PENERAPAN KEAMANAN SMS (*SHORT MASSAGE SERVICE*) DENGAN ALGORITMA ELGAMAL MENGGUNAKAN METODE *ELECTRONIK CODE BOOK* (ECB)

Disusun dan Disajikan Untuk Memenuhi Persyaratan Ujian
Akhir Memperoleh Gelar Sarjana Komputer
Fakultas Sains Dan Teknologi
Universitas Pembangunan Panca Budi
Medan

SKRIPSI

OLEH :

NAMA	**: RANIA**
N.P.M	**: 1514370114**
Program Studi	**: SISTEM KOMPUTER**

PROGRAM STUDI SISTEM KOMPUTER
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS PEMBANGUNAN PANCA BUDI
MEDAN
2019

ABSTRAK

RANIA

"Penerapan Keamanan Sms (*Short Massage Service*) Dengan Algoritma Elgamal Menggunakan Metode *Electronik Code Book* (ECB)"

2019

Teknologi komunikasi saat ini begitu sangat pesat perkembanganya, dimana perkembangan itu dapat mempermudah setiap elemen masyarakat dalam bertukar pesan informasi dengan mudah dan cepat. Teknologi informasi komunikasi yang sangat banyak digunakan saat ini yaitu teknologi komunikasi dengan menggunakan SMS *(Short Message Service).* namun dalam pengiriman suatu pesan SMS *(Short Message Service)* terkadang memiliki suatu kendala dimana kendala itu dapat berupa lemahnya suatu keamanan data yang membuat pesan yang kita kirim dapat disadap oleh orang lain. Maka dari itu suatu pesan harus dilindungi dengan menerapkan keamanan pada SMS *(Short Message Service)* menggunakan Algoritma Elgamal dengan Metode ECB. Dimana mekanisme kerja akan dilakukan dengan mengirim pesan singkat ke terminal-terminal lain dengan menggunakan suatu modem wavecome. Dimana modem itu berfungsi sebagai penerima dan Metode enkripsi yang digunakan yaitu dengan teknik XOR menggunakan pemograman PHP sebagai pemograman aplikasi yang dibangun.

Kata Kunci : Algoritma Elgamal, ECB *(Electronik Code Book)*, Modem Wavecome, SMS *(Short Massage Service)*, XOR

DAFTAR ISI

KATA PENGANTAR

Assalamu'alaikum Warahmatullah Wabarakatuh

Puji syukur penulis ucapkan kepada Allah SWT atas berkat, kesehatan, rahmat, taufik dan hidayah yang diberikan sehingga dapat menyelesaikan laporan Skripsi yang berjudul "PENERAPAN KEAMANAN SMS (*SHORT MASSAGE SERVICE*) DENGAN ALGORITMA ELGAMAL MENGGUNAKAN METODE *ELECTRONIK CODE BOOK* (ECB) " yang dapat diselesaikan dengan baik.

Dalam penulisan laporan Skripsi ini penulis menyampaikan ucapan terimakasi yang tak terhingga kepada nama-nama yang tertera dibawah ini:

1. Kepada Ayahanda Yamisdan Tanjung dan ibunda Marsinem tercinta yang telah begitu berjasa memberikan dukungan moril juga materil serta Doa yang telah dipanjatkan pada Allah SWT
2. Bapak Ir. Bhakti Alamsyah, M.T., Ph.D. selaku Rektor I Universitas Pembangunanan Panca Budi Medan.
3. Ibu Sri Shindi Indira, S.T.,M.Sc. selaku Dekan Fakultas Sains & Teknologi Universitas Pembangunan Panca Budi Medan
4. Bapak Dr. Muhammad Iqbal, S.Kom.,M.Kom selaku ketua Program Studi Sistem Komputer Fakultas Sains & Teknologi Universitas Pembangunan Panca Budi Medan.
5. Kepada Dosen Pembimbing I Bapak Supiyandi, S.Kom,.M.kom
6. Kepada Dosen Pembimbing II Ibu Ika Devi Perwitasari, S.Kom,.M.kom
7. Seluruh Dosen dan Staf Pegawai Fakultas Ilmu Komputer yang telah banyak membantu dalam kelancaran seluruh aktivitas perkuliahan.
8. Kepada kakak penulis yaitu Dahlia Tanjung, Ratna sari Tanjung, Rasia Tanjung dan Sopian Toni Tanjung yang telah memberikan dukungan dan doa yang begitu besar.
9. Temen-temen yang telah memberikan masukan, doa dukungan serta motivasi sehingga penulis dapat menyelesaikan laporan Skripsi ini.

Semoga kebaikan atas Doa dukungan dan motivasi yang diberikan dapat dibalas kelak oleh Allah SWT dengan pahala yang berlipat ganda, Amin. Penulis menyadari bahwa ada banyak kekurangan yang ada dalam laporan tugas akhir skripsi. oleh karena itu, penulis mengharapkan kritik dan saran fositif. Semoga laporan Skripsi ini dapat memberikan manfaat bagi pembaca. Akhir penulis ucapkan terima kasih.

Wassalamu'alaikum Warahmatullah Wabarakatuh

Medan, 02 September 2019
Penulis

RANIA
1514370114

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pengujiaan sistem yang telah dilakukan, maka dapat diambil beberapa rumusan masalah sebagai berikut:

1. Keamanan pesan SMS dapat ditingkatkan dengan kombinasi teknik XOR pada mode operasi ECB 128 *bits* yang digunakan untuk mengenkripsi pesan, dan algoritma Elgamal yang digunakan untuk mengenkripsi kunci simetris ECB. Dengan kombinasi kedua algoritma ini pesan akan jauh lebih aman dan tidak ada kunci rahasia yang dikirimkan sehingga keamanan jauh lebih terjaga.
2. Kombinasi algoritma Elgamal dengan algoritma XOR pada mode operasi blok *cipher* ECB untuk keamanan pesan SMS dengan aplikasi berbasis *web* harus menggunakan *library Gammu* sebagai *service* layanan pengiriman dan penerimaan SMS. Panjang pesan yang dikirim akan semakin panjang, hal ini dikarenakan *cipher key* yang dibutuhkan untuk proses dekripsi ditambahkan ke dalam pesan. Selain itu, penerima pesan harus mengirimkan terlebih dahulu *public key* kepada pengirim pesan agar pengirim pesan dapat mengenkripsi kunci simetris ECB.

5.2. Saran

Beberapa saran yang diusulkan untuk mengembangkan penelitian ini menjadi lebih baik adalah sebagai berikut:

1. Menggunakan algoritma simetris dan asimetris yang lainnya untuk ditetapkan dalam pengamanan pesan SMS untuk menemukan algoritma mana yang paling efisien dalam memberikan keamanan pada pesan SMS.
2. Aplikasi dapat dikembangkan ke dalam aplikasi berbasis mobile, baik berbasis sistem operasi andorid maupun sistem operasi iOS untuk memudahkan dalam pengiriman pesan SMS.
3. Disarankan juga untuk menerapkan algoritma yang diusulkan dalam penelitian ini untuk mengenkripsi pesan *chatting online* berbasis internet.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Kebutuhan Spesifikasi Minimum *Hardware* dan *Software*

Setelah sistem telah berhasil dibangun dalam bentuk aplikasi, maka untuk dilakukan implementasi agar aplikasi berjalan dengan lancar, dibutuhkan beberapa perangkat yang harus tersedia, beberapa perangkat tersebut antara lain :

4.1.1 Perangkat Keras

1. Komputer

Saat melakukan pengimplementasian sistem ini untuk diuji, digunakan sepesifikasi komputer sebagai berikut :

Processor : *Intel Pentium* Core i3 CPU G2030 @ 3.00 GHz

RAM : 2 GB

HD : 500 GB

Monitor : Samsung 21”

VGA : *Intel (R) HD Graphics*

Keyboard : *Logitech*

Mouse : *Powerlock*

Adapun perangkat minimum yang dapat digunakan agar sistem tetap dapat berjalan secara optimal adalah sebagai berikut :

Processor : *Intel Pentium* IV 2.4GHz

RAM : 512 MB

HD : *free* 1 GB

Monitor : 19”

VGA : *onboard* dari *motherboard*

Keyboard : apapun

Mouse : apapun

2. Modem

Modem digunakan sebagai perantara pengiriman SMS melalui aplikasi yang berhasil dibangun pada komputer. Sehingga pada modem ini lah terjadi pengiriman dan penerimaan pesan.

3. Kartu GSM

Kartu GSM diperlukan sebagai penyedia layanan SMS dari provider seperti telkomsel, indosat, atau XL.

4. Dua buah laptop

Dua buah laptop dibutuhkan sebagai pengirim dan penerima pesan dari aplikasi yang dibangun pada setiap masing-masing laptop yang mampu mengenkripsi pesan dan mendekripsi pesan yang dikirim maupun yang diterima oleh masing-masing pengguna. Kedua laptop harus menggunakan modem sebagai perantara pengirim dan penerima pesan SMS

4.1.2 Perangkat Lunak Komputer

1. Sistem Operasi

Sistem Operasi yang digunakan dalam pengujian adalah *Windows* 7 Professional buatan pabrikan *Microsoft*. Aplikasi juga masih dapat berjalan dengan baik pada Sistem Operasi *Windows* XP, *Windows* 8, *Windows* 8.1, maupun *Windows* 10 dan juga pada sistem operasi Linux dari berbagai distro.

2. XAMPP 1.7.3

Merupakan aplikasi yang menyediakan paket perangkat lunak yang terdiri dari beberapa perangkat lunak yang diperlukan untuk menjalakan aplikasi yang dibangun, yaitu : *Apache* (*web server*), MYSQL (*database*), PHP (*server side scripting*) dan *PhpMyAdmin*. Aplikasi masih dapat berjalan dengan baik dengan menggunakan XAMPP versi yang lebih rendah.

Aplikasi ini dibutuhkan jika pengujian dilakukan secara *local server* atau *localhost.* Sedangkan jika aplikasi telah memiliki domain sendiri dan terhubung dengan *internet,* maka aplikasi ini tidak lagi dibutuhkan, aplikasi akan dapat diakses langsung dari komputer menggunakan *browser*.

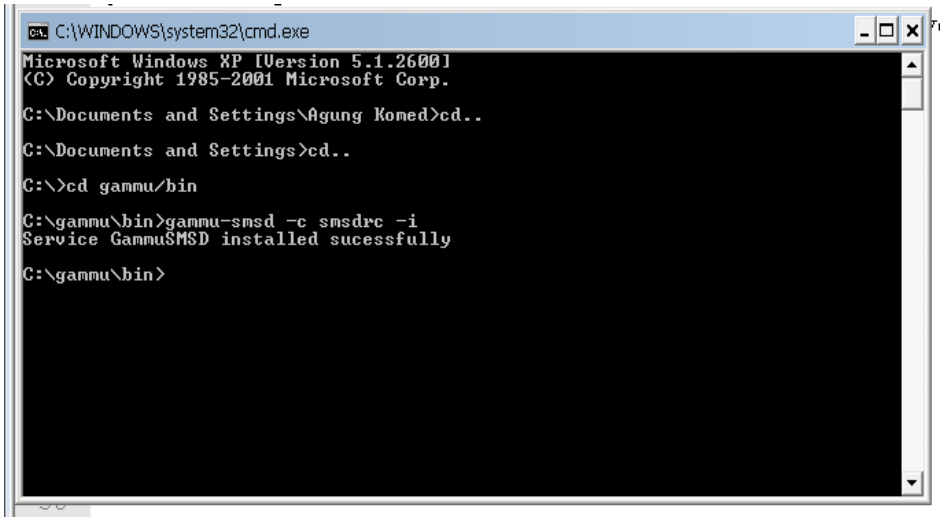
3. *Browser*

Digunakan untuk mengeksekusi aplikasi, *browser* yang digunakan adalah *Google Chrome 73.0.* Aplikasi masih dapat berjalan dengan baik menggunakan *browser* lain seperti *Mozilla Firefox*, *Internet Explore* atau *Opera*.

4.2 Pengujian Aplikasi dan Pembahasan

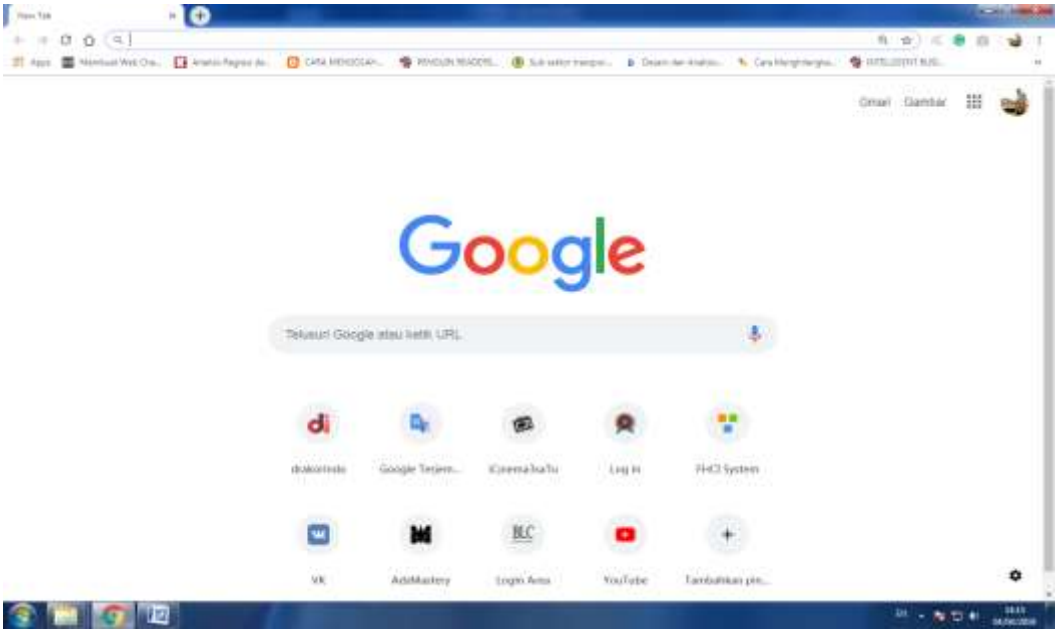
Untuk mengimplementasikan aplikasi, maka copy folder gammu ke alamat file local disk C. lalu untuk mengaktifkan fasilitas pengiriman pesan atau SMS (*Short Message Service*) yang terdapat didalam aplikasi, maka terlebih dahulu *user* harus menginstal *service* gammu ke dalam komputer. Cara-caranya adalah sebagai berikut :

1. Colokkan modem ke dalam usb komputer yang akan digunakan oleh aplikasi sebagai sarana pengiriman pesan yang akan digunakan oleh aplikasi.
2. Buka cmd (Command prompt) lalu ketikkan perintah seperti yang ada digambar dibawah ini:



