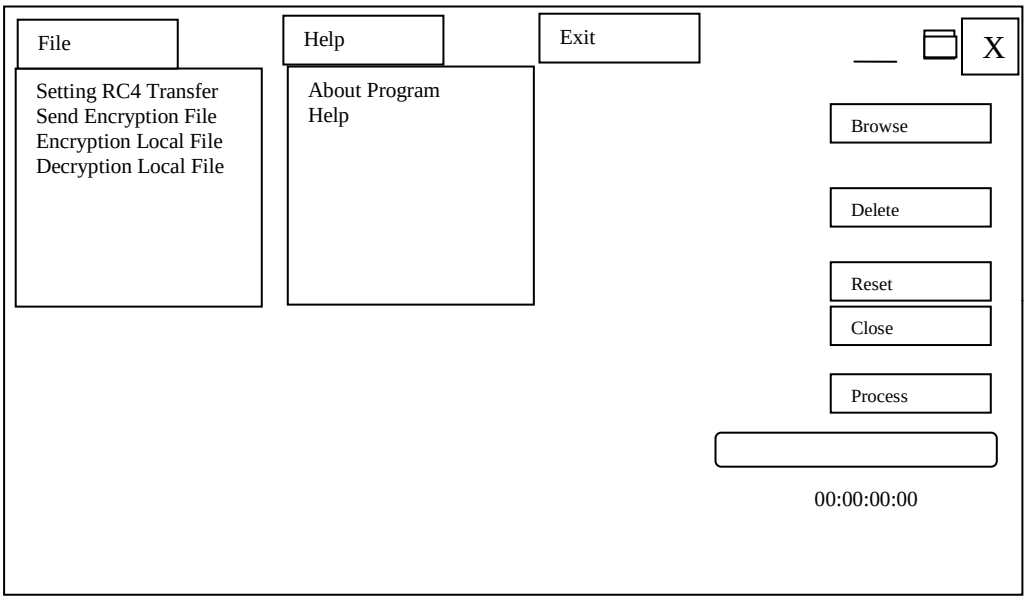
Gambar III.10. *Flowchart Client*

III.5 Rancangan Aplikasi

Adapun rancangan dari aplikasi transfer file gambar dengan algoritma *RC4* yang berbasis client-server adalah sebagai berikut :

III.5.1. Rancangan Layar Form Utama

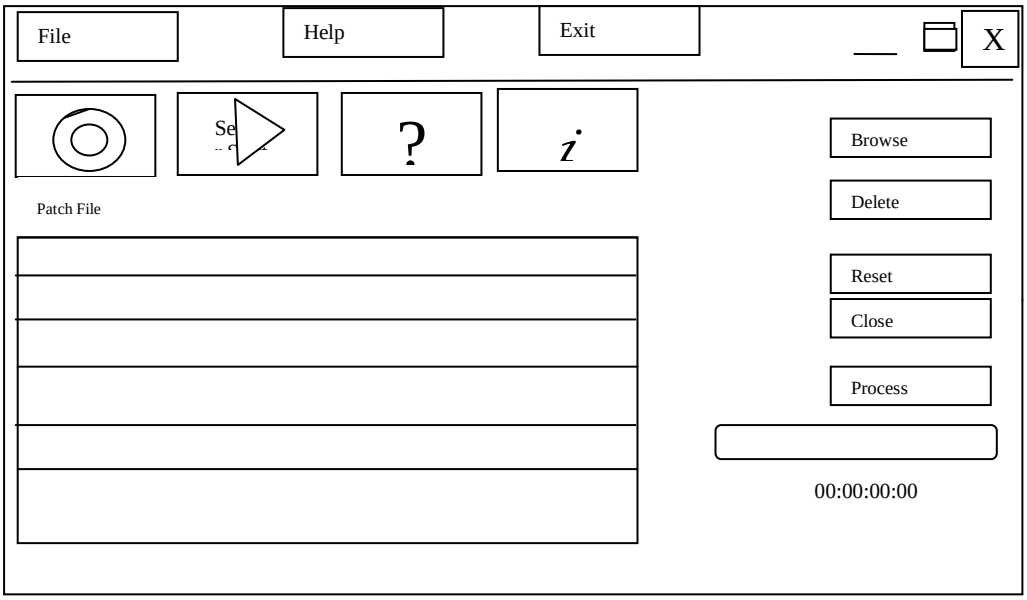
Pada halaman *form utama* menampilkan *form* yang berfungsi digunakan oleh pengguna dari sisi *client*, yang untuk dapat seperti gambar III.11 berikut :



Gambar III.11. Form Utama

1. Rancangan Form Menu

   Adapaun rancangan Form menu yang akan digunakan untuk proses enkripsi dapat dilihat pada gambar III.12 berikut.



Gambar III.12. Form Menu