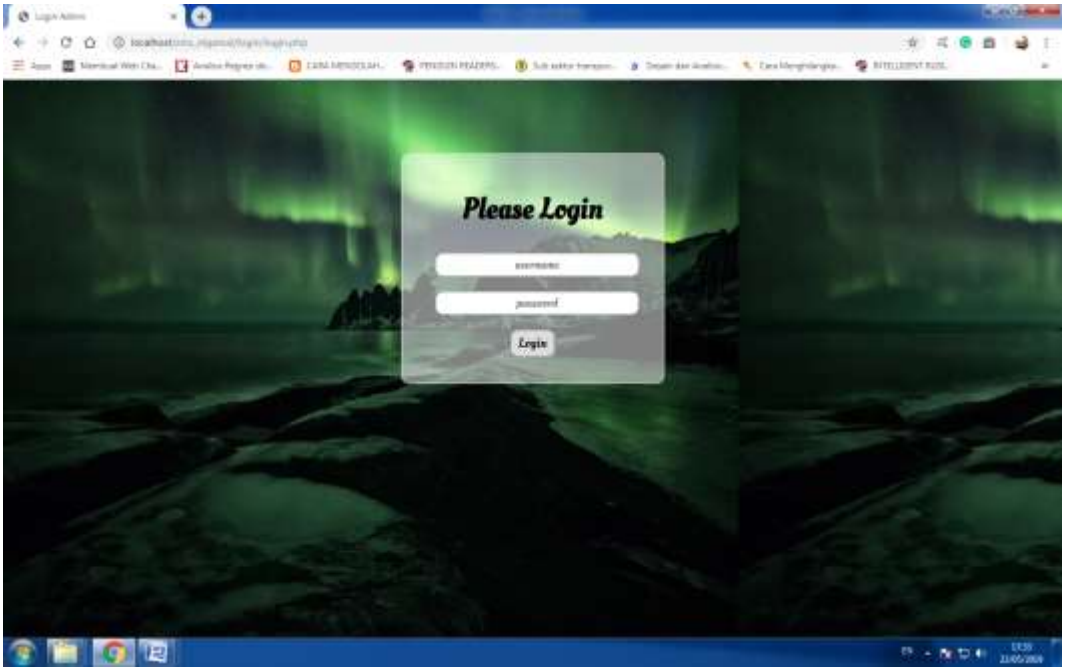
Gambar 4.1 Tampilan CMD untuk Menginstal Service Gammu
Sumber: Hasil Pengujian Aplikasi (2019)

File aplikasi diletakkan ke dalam folder sms_elgamal dengan alamat file C:/xampplite/htdocs/sms_elgamal/. Untuk menjalankan aplikasi, maka dibutuhkan sebuah *browser*, dalam pengujian yang dilakukan menggunakan *browser* Google Chrome. Setelah menjalankan aplikasi Google Chrome, maka tampilan Google Chrome sebagai berikut:



Gambar 4.2 Tampilan Awal Google Chrome
Sumber: Hasil Pengujian Aplikasi (2019)

Ketikkan alamat http:localhost/sms_elgamal pada area *address* pada aplikasi Google Chrome, maka aplikasi akan ditampilkan seperti gambar berikut:



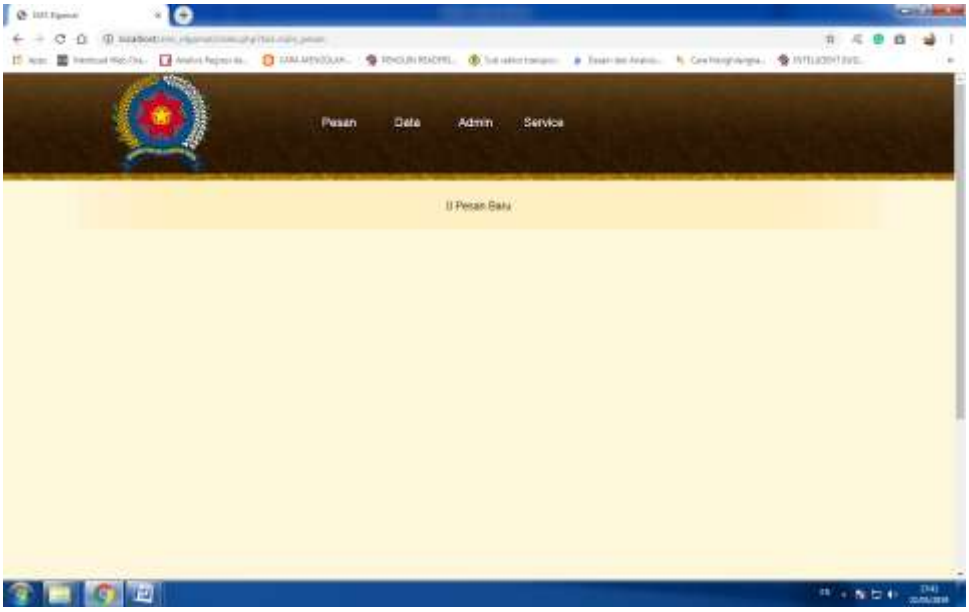
Gambar 4.3 Tampilan Halaman Login Admin dari Aplikasi
Sumber: Hasil Pengujian Aplikasi (2019)

Untuk *login* ke dalam aplikasi, maka ketikkan *username* dan *password* yang tepat, yaitu:

Username = admin

Password = system

Tampilan utama dari aplikasi akan ditampilkan sebagai berikut:

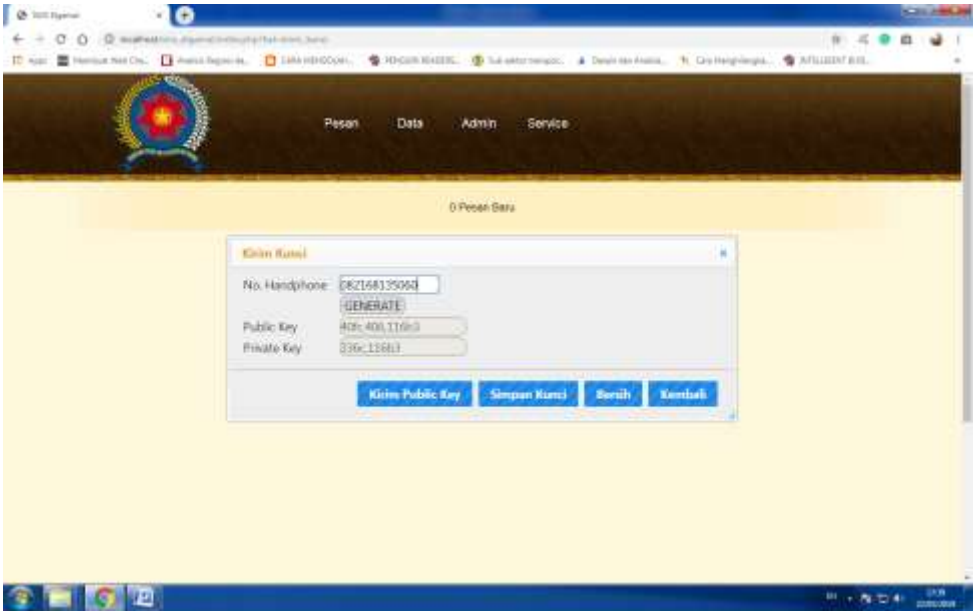


Gambar 4.4 Tampilan Utama dari Aplikasi
Sumber: Hasil Pengujian Aplikasi (2019)

Pada tampilan terdapat area informasi atau notifikasi pesan yang masuk dan belum di baca. Pada gambar notifikasi tertulis 0 Pesan Baru sehingga menunjukkan tidak terdapat pesan masuk yang belum di baca. Pengujian dilakukan dengan beberapa tahapan, yaitu:

1. **Pembangkitan Kunci**

   Untuk menjalankan proses ini, maka klik menu Pesan lalu sub menu Kirim Kunci, maka *form* untuk pembangkitan kunci akan ditampilkan oleh aplikasi seperti gambar berikut:



Gambar 4.5 Tampilan Pembangkitan Kunci Elgamal
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat terdapat sebuah tombol *Generate* yang berfungsi untuk membangkitkan *public key* dan *private key* secara acak. Tombol ini dapat diklik berulang kali dan akan selalu membangkitkan *public key* dan *private key* yang berbeda-beda. Lalu terdapat empat buah tombol di akhir dari *form* pengiriman pesan, yaitu:

a. Kirim *Public key*

Tombol ini berfungsi untuk mengirim *public key* yang berhasil dibangkitkan kepada nomor *Handphone* penerima yang di*input*kan oleh *user*.

b. Tombol Simpan Kunci

Tombol ini berfungsi untuk menyimpan *public key* dan *private key* ke dalam database, sehingga dapat digunakan oleh *user* untuk melakukan dekripsi.

c. Tombol Bersih

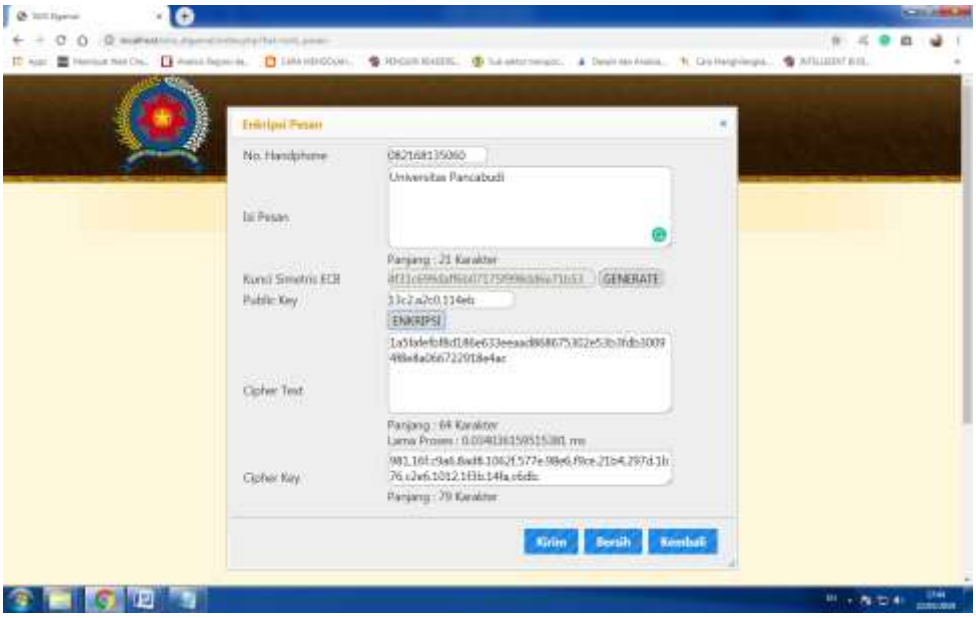
Tombol ini berfungsi untuk membersihkan seluruh *form* pembangkitan kunci seperti semula.

d. Tombol Kembali

Tombol ini berfungsi untuk menutup *form* atau kotak dialog pembangkitan kunci.

2. Enkripsi dan Pengiriman Pesan

Menu ini dapat dijalankan dengan mengklik Menu utama Pesan lalu sub menu Tulis Pesan, maka aplikasi akan menampilkan *form* penulisan pesan seperti gambar berikut:



Gambar 4.6 Tampilan Form Pengiriman Pesan
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat terdapat beberapa *textfield* yang harus diisi atau terisi otomatis, yaitu:

a. Nomor Kartu Seluler

Textfield ini menampung nomor Kartu Seluler yang di*input* oleh *user*.

b. Pesan

Area ini merupakan *textarea* dimana pesan utama yang akan dikirimkan oleh pengirim pesan kepada penerima pesan.

c. Kunci Simetris ECB

Textfield ini akan terisi secara otomatis jika *user* mengklik tombol *Generate* yang ada disebelah *textfield* Kunci Simetris ECB. Aplikasi akan membangkitkan 128 *bits* atau 32 karakter heksadesimal secara acak sebagai Kunci Simetris ECB.

d. *Public key*

Textfield yang menampung *public key* yang di*input*kan oleh *user* yang digunakan untuk mendekripsi Kunci Simetris ECB. *Public key* diperoleh oleh pengirim pesan dari penerima pesan.

e. Tombol Enkripsi

Tombol ini digunakan untuk melakukan proses enkripsi terhadap pesan dan Kunci Simetris ECB menjadi *cipher text* dan *cipher key*.

f. *Cipher text*

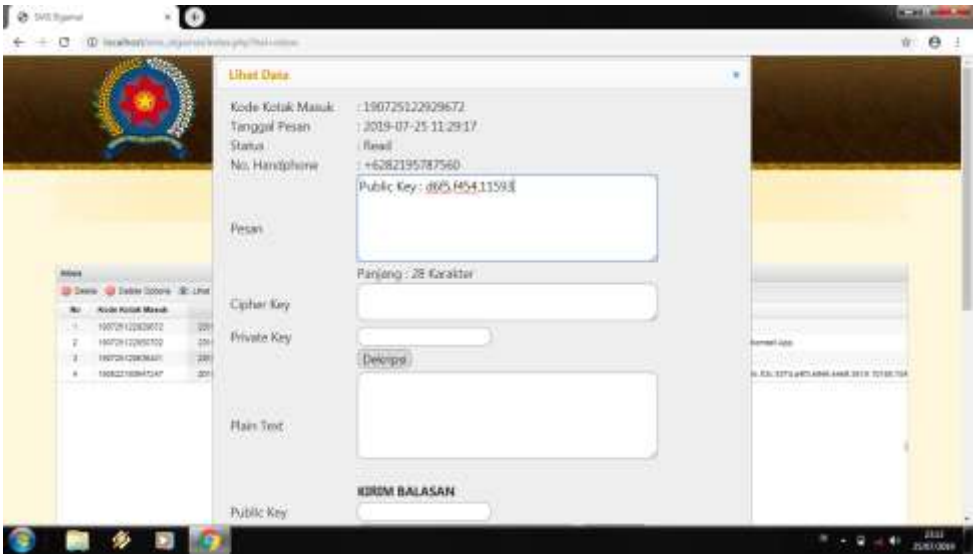
Textarea yang menampung *cipher text* hasil enkripsi pesan dengan mode operasi ECB 128 *bits* menggunakan Kunci Simetris ECB.

g. *Cipher key*

Textarea yang menampung *cipher text* hasil enkripsi dari Kunci Simteris ECB dengan algoritma Elgamal menggunakan *Public key*

Di bagian bawah dari *cipher text* terdapat keterangan panjang *cipher text* yang dihasilkan dan lama waktu proses untuk enkripsi pesan dan Kunci

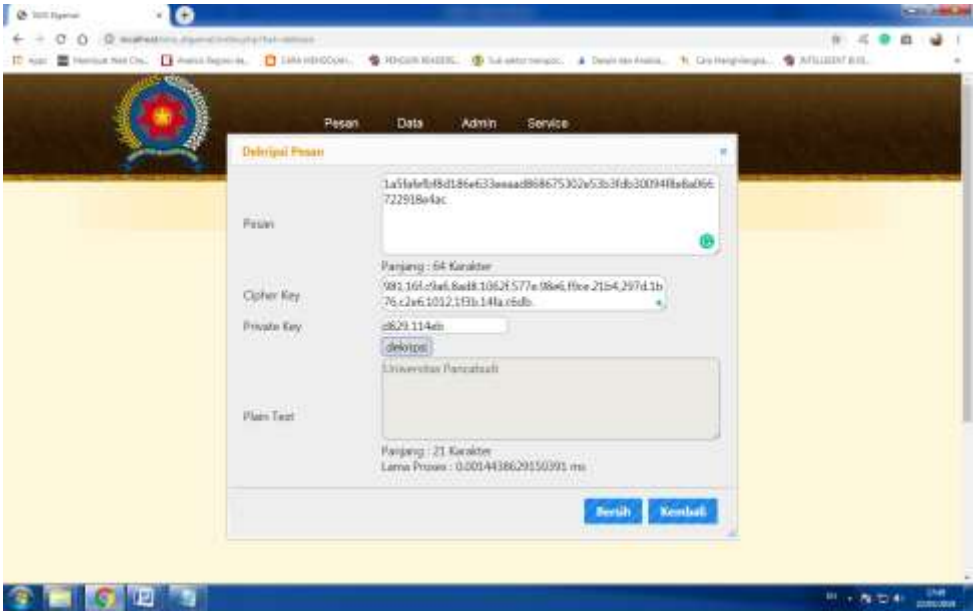
Simetris ECB. Sedangkan di bagian bawah dari *cipher key* terdapat keterangan panjang *cipher key* yang dihasilkan. *Form* pengiriman pesan memiliki tiga buah tombol, yaitu: tombol Kirim yang digunakan untuk mengirim *Cipher text* dan *Cipher key* secara bersamaan dalam satu pesan ke nomor nomor tujuan. Tombol Bersih untuk membersihkan *form* pengiriman pesan seperti semula. Tombol Kembali berfungsi untuk menutup *form* pengiriman pesan atau kotak dialog pengiriman pesan.



Gambar 4.7 Hasil Pesan yang Diterima melalui Seluler
Sumber: Hasil Pengujian Aplikasi (2019)

3. Dekripsi Pesan

Menu dekripsi dapat dijalankan dengan mengklik menu utama Pesan, lalu sub menu Dekripsi, maka aplikasi akan segera menampilkan *form* dekripsi pesan seperti yang ditunjukkan gambar berikut:



Gambar 4.8 Tampilan Form Dekripsi Pesan
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat terdapat beberapa *textfield* pada *form* yang dapat isi oleh *user* ataupun terisi secara otomatis, yaitu:

a. Pesan

Textfield yang menampung *cipher text* yang akan didekripsi menjadi *plain text*.

b. *Cipher key*

Textfield yang menampung *cipher key* yang akan didekripsi menjadi Kunci Simetris ECB yang akan digunakan untuk mendekripsi *cipher text* menjadi *plain text*.

c. *Private key*

Textfield yang menampung *private key* yang digunakan untuk mendekripsi *cipher key* menjadi kunci simetris ECB.

d. Tombol Dekripsi

Tombol ini berfungsi untuk menjalankan proses dekripsi terhadap *cipher text* dan *cipher key*.

e. *Plain text*

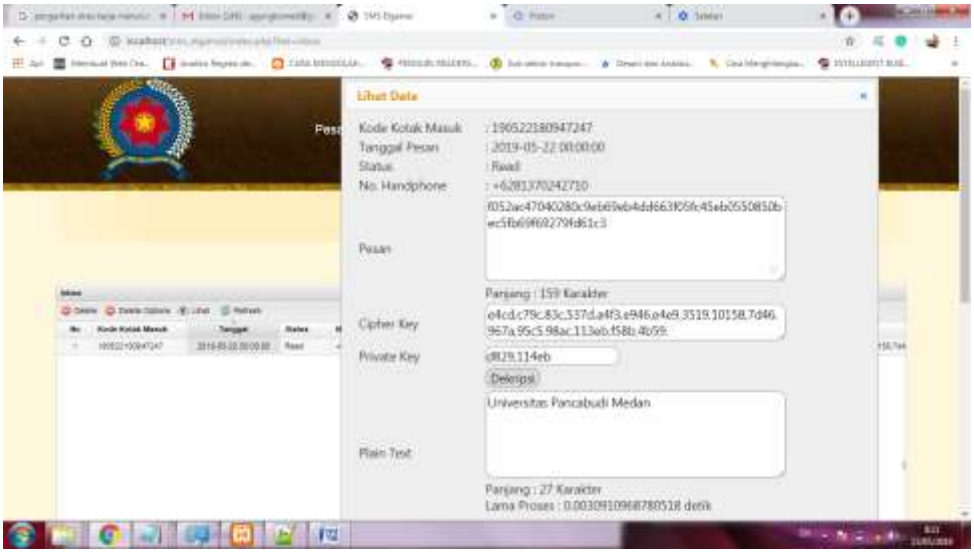
Textfield yang menampung hasil dekripsi *cipher text* menjadi pesan yang bermakna.

Dibagian bawah *plain text* terdapat keterangan panjang *plain text* yang dihasilkan serta waktu proses dekripsi. Terdapat dua tombol utama pada *form*, yaitu: tombol Bersih untuk menghapus *form* seperti semula dan tombol Kembali untuk menutup *form* dekripsi.

4. Dekripsi melalui *Inbox* dan Membuat Balasan

Terdapat fasilitas untuk melakukan dekripsi pesan yang masuk ke *Inbox* secara langsung, memberikan pesan balasan yang terenkripsi terhadap pesan masuk tersebut. Untuk menggunakan fasilitas ini klik menu utama Data lalu sub menu *Inbox*. Maka aplikasi akan segera membuka tabel penyimpanan pesan masuk di tabel *Inbox*. Pada tabel *Inbox* dapat dilihat keterangan Read dan Unread pada masing-masing pesan. Read berarti pesan telah dibuka atau dibaca, sedangkan Unread berarti pesan belum dibaca.

Untuk melihat pesan masuk, pilih salah satu pesan masuk yang ingin dibaca, kemudian klik tombol lihat, maka *form* dekripsi dan balasan pesan akan terbuka seperti gambar berikut ini:



Gambar 4.9 Tampilan Form Dekripsi Melalui Tabel Inbox

Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar dapat dilihat bahwa aplikasi secara otomatis akan memotong isi pesan, dimana *cipher text* akan diletakkan ke dalam *textarea* pesan, lalu *cipher key* akan diletakkan ke dalam *textarea cipher key*. *Form form* dekripsi di atas memiliki beberapa area sebagai berikut:

a. Kode Kotak Masuk

   Area ini menginformasikan kode unik dari pesan sebagai identifier.

b. Tanggal Pesan

   Area ini menginformasikan tanggal pesan diterima.

c. Status

   Area ini menginformasikan status pesan apakah telah dibaca sebelumnya atau belum pernah dibaca.

d. Nomor kartu seluler

   Area ini menginformasikan nomor kartu seluler pengirim pesan.

e. Pesan

Textfield yang memuat *cipher text* secara otomatis dari pesan yang dikirim oleh pengirim pesan.

f. *Cipher key*

Textfield yang memuat *cipher key* secara otomatis dari pesan yang dikirimkan oleh pengirim pesan.

g. *Private key*

Textfield ini diisi dengan *private key* yang dimiliki oleh *user* sebagai penerima pesan

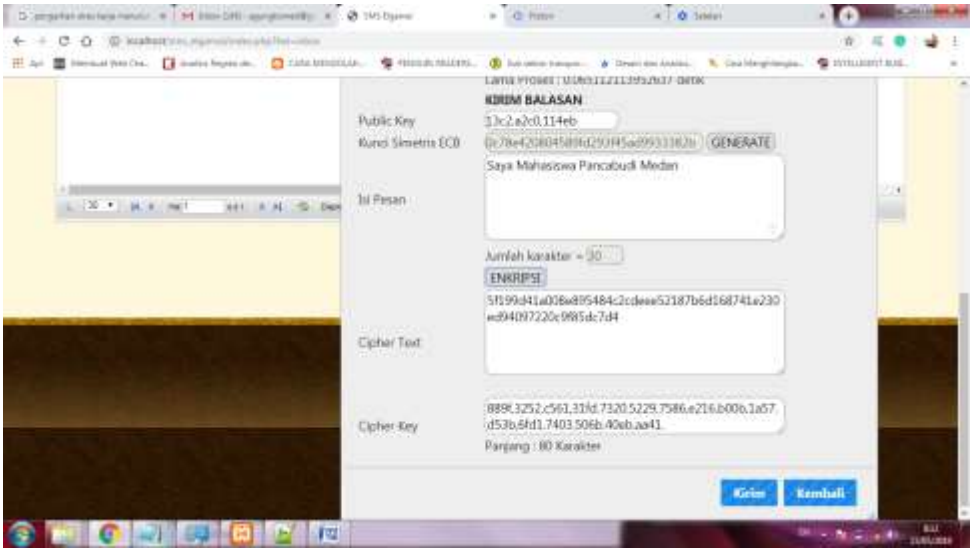
h. Tombol Dekripsi

Tombol ini berfungsi untuk mendekripsi *cipher text* yang diterima dan meletakkan *plain text* hasil dekripsi ke *textfield plain text*.

i. *Plain text*

Textfield yang memuat secara otomatis hasil dekripsi dari pesan atau *cipher text*.

Setelah mendekripsi isi pesan, maka *user* dapat langsung memberikan pesan balasan kepada pengirim pesan dengan mengisi pesan balasan di *form* balasan yang ada dibawah *form* dekripsi pesan, seperti yang ditunjukkan pada gambar berikut:



Gambar 4.10 Tampilan Form Dekripsi dan Balasan Melalui Tabel Inbox
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas menunjukkan *form* balasan pesan yang memiliki beberapa area sebagai berikut:

a. *Public key*

Textfield yang menampung *public key* yang di*input*kan oleh *user* yang digunakan untuk mendekripsi Kunci Simetris ECB. *Public key* diperoleh oleh pengirim pesan dari penerima pesan.

b. Kunci Simetris ECB

Textfield ini akan terisi secara otomatis jika *user* mengklik tombol *Generate* yang ada disebelah *textfield* Kunci Simetris ECB. Aplikasi akan membangkitkan 128 *bits* atau 32 karakter heksadesimal secara acak sebagai Kunci Simetris ECB.

c. Isi Pesan

Area ini merupakan *textarea* dimana pesan utama sebagai balasan yang akan dikirimkan oleh pengirim pesan kepada penerima pesan.

d. Tombol Enkripsi

Tombol ini digunakan untuk melakukan proses enkripsi terhadap pesan dan Kunci Simetris ECB menjadi *cipher text* dan *cipher key*.

e. *Cipher text*

Textarea yang menampung *cipher text* hasil enkripsi pesan dengan mode operasi ECB 128 *bits* menggunakan Kunci Simetris ECB.

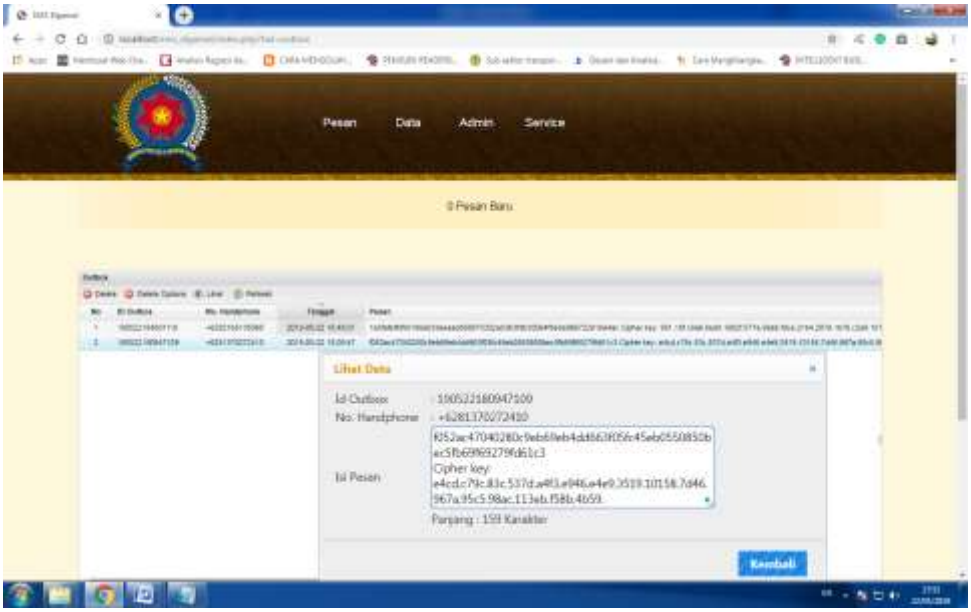
f. *Cipher key*

Textarea yang menampung *cipher text* hasil enkripsi dari Kunci Simteris ECB dengan algoritma Elgamal menggunakan *Public key*

Di bagian bawah dari *cipher text* dan isi pesan terdapat keterangan panjang *cipher text* yang dihasilkan. *Form* balasan pesan memiliki dua buah tombol, yaitu: tombol Kirim yang digunakan untuk mengirim *Cipher text* dan *Cipher key* secara bersamaan dalam satu pesan ke nomor *Handphone* tujuan. Tombol Kembali berfungsi untuk menutup *form* pengiriman pesan atau kotak dialog pengiriman pesan

5. Melihat Kotak Keluar

Menu ini digunakan untuk melihat daftar pesan yang dikirimkan. Untuk menggunakan fasilitas ini maka ketikkan menu utama Data lalu sub menu *Outbox*, maka aplikasi akan menampilkan tabel penyimpanan pesan keluar seperti gambar berikut:

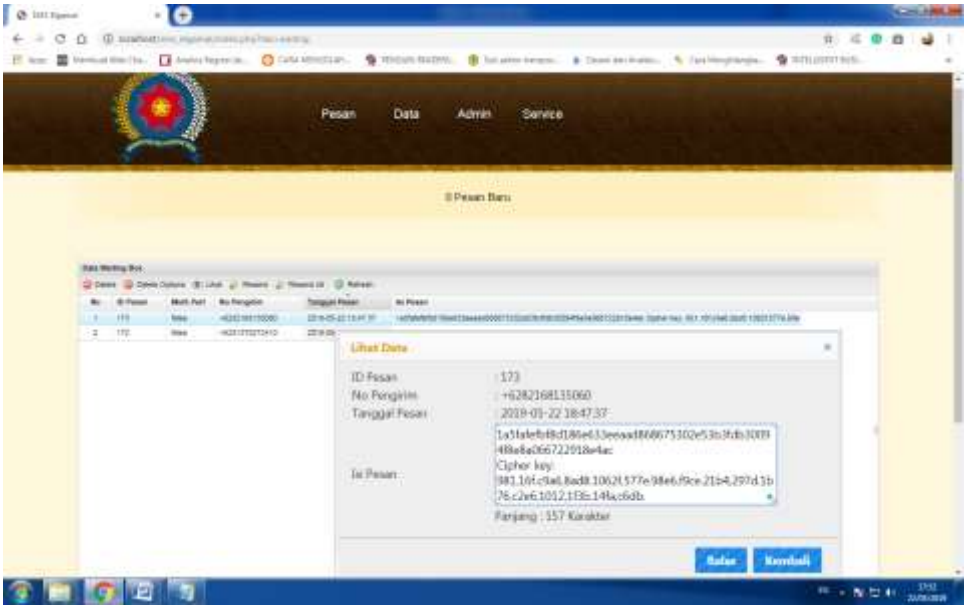


Gambar 4.11 Tampilan Tabel Kotak Keluar
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat tabel yang menampung data pesan keluar yang dikirimkan melalui aplikasi. Untuk membaca pesan keluar tersebut maka klik pesan yang akan dibaca lalu klik tombol lihat, maka *form* baca pesan akan ditampilkan seperti yng ditunjukkan pada gambar di atas.

6. *Waiting Box*

Menu ini merupakan fasilitas untuk melihat daftar pesan keluar yang telah dikirimkan tetapi belum sampai ke tujuan. Seluruh pesan yang masih pendding dan belum sampai ke tujuan akan ditampung di tabel *Waiting Box*. Jika pesan telah sampai ditujuan, maka pesan tersebut secara otomatis akan dihapus dari tabel *Waiting Box*. Untuk menggunakan fasilitas ini maka klik menu *Service* lalu sub menu *Waiting Box*, maka aplikasi akan menampilkan tabel peyimpanan *Waiting Box* seperti pada gambar berikut:



Gambar 4.12 Tampilan Tabel Penyimpanan Waiting Box
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat daftar pesan tunggu yang sedang dikirimkan. Terdapat beberapa tombol pada tebel dengan fungsi yang berbeda-beda, yaitu:

a. *Delete*

Tombol ini digunakan untuk menghapus salah satu pesan di *Waiting Box*, tombol ini hanya dapat menghapus sebuah pesan untuk satu proses.

b. *Delete Options*

Tombol ini digunakan untuk menghapus banyak pesan di *Waiting Box* sekaligus.

c. Lihat

Tombol ini digunakan untuk melihat pesan yang ada di *Waiting Box* ke dalam Kotak Dialog yang baru.

d. *Resend*

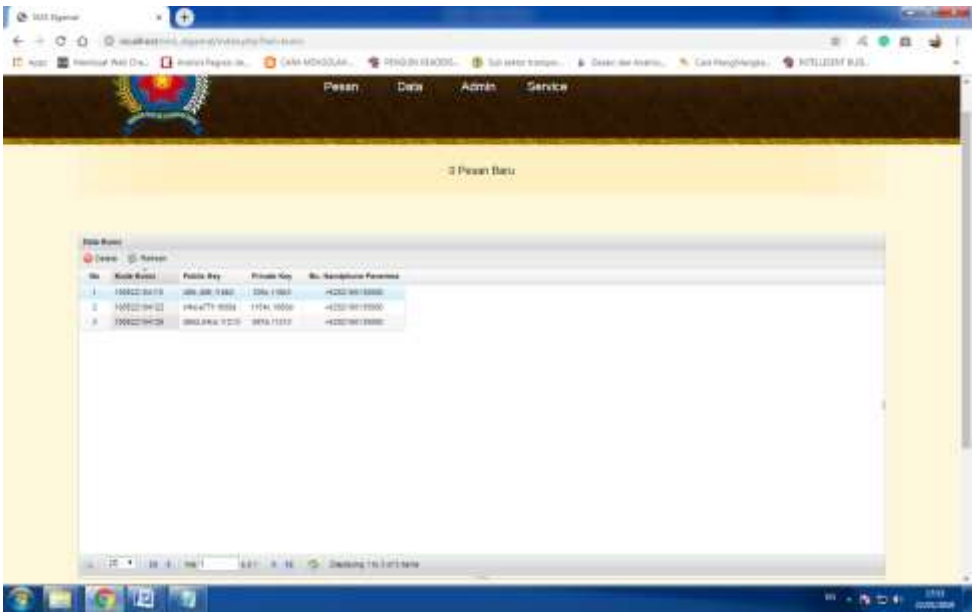
Tombol ini digunakan untuk mengirim ulang sebuah pesan yang dipilih yang ada di *Waiting Box*.

e. *Resend All*

Tombol ini digunakan untuk mengirim ulang seluruh pesan yang ada di *Waiting Box*.

7. Melihat Kunci Tersimpan

Menu ini digunakan untuk menyimpan *public key* dan *private key* yang berhasil dibangkitkan *user*. Kedua kunci ini penting untuk disimpan agar dapat digunakan kembali oleh *user* untuk proses dekripsi. Untuk menggunakan fasilitas ini maka klik menu utama Data lalu sub menu Kunci, maka aplikasi akan menampilkan tabel penyimpanan kunci seperti yang ditunjukkan pada gambar berikut ini:



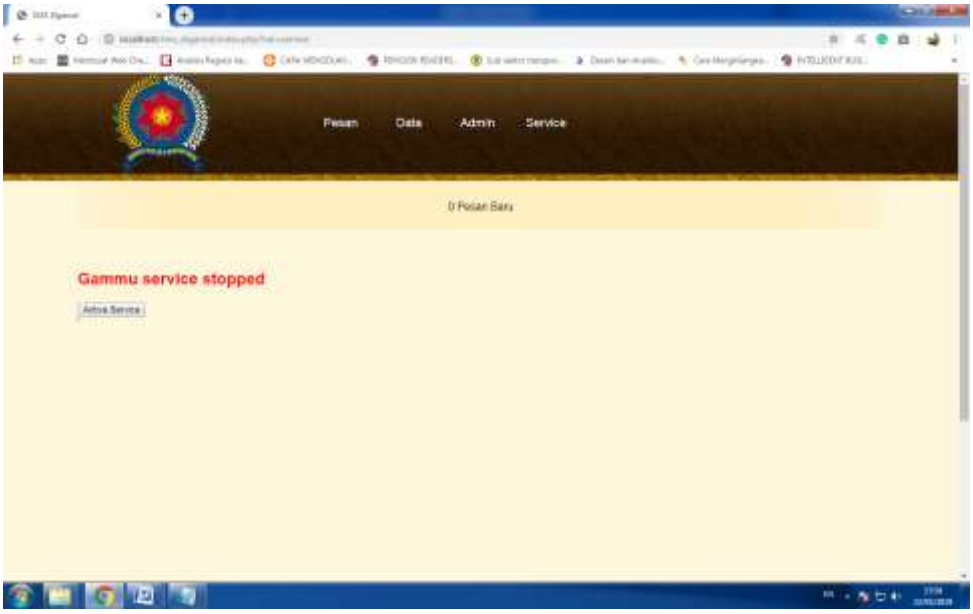
Gambar 4.13 Tampilan Tabel Penyimpanan Kunci Elgamal

Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat kunci yang disimpan adalah *private key*, *public key*, dan nomor seluler tujuan. Nomor seluler tujuan penting untuk disimpan untuk mengidentifikasi *private key* apa yang akan digunakan untuk mendekripsi pesan masuk berdasarkan nomor seluler pengirim.

8. *Service Gammu*

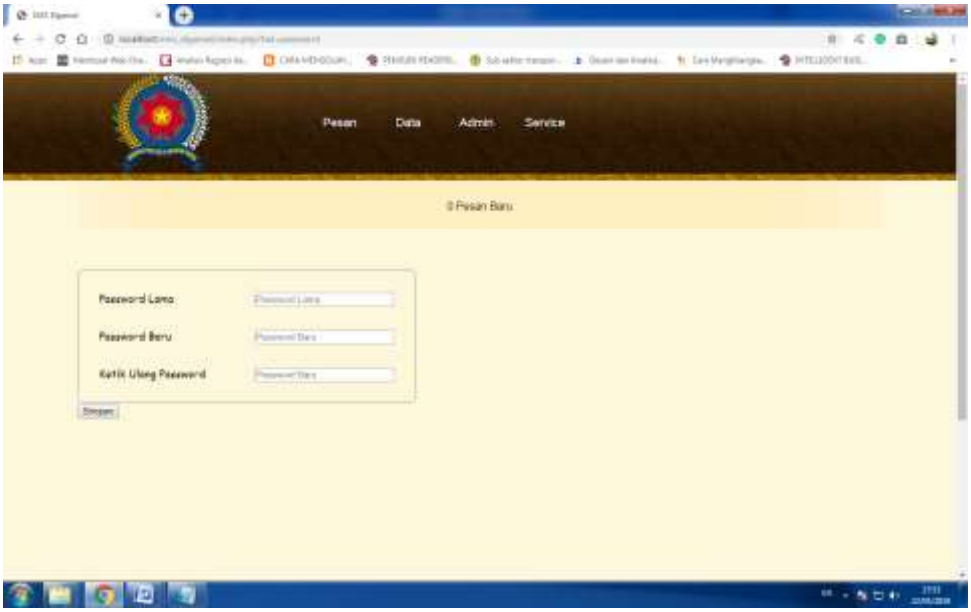
Menu ini digunakan untuk mengaktifkan fasilitas SMS pada aplikasi, dimana jika *Gammu Service is Running* berarti fasilitas SMS dapat digunakan, sehingga aplikasi dapat mengirim dan menerima SMS, tetapi jika *Gammu Service Stopped* berarti fasilitas SMS tidak dapat digunakan sehingga tidak dapat melakukan pengiriman maupun penerimaan pesan. Untuk menjalankan fasilitas ini klik menu utama *Service* lalu klik sub menu *Service* Gammu, maka aplikasi akan menampilkan halaman untuk mengaktifkan atau menonatifkan *service* Gammu seperti gambar berikut:



Gambar 4.14 Tampilan Form Service Gammu
Sumber: Hasil Pengujian Aplikasi (2019)

9. *Password*

User memiliki *password* yang dapat diubah sesuai dengan keinginan, untuk mengubah *password* maka klik menu utama Admin lalu sub menu *Password*, maka aplikasi akan menampilkan *form* penggantian *password user* seperti gambar berikut:



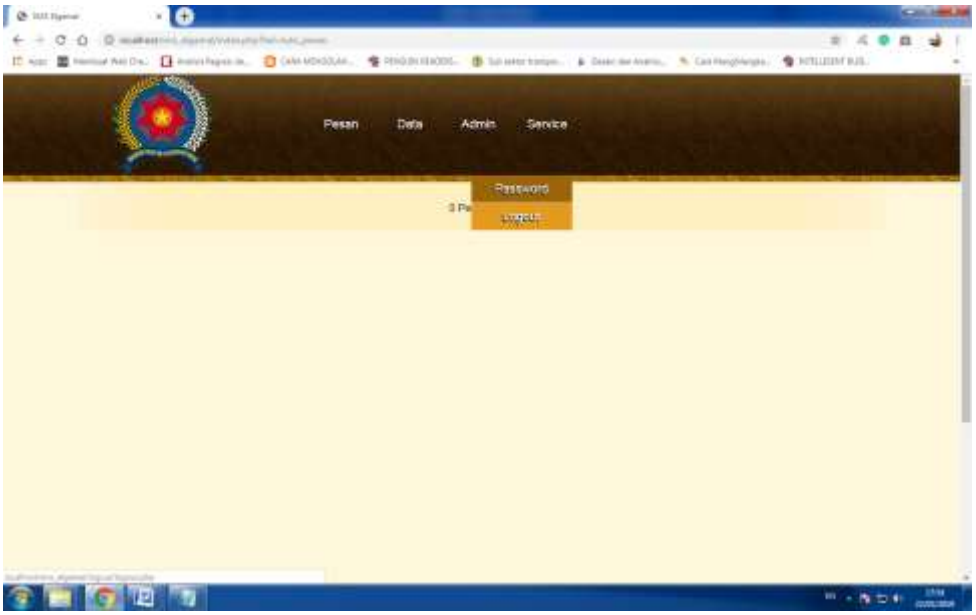
Gambar 4.15 Tampilan Utama Form Ganti Password
Sumber: Hasil Pengujian Aplikasi (2019)

Pada gambar di atas dapat dilihat *user* harus meng-*input*-kan *password* lamanya untuk menjamin bahwa *user* yang mengganti *password* benar-benar *user* yang sah, selain itu *user* juga harus meng-*input password* baru sebanyak dua kali untuk mencegah kesalahan pengetikan *password* sehingga menyebabkan akun tidak dapat terbuka nantinya.

10. *Logout*

Menu ini digunakan untuk keluar dari aplikasi dengan aman, dimana seluruh sesi *user* akan dihapus sehingga tidak meninggalkan jejak pada

aplikasi, hal ini menyebabkan *user* lain tidak dapat memasuki akun *user* yang lama tanpa *password*. Untuk melakukan *logout* maka klik menu utama Admin lalu klik sub menu *Logout* seperti gambar berikut:



Gambar 4.16 Tampilan Menu Logout
Sumber: Hasil Pengujian Aplikasi (2019)

BAB III
METODE PENELITIAN

3.1. Tahapan Penelitian

Penelitian ini dilakukan dengan beberapa tahapan, yaitu:

1. **Analisis Masalah**

   Tahap ini dilakukan untuk menganalisis masalah yang akan menjadi fokus penelitian untuk kemudian diteliti dan mendapatkan penyeesaian masalahnya

2. **Pengumpulan Data**

   Teknik pengumpulan data yang dilakukan untuk mendukung penelitian yang sedang dilakukan

3. **Analisis Penyelesaian Masalah**

   Analisis penyelesaian masalah dari masalah yang menjadi fokus penelitian berdasarkan berbagai data dan teori yang berhasil dikumpulkan.

4. **Analisis Proses**

   Analisis proses dari penyelesaian masalah yang diusulkan yang terdiri dari tahap pembentukan kunci, tahap enkripsi, dan tahap dekripsi

5. **Analisis Kebutuhan Sistem**

   Analisis terhadap berbagai kebutuhan untuk membangun sistem dalam menerapkan penyelesaian masalah yang diusulkan.

6. **Perancangan Sistem**

   Melakukan perancangan untuk membangun sistem yang terdiri dari rancangan sistem, rancangan database, dan rancangan *interface*.

7. **Pengujian**

   Melakukan pengujian terhadap sistem yang berhasil dibangun. Pengujian dimaksudnya untuk membuktikan apakah penyelesaian masalah yang diusulkan mampu menyelesaikan masalah yang menjadi fokus penelitian.

8. **Pengambilan Kesimpulan**

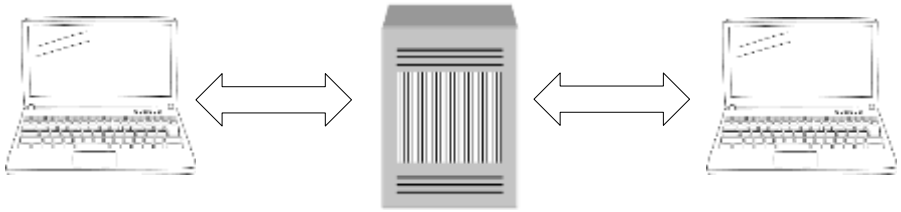
   Mengambil kesimpulan berdasarkan pengujian yang berhasil dilakukan.

3.2 Metode Pengumpulan Data

Penelitian ini berjenis penelitian laboratorium. Pengumpulan data dilakukan dengan mengumpulkan berbagai referensi melalui buku, jurnal nasional, dan jurnal international. Penelitian dilakukan dengan melakukan eksperimen komputer dengan menggunakan tools bahasa pemrograman.

3.3 Analisa Sistem yang Berjalan

Short Messaging Service merupakan salah satu fitur dari GSM yang dikembangkan dan distandarisasi oleh ETSI. Pada saat sebuah pesan SMS dikirim, maka pesan SMS tersebut tidak langsung dikirim ke nomor tujuan, akan tetapi terlebih dahulu dikirim ke SMS *Center* (SMSC) dengan prinsip *Store and Forward*, setelah itu baru dikirimkan kepada yang dituju. Proses pengiriman SMS dapat dilihat pada gambar dibawah ini:



Gambar 3.1 Skema Cara Kerja Pengiriman SMS.

Dengan adanya SMSC ini, status dari SMS yang dikirim akan diketahui, apakah telah sampai atau gagal diterima oleh laptop tujuan. Apabila laptop tujuan dalam keadaan aktif dan menerima SMS yang dikirim, maka laptop akan mengirim kembali pesan konfirmasi ke SMSC yang menyatakan bahwa SMS telah diterima. Kemudian SMSC mengirimkan kembali status tersebut kepada pengirim. Tetapi jika laptop tujuan dalam keadaan mati atau diluar jangkauan, SMS yang dikirimkan akan disimpan pada SMSC sampai periode validitas terpenuhi, Jika periode validitas terlewati maka SMS itu akan dihapus dari SMSC dan tidak dikirimkan ke laptop tujuan. Disamping itu juga SMSC akan mengirim pesan Informasi ke nomor pengirim yang menyatakan pesan yang dikirim belum diterima atau gagal.

3.3.1 Analisis Masalah

Pendistribusian data antar pengguna membutuhkan jaminan keamanan dimana data yang dikirim tidak dapat dibaca oleh orang lain, sehingga dibutuhkan suatu teknik untuk mengamankannya. Pengiriman pesan dengan menggunakan *Short Message Service* (SMS) sangat rentan disadap oleh orang lain, sehingga kerahasian pesan yang dikirim oleh pengirim pesan menjadi terancam. Jika pesan

tersebut merupakan pesan-pesan yang sangat rahasia, maka sangat berbahaya bila pesan tersebut sampai terbaca oleh orang lain yang tidak berhak. Untuk itu, dibutuhkan sebuah teknik pengaman data yang dapat menjaga kerahasiaan pesan yang dikirim, baik menjaga kerahasiaan pesan saat ditransmisikan atau saat pesan telah diterima dan disimpan oleh penerima pesan.

3.3.2 Analisis Penyelesaian Masalah

Kriptografi adalah salah satu teknik untuk mengamankan data dengan cara mengubah bentuk data menjadi bentuk yang tidak dapat dikenali sehingga pihak asing hanya dapat mengetahui isi data tersebut jika memiliki kunci deskripsi untuk mengembalikan pesan tersebut menjadi pesan aslinya.

Kriptografi terbagi menjadi dua jenis, yaitu kriptografi kunci simetris dan kriptografi kunci asimetris. Kekurangan dari algoritma simetris adalah kunci yang digunakan untuk melakukan enkripsi dan deskripsi adalah sama, sehingga proses pertukaran kunci menjadi sangat sulit dilakukan dengan aman antara pengirim pesan dan penerima pesan. Sedangkan kriptografi kunci asimtris memiliki kunci enkripsi dan kunci deskripsi yang berbeda satu dengan yang lainnya.

Kriptografi kunci simetris memiliki kemampuan proses enkripsi dan dekripsi yang cepat dan kuat, sehingga kriptografi kunci simetris sangat tepat digunakan untuk mengenkripsi pesan pada SMS, terlebih lagi jika kunci yang digunakan dibangkitkan secara acak dan memiliki panjang yang cukup. Salah satu teknik enkripsi dengan kriptografi yang paling sederhana, cepat, hemat, namun sangat kuat adalah dengan teknik XOR. Teknik XOR memiliki proses enkripsi dan dekripsi yang sangat sederhana namun sanga cepat dan hemat sumber daya

komputer. Walau sederhana tetapi teknik XOR sangat kuat jika digunakan pada mode operasi yang tepat.

Untuk meningkatkan keamanan dari *cipher text*, maka teknik XOR dapat dikombinasikan dengan mode operasi *Electronic Code Book* (ECB) dengan panjang blok sepanjang 128-*bits*. Karena panjang blok sepanjang 128-*bits*, maka panjang kunci yang digunakan harus sepanjang 128-*bits* dan dibangkitkan secara acak.

Walau kriptografi kunci simetris memiliki kelebihan dalam kecepatan, namun terdapat kekurangan pada kriptografi kunci simetris, yaitu masalah *key distribution*. Hal ini dikarenakan kunci untuk enkripsi dan dekripsi menggunakan kunci yang sama, dan kunci tersebut sangat rahasia. Sehingga agar penerima pesan dapat melakukan dekripsi, maka penerima pesan harus memiliki kunci simetris yang dimiliki oleh pengirim pesan, sehingga terjadi pertukaran kunci rahasia. Pertukaran ini sangat rentan terhadap penyadapan, sehingga tidak ada cara yang aman untuk mengirimkan kunci rahasia dari pengirim ke penerima pesan. Untuk mengatasi masalah ini, kunci rahasia harus diubah menjadi bentuk yang tidak lagi rahasia sehingga bisa dapat dengan aman dikirimkan ke penerima pesan, salah satu teknik yang dapat digunakan adalah dengan kriptografi kunci asimetris atau kunci publik.

Kriptografi kunci asimetris yang dikenal dengan algoritma kunci publik memiliki dua buah kunci yang berbeda dan untuk penggunaan yang berbeda. *Public key* digunakan untuk proses enkripsi dan bersifat tidak rahasia sehingga dapat didistribusikan atau dikirim dengan bebas tanpa rahasia atau takut disadap.

Private key digunakan untuk proses dekripsi dan bersifat sangat rahasia. Berbeda dengan kunci simetris yang dibangkitkan oleh pengirim pesan, *public key* dan *private key* dibangkitkan oleh penerima pesan, kemudia *public key* dikirim kepada pengirim pesan untuk digunakan pengirim pesan untuk proses enkripsi. Sedangkan *private key* tetap disimpan oleh penerima pesan untuk digunakan pada proses dekripsi. Dalam skenario ini dapat dilihat bahwa tidak terjadi pertukaran kunci rahasia, sehingga kunci rahasia yaitu *private key* tetap aman karena tidak pernah didistribusikan.

Kriptografi kunci asimetris dapat dimanfaatkan untuk mengenkripsi kunci simetris yang harus didistibusikan kepada penerima pesan sehingga kunci simetris yang dienkripsi dengan *public key* berubah ke dalam bentuk *cipher key* (bentuk kunci berubah menjadi bentuk yang acak dan tidak dipahami). *Cipher key* ini tidak rahasia dan tidak dapat digunakan untuk proses dekripsi, kecuali *cipher text* ini didekripsi terlebih dahulu dengan *private key* sehingga kembali menjadi kunci simetris. Dengan ini maka masalah *key* distribution telah teratasi.

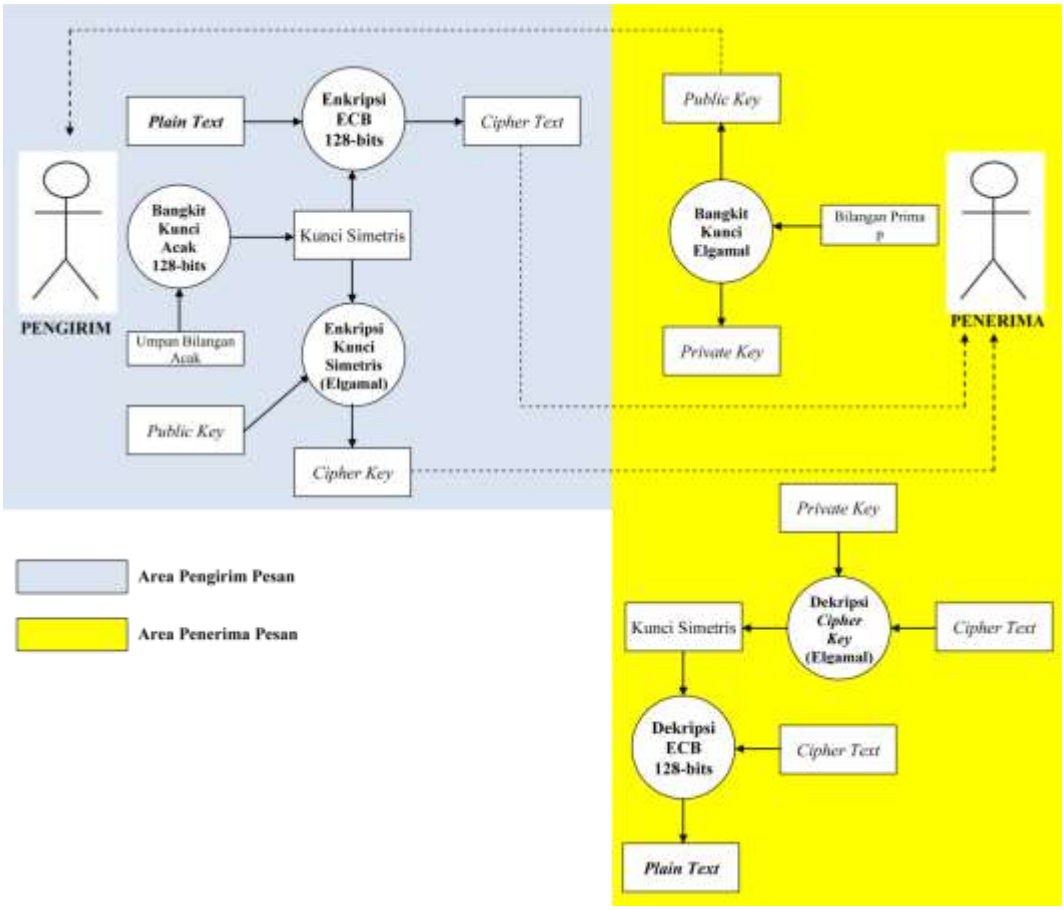
digunakan hanya untuk melakukan enkripsi pesan, sehingga *public key* boleh dikirim dari jalur yang tidak aman, dan boleh diketahui oleh orang lain, sedangkan kunci deskripsi atau dikenal dengan *private key* digunakan untuk melakukan pendeskripsian *cipher text* atau pesan yang telah dienkripsi untuk kemudian dikembalikan menjadi pesan asli (*plain text*).

Salah satu algoritma kunci asimetris yang kuat dan aman adalah algoritma Elgamal. Salah satu kelebihan algoritma ini adalah *plain text* yang sama akan memberkan *cipher text* yang berbeda walau menggunakan kunci yang sama,

sehingga akan sngat membingungkan intruder atau penyadap. Algoritma ini yang akan digunakan pada penelitian ini untuk mengenkripsi kunci simetris sebelum dikirim ke penerima pesan.

3.3.3 Analisis Proses

Analisis penyelesaian masalah yang telah dijelaskan dapat dirangkum ke dalam diagram proses berikut ini:



Gambar 3.2 Diagram Proses Penyelesaian Masalah yang Diusulkan
Sumber: Oleh Penulis (2019)

Proses di atas terbagi menjadi dua buah area, yaitu area pengirim pesan dan area penerima pesan. Tahap pertama penerima membangkitkan sepasang

kunci yaitu *public key* dan *private key*, kemudian *public key* dikirimkan kepada pengirim pesan untuk digunakan mengenkripsi kunci simetris. Kemudian pengirim pesan membangkitkan kunci simetris sepanjang 128-*bits* secara acak dengan memberikan sebuah umpan bilangan acak. Kemudian pengirim pesan melakukan enkripsi terhadap *plain text* menggunakan kunci simetris dengan tekniks XOR pada mode operasi blok cipher ECB 128-*bits*. Hasil proses enkripsi ini menghasilkan *cipher text*. Kemudian kunci simetris dienkripsi menggunakan *public key* yang pengirim terima dari penerima pesan. Proses enkripsi kunci simetris menggunakan algoritma Elgamal sehingga dihasilkan *cipher key*. *Plain text* dan *cipher key* yang dihasilkan kemudian dikirimkan kepada penerima pesan. *Cipher text* dan *cipher key* ini bersifat tidak rahasia sehingga aman jika disadap.

Penerima pesan menerima *cipher text* dan *cipher key* dari pengirim pesan. Kemudian agar *cipher text* dapat didekripsi, maka penerima pesan harus mengetahui kunci simetris yang digunakan. Oleh karena itu, penerima pesan harus mendekripsi *cipher key* terlebih dahulu menjadi kunci simetris dengan menggunakan *private key* yang masih disimpan oleh penerima pesan. Proses dekripsi *cipher key* menggunakan algoritma Elgamal. Setelah *cipher key* berhasil didekripsi menjadi kunci simetris, maka *cipher text* kemudian didekripsi menggunakan kunci simetris dengan teknik XOR pada mode operasi blok cipher ECB 128-*bits* sehingga menghasilkan *plain text*.

Berdasarkan keterangan dari diagram proses di atas, maka diagram proses di atas terbagi menjadi 3 buah tahap proses utama, yaitu tahap pembangkitan kunci, tahap enkripsi, dan tahap dekripsi. Pada penjelasan berikut ini, akan

dilakukan proses perhitungan manual dari diagram proses di atas dimana *plain text* yang akan dienkripsi adalah:

Plain text = Hidup adalah Perjuangan

Panjang = 23 karakter = 184 *bits*

1. Tahap Pembangkitan Kunci

Tahap pembangkitan kunci terbagi menjadi dua buah bagian, yaitu pembangkitan kunci Elgamal dan Pembangkitan kunci ECB.

a. Pembangkitan Kunci Elgamal

Proses ini bertujuan untuk membangkitkan *public key* dan *private key*, yang digunakan pada proses enkripsi dan dekripsi menggunakan algoritma Elgamal. Proses ini dilakukan oleh penerima pesan, yang kemudian *public key* yang dihasilkan dikirimkan kepada pengirim pesan dan *private key* tetap disimpan oleh penerima pesan. Proses pembangkitan kunci Elgamal sebagai berikut:

1) Pilih bilangan prima yang besar secara acak. Lebih besar akan jauh lebih aman dan lebih kuat
2) Pilih dua buah bilangan acak, yaitu g dan x. dimana dengan syarat g < p dan $1 \leq x \leq p-2$
3) Cari nilai y dengan rumus $y = g^x \bmod p$

   Kemudian penerima data memberikan kunci enkripsi yaitu *Public key* = (y,g,p) dan menyimpan kunci dekripsi yaitu *Private key* = (x.p)..

Contoh proses:

1) Bilangan prima yang dipilih p = 84401.
2) Pilih dua buah bilangan acak, yaitu g dan x. dimana dengan syarat g < p dan $1 \leq x \leq p-2$.

   Bilangan acak g yang dipilih, g = 29325

   Bilangan acak X yang dipilih, x = 82911
3) Cari nilai y dengan rumus $y = g^x \bmod p$

   $y = g^x \bmod p$

   $y = 29325^{82911} \bmod 84401$

   y = 46791
4) Menentukan *Key* atau kunci

   Setelah didapatkan nilai p, g, x, dan y, maka dapat dibentuk *private key* dan *public key* Dimana

   Public key = (y,g,p), oleh karena itu maka:

   Public key = (46791, 29325, 84401)

   Private key = (x.p), oleh karena itu maka:

   Private key = (82911, 84401)

b. Pembangkitan Kunci Simetris ECB

Kunci simetris ECB digunakan untuk proses enkripsi da dekripsi pada mode operasi blok cipher 128-*bits* dengan teknik XOR. Kunci ini dibangkitkan sepanjang 128-*bits* secara acak. Algoritma pembangkit bilangan acak tidak dibahas pada penelitian ini, sehingga praktisi dapat menggunakan algoritma pembangkit bilangan acak apapun yang

diinginkan. Proses ini dilakukan oleh pengirim pesan. Misal kunci simetris ECB yang berhasil dibangkitkan adalah sebagai berikut:

Kunci Simetris ECB =

01001011001001110111011101010110001100111001001101010011010100110001011111011100001000001000000011100100000111110100110011101000

Kemudian konversi kunci simetris ECB ke dalam bentuk heksadesimal sehingga menjadi:

Kunci Simetris ECB = 4b2777563393535317dc2080e41f4ce8

2. Tahap Enkripsi

Tahap enkripsi terbagi menjadi dua buah bagian, yaitu enkripsi *plain text* dan enkripsi kunci simetris ECB.

a. Enkripsi *Plain text*

Misal, terdapat sebuah pesan atau *plain text* yang sangat rahasia yang akan dikirimkan via SMS (*Short Message Service*), maka pesan ini harus dienkripsi terlebih dahulu agar terjaga kerahasiaannya. Andai pesan tersebut adalah:

Plain text = Hidup adalah Perjuangan

Pesan tersebut memiliki:

Panjang Karakter = 23

Panjang Biner = 184

Kemudian konversi setiap karakter dari *plain text* ke dalam bentuk biner 8-*bits*, sehingga menjadi:

H = 01001000

i = 01101001

d = 01100100

u = 01110101

p = 01110000

<spasi> = 00100000

a = 01100001

d = 01100100

a = 01100001

l = 01101100

a = 01100001

h = 01101000

<spasi> = 00100000

P = 01010000

e = 01100101

r = 01110010

j = 01101010

u = 01110101

a = 01100001

n = 01101110

g = 01100111

a = 01100001

n = 01101110

Sehingga *plain text* menjadi:

Plain text =

01001000011010010110010001110101011100000010000001100001011001000110000101101100011000010110100000100000010100000110010101110010011010100111010101100001011011100110011101100001011011101101

Mengingat mode operasi blok cipher ECB yang dipilih beroperasi dengan panjang blok 128-*bits*, maka *plain text* dibagi ke dalam blok-blok berukuran 128-*bits*, dimana jika blok terakhir panjangnya kurang dari 128-*bits*, maka ditambahkan bit padding '1' sepanjang yang dibutuhkan sehingga panjang blok terakhir tetap 128-*bits*. Seperti contoh berikut ini:

Blok ke – 0

01001000011010010110010001110101011100000010000001100001011001000110000101101100011000010110100000100000010100000110010101110010

Blok ke – 1

01101010011101010110000101101110011001110110000101101110**1**

11111111111111111111111111111111111111111111111111111111111111111111111

72 *bits* '1' di atas merupakan bit padding (bit tambahan) agar blok terakhir tetap memiliki panjang 128-*bits*.

Kemudian lakukan proses enkripsi untuk masing-masing blok dengan kunci simetris yang sebelumnya berhasil dibangkitkan, yaitu:

Kunci Simetris ECB = 4b2777563393535317dc2080e41f4ce8

Kemudian konversi kunci simetris ECB ke dalam bentuk biner sehingga menjadi:

Kunci Simetris ECB =

0100101100100111011101110101011000110011100100110101001101010011000101111101110000010000001000000011100100000111111010011001110100

Proses enkripsi dari setiap blok dilakukan dengan teknik XOR terhhadap kunci simetris ECB. Proses tersebut dapat dilihat pada tabel berikut:

Tabel 3.1 Proses Enkripsi Plain text dengan Teknik XOR

Blok Pesan	***Plain text***	**Kunci Simetris ECB**	***Cipher text***
Blok ke – 0	010010000110100101 100100011101010111 000000100000011000 010110010001100001 011011000110000101 101000001000000101 000001100101011100 10	010010110010011101 110111010101100011 001110010011010100 110101001100010111 110111000010000010 000000111001000001 111101001100111010 00	000000110100111000 010011001000110100 001110110011001100 100011011101110110 101100000100000111 101000110001000100 111100101001100110 10
Blok ke – 1	011010100111010101 100001011011100110 011101100001011011 101111111111111111 111111111111111111 111111111111111111 111111111111111111 11	010010110010011101 110111010101100011 001110010011010100 110101001100010111 110111000010000010 000000111001000001 111101001100111010 00	001000010101001000 010110001110000101 010011110010001111 011010110011101000 001000111101111101 111111000110111110 000010110011000101 11

Sumber : Oleh Penulis (2019)

Kedua *cipher text* yang dihasilkan kemudian digabungkan sehingga menjadi:

Cipher text =

00000011010011100001001100100011010000111011001100100011011101110110101100000100000111101000110001000100111100101001100110100010000101010010000010110001110000101010011110010001111010110011100100001000111101111011111110001101111100000010011001100010111

Kemudian *cipher text* dikonversi ke dalam bentuk heksadesimal sehingga menjadi:

Cipher text =

034e132343b3323776b041e8c44f299a2152163854f23dace823df7f1be0b317

b. Enkripsi Kunci simetris ECB

Kunci simetris ECB yang dihasilkan kemudian dienkripsi dengan algoritma Elgamal menggunakan *public key* yang pengirim pesan peroleh dari penerima pesan. Kunci simetris ECB dienkripsi menjadi *cipher key*. *Public key* yang diterima pengirim pesan dari penerima pesan sebagai berikut:

Public key = (46791, 29325, 84401)

Sehingga berdasarkan *public key* di atas diketahui:

y = 46791

g = 29325

p = 84401

Kunci simetris ECB yang berhasil dibangkitkan pengirim pesan sebagai berikut:

Kunci Simetris ECB = 4b2777563393535317dc2080e41f4ce8

Bagi kunci simetris ECB ke dalam blok-blok masing-masing terdiri dari 4 karakter heksadesimal sehingga menjadi:

Kunci Simetris ECB = 4b27 7756 3393 5353 17dc 2080 e41f 4ce8

Sehingga setiap bloknya adalah:

Blok ke - 0 = 4b27

Blok ke - 1 = 7756

Blok ke - 2 = 3393

Blok ke - 3 = 5353

Blok ke - 4 = 17dc

Blok ke - 5 = 2080

Blok ke - 6 = e41f

Blok ke - 7 = 4ce8

Kemudian enkripsi setiap blok tersebut dengan algoritma Elgamal dengan persamaan berikut:

$a = g^k \bmod p$

$b = (y^k)(m) \bmod p$

Dimana k adalah bilangan acak dengan syarat $1 \leq k \leq p-2$. Nilai k harus selalu berbeda untuk setiap blok yang dienkripsi. Proses enkripsi pada kunci simetris ECB sebagai berikut:

Blok ke - 0

Sub plain *key* = 4b27

m_0 = 19239 (hasil konversi 4b27 ke dalam bentuk desimal)

k = 19861

a = g^k mod p = 29325^{19861} mod 84401 = 20028 =4e3c

b = $(y^k)(m)$ mod p = (46791^{19861}) 19239 mod 84401 = 28226 = 6e42

Blok ke - 1

Sub plain *key* = 7756

m_1 = 30550 (hasil konversi 7756 ke dalam bentuk desimal)

k = 77425

a = g^k mod p = 29325^{77425} mod 84401 = 4193 =1061

b = $(y^k)(m)$ mod p = (46791^{77425}) 30550 mod 84401 = 79280 = 135b0

Blok ke - 2

Sub plain *key* = 3393

m_2 = 13203 (hasil konversi 3393 ke dalam bentuk desimal)

k = 15302

a = g^k mod p = 29325^{15302} mod 84401 = 47511 =b997

b = $(y^k)(m)$ mod p = (46791^{15302}) 13203 mod 84401 = 49113 = bfd9

Blok ke - 3

Sub plain *key* = 5353

m_3 = 21331 (hasil konversi 5353 ke dalam bentuk desimal)

k = 36786

a = g^k mod p = 29325^{36786} mod 84401 = 57537 =e0c1

b = $(y^k)(m)$ mod p = (46791^{36786}) 21331 mod 84401 = 40562 = 9e72

Blok ke - 4

Sub plain *key* = 17dc

m_4 = 6108 (hasil konversi 17dc ke dalam bentuk desimal)

k = 82573

a = g^k mod p = 29325^{82573} mod 84401 = 56758 =ddb6

b = $(y^k)(m)$ mod p = (46791^{82573}) 6108 mod 84401 = 19688 = 4ce8

Blok ke - 5

Sub plain *key* = 2080

m_5 = 8320 (hasil konversi 2080 ke dalam bentuk desimal)

k = 81989

a = g^k mod p = 29325^{81989} mod 84401 = 75918 =1288e

b = $(y^k)(m)$ mod p = (46791^{81989}) 8320 mod 84401 = 15800 = 3db8

Blok ke - 6

Sub plain *key* = e41f

m_6 = 58399 (hasil konversi e41f ke dalam bentuk desimal)

k = 74965

a = g^k mod p = 29325^{74965} mod 84401 = 48344 =bcd8

b = (y^k)(m) mod p = (46791^{74965}) 58399 mod 84401 = 53682 = d1b2

Blok ke - 7

Sub plain *key* = 4ce8

m_7 = 19688 (hasil konversi 4ce8 ke dalam bentuk desimal)

k = 4343

a = g^k mod p = 29325^{4343} mod 84401 = 26415 =672f

b = (y^k)(m) mod p = (46791^{4343}) 19688 mod 84401 = 82073 = 14099

Gabungkan seluruh *cipher key* yang berhasil diperoleh, dimana bilangan a dan b dipisah dengan tanda koma (,) kemudian setiap blok *cipher key* dipisah dengan tanda titik (.) sehingga menjadi seperti berikut ini:

Cipher key =

4e3c,6e42.1061,135b0.b997,bfd9.e0c1,9e72.ddb6,4ce8.1288e,3db8.bcd8,d1b2.672f,14099.

3. Tahap Dekripsi

Proses dekripsi dilakukan oleh penerima pesan setelah penerima pesan menerima dua buah data dari pengirim pesan, yaitu *cipher text* dan *cipher key*. Tahap proses dekripsi terbagi menjadi dua bagian, yaitu tahap dekripsi *cipher key* dan tahap dekripsi *cipher text*.

a. Dekripsi *Cipher key*

Tahap pertama yang harus dilakukan penerima pesan adalah melakukan ekripsi terhadap *cipher text* yang diperoleh. Hal ini dilakukan karena *cipher text* tidak dapat didekripsi tanpa adanya kunci simetris ECB. Agar penerima pesan memiliki kunci simetris ECB yang maka *cipher key* harus didekripsi terlebih dahulu. Penerima pesan menerima *cipher key* sebagai berikut:

Cipher key =

4e3c,6e42.1061,135b0.b997,bfd9.e0c1,9e72.ddb6,4ce8.1288e,3db8.bcd8,d1b2.672f,14099.

Titik pada *cipher key* di atas merupakan pemisah antar blok *cipher key*, seperti yang ditunjukkan sebagai berikut:

Blok ke – 0 = 4e3c,6e42.

a = 4e3c dan b = 6e42.

Blok ke – 1 = 1061,135b0

a = 1061 dan b = 135b0

Blok ke – 2 = b997,bfd9

a = b997 dan b = bfd9

Blok ke – 3 = e0c1,9e72

a = e0c1 dan b = 9e72

Blok ke – 4 = ddb6,4ce8

a = ddb6 dan b = 4ce8

Blok ke – 5 = 1288e,3db8

a = 1288e dan b = 3db8

Blok ke – 6 = bcd8,d1b2

a = bcd8 dan b = d1b2

Blok ke – 7 = 672f,14099.

a = 672f dan b = 14099.

Proses dekripsi dilakukan terhadap setiap blok dari *cipher key* menggunakan *private key* yang masih disimpan oleh penerima pesan. *Private key* yang masih disimpan adalah sebagai berikut:

Private key = (82911, 84401)

Berdasarkan *private key* tersebut diketahui bahwa:

x = 82911

p = 84401

Proses dekripsi dilakukan dengan persamaan sebagai berikut:

$c = a^{p-1-x} \bmod p$

m = (b)(c) mod p

Hasil proses dekripsi kemudian dikonversikan ke dalam bentuk heksadesimal. Proses dekripsi *cipher key* dapat dilihat pada penjelasan berikut:

Blok ke - 0

a = 4e3c = 20028 (konversi dari heksadesimal ke desimal)

b = 6e42 = 28226 (konversi dari heksadesimal ke desimal)

$c = a^{p-1-x} \bmod p = 20028^{84401-1-82911} \bmod 84401 = 43780$

m_0 = (b)(c) mod p = (28226)(43780) mod 84401 = 19239 = 4b27

Blok ke - 1

a = 1061 = 4193 (konversi dari heksadesimal ke desimal)

b = 135b0 = 79280 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $4193^{84401-1-82911}$ mod 84401= 13723

m_1 = (b)(c) mod p = (79280)(13723) mod 84401 = 30550 = 7756

Blok ke - 2

a = b997 = 47511 (konversi dari heksadesimal ke desimal)

b = bfd9 = 49113 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $47511^{84401-1-82911}$ mod 84401= 37816

m_2 = (b)(c) mod p = (49113)(37816) mod 84401 = 13203 = 3393

Blok ke - 3

a = e0c1 = 57537 (konversi dari heksadesimal ke desimal)

b = 9e72 = 40562 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $57537^{84401-1-82911}$ mod 84401= 77537

m_3 = (b)(c) mod p = (40562)(77537) mod 84401 = 21331 = 5353

Blok ke - 4

a = ddb6 = 56758 (konversi dari heksadesimal ke desimal)

b = 4ce8 = 19688 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $56758^{84401-1-82911}$ mod 84401= 36302

m_4 = (b)(c) mod p = (19688)(36302) mod 84401 = 6108 = 17dc

Blok ke - 5

a = 1288e = 75918 (konversi dari heksadesimal ke desimal)

b = 3db8 = 15800 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $75918^{84401-1-82911}$ mod 84401= 67735

m_5 = (b)(c) mod p = (15800)(67735) mod 84401 = 8320 = 2080

Blok ke - 6

a = bcd8 = 48344 (konversi dari heksadesimal ke desimal)

b = d1b2 = 53682 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $48344^{84401-1-82911}$ mod 84401= 52962

m_6 = (b)(c) mod p = (53682)(52962) mod 84401 = 58399 = e41f

Blok ke - 7

a = 672f = 26415 (konversi dari heksadesimal ke desimal)

b = 14099 = 82073 (konversi dari heksadesimal ke desimal)

c = a^{p-1-x} mod p = $26415^{84401-1-82911}$ mod 84401= 18844

m_7 = (b)(c) mod p = (82073)(18844) mod 84401 = 19688 = 4ce8

Gabungkan seluruh hasil dekripsi yang diperoleh sehingga menjadi:

Kunci Simetris ECB = 4b2777563393535317dc2080e41f4ce8

b. Dekripsi *Cipher text*

Setelah kunci simetris ECB diperoleh melalui proses dekripsi *cipher key*, maka proses dekripsi *cipher text* dapat dilakukan. Kunci simetris ECB yang berhasil didekripsi dari *cipher text* sebagai berikut:

Kunci Simetris ECB = 4b2777563393535317dc2080e41f4ce8

Konversi setiap karakter dari heksadesimal apda kunci simetris ECB ke dalam bentuk biner 4 *bits* sebagai berikut:

4 = 100 = 0100

b = 1011 = 1011

2 = 10 = 0010

7 = 111 = 0111

7 = 111 = 0111

7 = 111 = 0111

5 = 101 = 0101

6 = 110 = 0110

3 = 11 = 0011

3 = 11 = 0011

9 = 1001 = 1001

3 = 11 = 0011

5 = 101 = 0101

3 = 11 = 0011

5 = 101 = 0101

3 = 11 = 0011

1 = 1 = 0001

7 = 111 = 0111

d = 1101 = 1101

c = 1100 = 1100

2 = 10 = 0010

0 = 0 = 0000

8 = 1000 = 1000

0 = 0 = 0000

e = 1110 = 1110

4 = 100 = 0100

1 = 1 = 0001

f = 1111 = 1111

4 = 100 = 0100

c = 1100 = 1100

e = 1110 = 1110

8 = 1000 = 1000

Gabungkan seluruh biner yang diperoleh sehingga menjadi:

Kunci Simetris ECB =

0100101100100111011101110101011000110011100100110101001101010011000101111101110000100000100000001110010000011111010011011101000

Cipher text yang diperoleh penerima pesan dari pengirim pesan adalah sebagai berikut:

Cipher text =

034e132343b3323776b041e8c44f299a2152163854f23dace823df7f1be0b317

Konversi setiap karakter heksadesimal dari *cipher text* ke dalam bentuk biner 4 *bits*, seperti berikut ini:

0 = 0 = 0000

3 = 11 = 0011

4 = 100 = 0100

e = 1110 = 1110

1 = 1 = 0001

3 = 11 = 0011

2 = 10 = 0010

3 = 11 = 0011

4 = 100 = 0100

3 = 11 = 0011

b = 1011 = 1011

3 = 11 = 0011

3 = 11 = 0011

2 = 10 = 0010

3 = 11 = 0011

7 = 111 = 0111

7 = 111 = 0111

6 = 110 = 0110

b = 1011 = 1011

0 = 0 = 0000

4 = 100 = 0100

1 = 1 = 0001

e = 1110 = 1110

8 = 1000 = 1000

c = 1100 = 1100

4 = 100 = 0100

4 = 100 = 0100

f = 1111 = 1111

2 = 10 = 0010

9 = 1001 = 1001

9 = 1001 = 1001

a = 1010 = 1010

2 = 10 = 0010

1 = 1 = 0001

5 = 101 = 0101

2 = 10 = 0010

1 = 1 = 0001

6 = 110 = 0110

3 = 11 = 0011

8 = 1000 = 1000

5 = 101 = 0101

4 = 100 = 0100

f = 1111 = 1111

2 = 10 = 0010

3 = 11 = 0011

d = 1101 = 1101

a = 1010 = 1010

c = 1100 = 1100

e = 1110 = 1110

8 = 1000 = 1000

2 = 10 = 0010

3 = 11 = 0011

d = 1101 = 1101

f = 1111 = 1111

7 = 111 = 0111

f = 1111 = 1111

1 = 1 = 0001

b = 1011 = 1011

e = 1110 = 1110

0 = 0 = 0000

b = 1011 = 1011

3 = 11 = 0011

1 = 1 = 0001

7 = 111 = 0111

Gabungkan seluruh biner yang diperoleh sehingga menjadi:

Cipher text =

00000011010011100001001100100011010000111011001100100

01101110111011010110000010000011110100011000100010011110 0

10100110011010001000010101001000010110001110000101010011 1

10010001111011010110011101000001000111101111101111111000 1

10111110000010110011000101 11

Lalu lakukan bagi *cipher text* ke dalam blok-blok dengan panjang 128 *bits* sebagai berikut:

Cipher text blok ke –0 =

00000011010011100001001100100011010000111011001100100

01101110111011010110000010000011110100011000100010011110 0

10100110011010

Cipher text blok ke – 1 =

00100001010100100001011000111000010101001111001000111101 1

01011001110100000100011110111110111111100011011111000001 0

11001100010111

Kemudian lakukan proses dekripsi dengan teknik XOR, seperti yang ditunjukkan pada tabel berikut:

Tabel 3.2 Proses Dekripsi Cipher text dengan Teknik XOR

Blok Pesan	***Cipher text***	**Kunci Simetris ECB**	***Plain text***
Blok ke – 0	000000110100111000 010011001000110100 001110110011001100 100011011101110110 101100000100000111 101000110001000100 111100101001100110 10	010010110010011101 110111010101100011 001110010011010100 110101001100010111 110111000010000010 000000111001000001 111101001100111010 00	010010000110100101 100100011101010111 000000100000011000 010110010001100001 011011000110000101 101000001000000101 000001100101011100 10
Blok ke – 1	001000010101001000 010110001110000101 010011110010001111 011010110011101000 001000111101111101 111111000110111110 000010110011000101 11	010010110010011101 110111010101100011 001110010011010100 110101001100010111 110111000010000010 000000111001000001 111101001100111010 00	011010100111010101 100001011011100110 011101100001011011 101111111111111111 111111111111111111 111111111111111111 111111111111111111 11

Sumber: Oleh Penulis (2019)

Gabungkan hasil dekripsi yang diperoleh sehingga *plain text* menjadi:

Plain text =

010010000110100101100100011101010111000000100000011000010

110010001100001011011000110000101101000001000000101000001

100101011100100110101001110101011000010110111001100111011

000010110111011111111111111111111111111111111111111111111

11111111111111111111111111111

Kemudian lakukan konversi setiap 8 *bits* dari *plain text* ke dalam bentuk karakter, namun jika terdapat 8 *bits* ‘1’ yang sama secara berurutan maka abaikan, seperti berikut:

01001000 = 72 = H

01101001 = 105 = i

01100100 = 100 = d

01110101 = 117 = u

01110000 = 112 = p

00100000 = 32 = <spasi>

01100001 = 97 = a

01100100 = 100 = d

01100001 = 97 = a

01101100 = 108 = l

01100001 = 97 = a

01101000 = 104 = h

00100000 = 32 = <spasi>

01010000 = 80 = P

01100101 = 101 = e

01110010 = 114 = r

01101010 = 106 = j

01110101 = 117 = u

01100001 = 97 = a

01101110 = 110 = n

01100111 = 103 = g

01100001 = 97 = a

01101110 = 110 = n

Gabungkan seluruh karakter yang diperoleh sehingga menjadi:

Plain text = Hidup adalah Perjuangan

3.3.4 Analisis Kebutuhan Sistem

Beberapa hal yang dibutuhkan untuk menjalankan sistem ini adalah :

1. ***Hardware***

   a. Perangkat Komputer

   Perangkat komputer yang meliputi : processor, RAM, harddisk, monitor, *mouse*, dan *keyboard* yang digunakan sebagai sistem komputer oleh aplikasi saat dijalankan dan juga digunakan untuk membangun aplikasi.

   b. Modem

   Modem digunakan sebagai perantara pengiriman SMS melalui aplikasi yang berhasil dibangun pada komputer. Sehingga pada modem ini lah terjadi pengiriman dan penerimaan pesan.

   c. Kartu GSM

   Kartu GSM diperlukan sebagai penyedia layanan SMS dari provider seperti telkomsel, indosat, atau XL.

   d. Dua buah laptop

   Dua buah laptop dibutuhkan sebagai pengirim dan penerima pesan dari aplikasi yang dibangun pada setiap masing-masing laptop yang mampu mengenkripsi pesan dan mendekripsi pesan yang dikirim maupun yang diterima oleh masing-masing pengguna. Kedua laptop harus menggunakan modem sebagai perantara pengirim dan penerima pesan SMS

2. *Software*

a. Sistem Operasi

Sistem operasi dibutuhakan sebagai *software* induk utama pada komputer. Sistem operasi yang dapat digunakan adalah Windows XP, 7, 8, dan 10. Atau juga bisa menggunakan Linux dengan berbagai varian distronya.

b. Gammu

Gammu digunakan sebagai *library* yang digunakan oleh aplikasi untuk melakukan pengiriman SMS. Versi gammu yang digunakan gammu 1.2.3

c. XAMPP

Sebuah aplikasi paket yang di dalamnya terdapat software PHP dan DBMS MySQL. PHP digunakan untuk membangun aplikasi yang berbasis *web*, sedangkan DBMS MySQL digunakan sebagai *database* yang akan menampung data-data pengiriman dan penerimaan pesan SMS.

d. Text Editor

Text editor digunakan untuk mengetikkan *source code* dari aplikasi atau pengkodingan dari aplikasi yang akan dibangun.

e. *Browser*

Browser digunakan untuk menjalankan aplikasi yang berbasis *web*. Dikarenakan aplikasi yang akan dibangun merupakan jenis aplikasi berbasis *web*, maka sebuah *browser* dibutuhkan untuk menjalankan aplikasi tersebut. *Browser* yang dapat digunakan adalah Mozilla Firefox, Google Chrome, Opera Mini, dan berbagai jenis *browser* lainnya.

3.4 Rancangan Penelitian

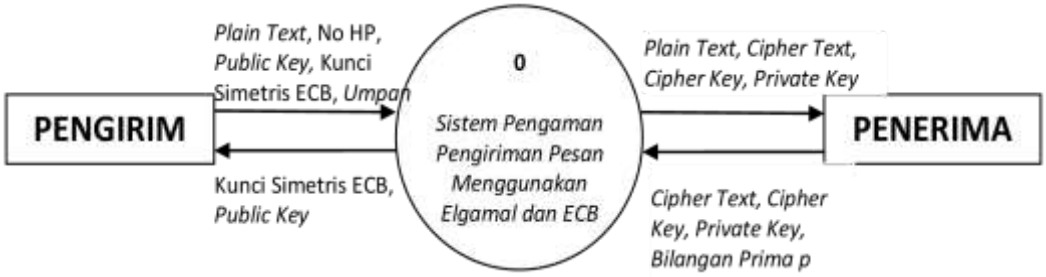
Rancangan penelitian dilakukan terhadap 3 objek, yaitu rancangan sistem, rancangan database, rancangan *interface*, dan *flowchart* proses.

3.4.1 Rancangan Sistem

Pemodelan sistem yang akan diberikan dalam bentuk diagram konteks dan DFD (*Data Flow Diagram*).

1. **Diagram Konteks**

Berikut merupakan diagram konteks dari sistem yang akan dibangun:

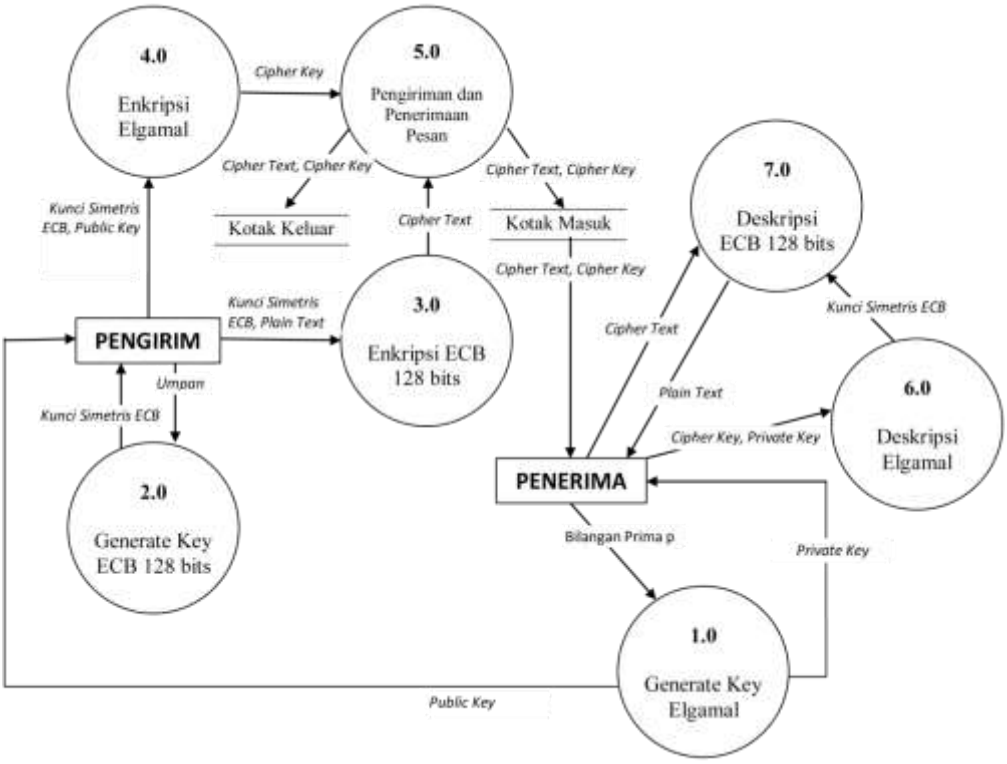


Gambar 3.3 Diagram Konteks dari Sistem
Sumber: Oleh Penulis (2019)

Pada diagram di atas, pengirim dapat memberikan *input* berupa *plain text*, nomor seluler, *public key*, kunci simetris ECB, dan umpan ke dalam sistem dan akan mendapatkan *output* dari sistem berupa kunci simetris ECB dan *public key*. sedangkan penerima pesan memberikan *input cipher text*, *cipher key*, *private key*, dan bilangan prima p ke dalam sistem, dan menerima *output* berupa *plain text*, *cipher text*, *cipher key*, dan *private key* dari sistem.

2. **Data Flow Diagram**

Berikut merupakan DFD Level 1 dari sistem yang dirancang:



Gambar 3.4 DFD Level 1 dari Sistem
Sumber: Oleh Penulis (2019)

Pada gambar diagram DFD diatas merupakan penjabaran dari diagram konteks dari sistem agar alur data dan proses yang ada menjadi lebih jelas dan mudah dipahami.

Pada DFD tersebut, terdapat tujuh buah proses, yaitu proses 1.0 (*generate key Elgamal*) yang berfungsi melakukan proses pembangkitan *private key*, dan juga *public key*. Pembangkitan kunci dilakukan oleh penerima, dengan memberikan bilangan prima p ke dalam proses, dan proses memberikan *output* berupa *private key* yang dikirim ke penerima, dan *public key* yang dikirim ke pengirim.

Proses 2.0 (*Generate Key* ECB 128 *Bits*) yang berfungsi untuk membangkitkan sebuah kunci sepanjang 128 *bits* secara acak. Proses ini

dilakukan oleh pengirim pesan dengan meng*input*kan sebuah umpan bilangan acak yang akan menghasilkan 128 *bits* deretan biner acak.

Proses 3.0 (enkripsi ECB 128 *bits*) berfungsi untuk melakukan proses enkripsi *plain text* dengan mode operasi ECB 128 *bits* dengan teknik XOR. Pengirim akan memberikan *input*-an ke dalam proses berupa *plain text* dan Kunci Simeteris ECB lalu hasil proses akan berupa *cipher text* dan akan dikirim ke proses 5.0 (pengiriman dan penerimaan pesan).

Proses 4.0 (enkripsi Elgamal) berfungsi untuk melakukan proses enkripsi terhadap Kunci Simetris ECB. Dimana pengirim akan memberikan *input*an ke dalam proses berupa Kunci Simetris ECB dan *public key*, lalu hasil proses ini menghasilkan sebuah *cipher key* yang kemudian dikirim ke proses 5.0 (pengiriman dan penerimaan pesan).

Proses 5.0 (pengiriman dan penerimaan pesan) berfungsi untuk mengirimkan pesan yang akan dikirim dan menerima pesan yang masuk. Pesan yang dikirim berupa *cipher text* dan *cipher key* akan disimpan di tabel kotak keluar dan dikirim ke penerima, sedangkan pesan masuk akan disimpan di tabel kotak masuk.

Proses 6.0 (deskripsi Elgamal) berfungsi untuk melakukan proses deskripsi tehadap *cipher key* yang diterima. Penerima pesan memberikan *input*an berupa *cipher key* dan *private key* ke dalam proses dan proses akan memberikan hasil berupa kunci simetris ECB. Kunci Simetris ECB kemudian dikirimkan ke proses 7.0 (dekripsi ECB 128 *bits*).

Proses 7.0 (dekripsi ECB 128 *bits*) berfungsi untuk melakukan proses dekripsi terhadap *cipher text* menjadi *plain text*. Penerima pesan memberikan *input*-an berupa *cipher text* dan kunci simeteris ECB ke dalam proses dan proses akan memberikan output berupa *plain text* ke penerima pesan.

3.4.2 Rancangan Database

1. Struktur Tabel

Berikut adalah struktur tabel yang digunakan untuk *database* aplikasi :

a. Tabel Kotak Keluar

Struktur tabel kotak keluar yang digunakan dalah sebagai berikut :

Tabel 3.3 Struktur Tabel Kotak Keluar

No	Field	Type	Keterangan
1	kode_kotak_keluar	bigint(20)	Kode untuk setiap pesan keluar
2	isi_pesan	text	Isi pesan terenkripsi
3	tanggal_pesan	datetime	Tanggal pesan dikirim
4	no_pengirim	varchar(15)	Nomo seluler pengirim pesan

Sumber : Oleh Penulis (2019)

b. Tabel Kotak Masuk

Struktur tabel kotak masuk yang digunakan adalah sebagai berikut:

Tabel 3.4 Struktur Tabel Kotak Masuk

No	Field	Type	Keterangan
1	kode_kotak_masuk	bigint(20)	Kode untuk setiap pesan masuk
2	isi_pesan	Text	Isi pesan terenkripsi
3	tanggal_pesan	Datetime	Tanggal pesan dikirim
4	no_penerima	varchar(15)	Nomor seluler penerima pesan
5	status	varchar(6)	Status pesan apakah telah dibaca atau belum

Sumber : Oleh Penulis (2019)

c. Tabel Kunci

Tabel ini akan digunakan untuk menyimpan kunci yang berhasil dibangkitkan. Struktur tabel kunci adalah sebagai berikut:

Tabel 3.5 Struktur Tabel Kunci

No	Field	Type	Keterangan
1	kode_kunci	int(11)	Kode untuk setiap pasangan kunci yang dibangkitkan
2	*public_key*	varchar(30)	*Public key* yang dihasilkan
3	*private_key*	varchar(30)	*Private key* yang dihasilkan
4	no_penerima	varchar(15)	Nomor seluler penerima pesan

Sumber : Oleh Penulis (2019)

d. Tabel Admin

Tabel ini akan digunakan untuk menyimpan data admin yang digunakan untuk *login* ke dalam aplikasi.

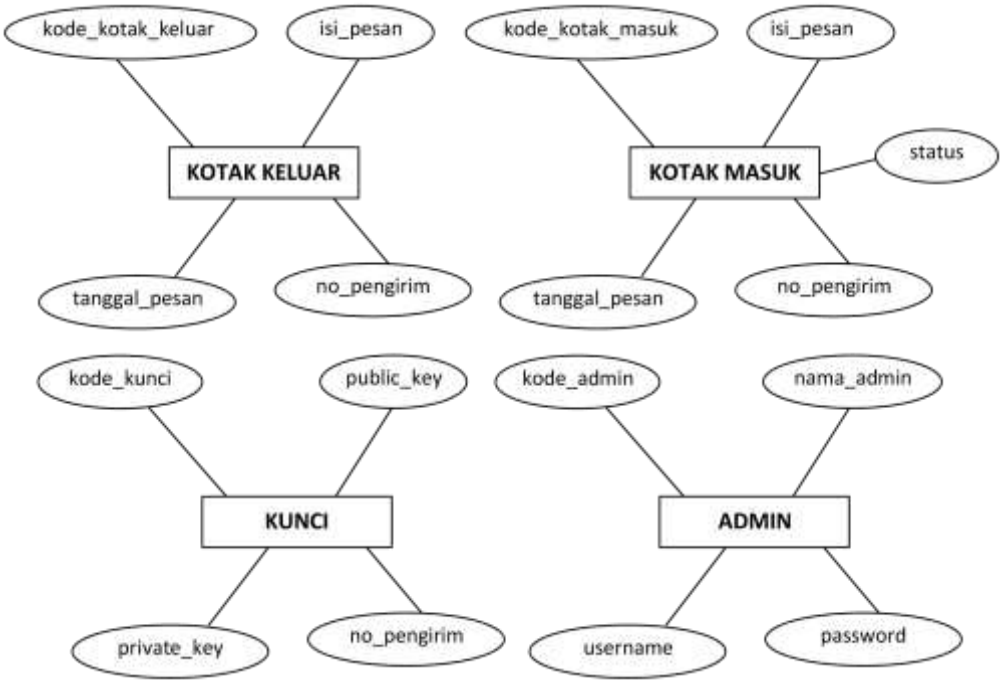
Tabel 3.6 Struktur Tabel Admin

No	Field	Type	Keterangan
1	kode_admin	int(11)	Kode unik untuk admin
2	nama_admin	varchar(50)	Nama lengkap dari admin
3	*username*	varchar(50)	*Username* untuk login
4	*password*	varchar(50)	*Password* untuk login

Sumber : Oleh Penulis (2019)

2. *Entity Relatipnship Diagram* (ERD)

ERD (*Entity Relationship Diagram*) dari rancangan *database* yang digunakan untuk aplikasi adalah sebagai berikut:



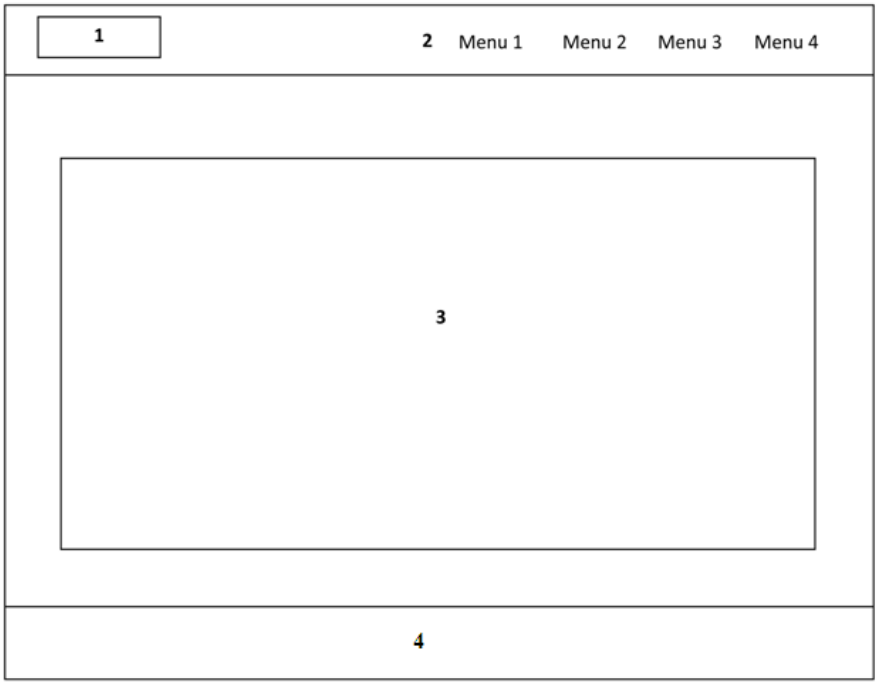
Gambar 3.5 ERD (Entity Relationship Diagram) dari Sistem
Sumber: Oleh Penulis (2019)

Pada diagram ERD di atas dapat terlihat, terdapat 4 buah entity pada rancangan database yang akan digunakan oleh aplikasi. Kempat buah entity tersebut tidak pernah berelasi satu dengan yang lainnya, sehingga setiap entity menyimpan data secara tunggal, dan tidak bergantung kepada entity lainnya, serta tidak terdapat relasi antara suatu entity dengan entity lainnya.

3.4.3 Rancangan Interface

1. Rancangan Beranda

Berikut merupakan tampilan utama atau beranda dari aplikasi :



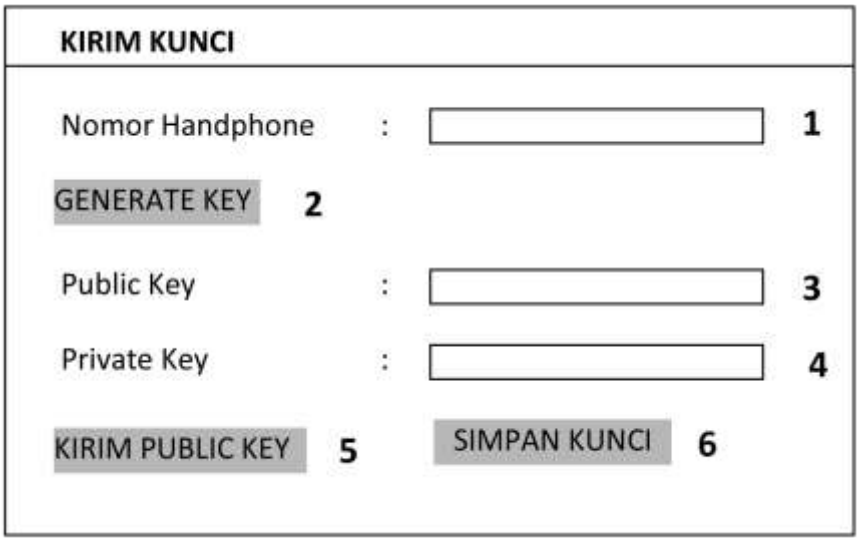
Gambar 3.6 Tampilan Utama Aplikasi
Sumber: Oleh Penulis (2019)

Keterangan dari gambar diatas adalah :

1) Logo Kampus
2) Berbagai menu yang tersedia, seperti pesan, kotak masuk, kotak keluar, enkripsi, dan deskripsi.
3) Area utama dimana form pesan dan tabel data ditampilkan
4) Footer, digunakan untuk informasi tahun pembuatan dan progammer.

2. Rancangan *Form Generate Key*

Berikut merupakan *form generate key* atau pembangkitan kunci pada aplikasi:



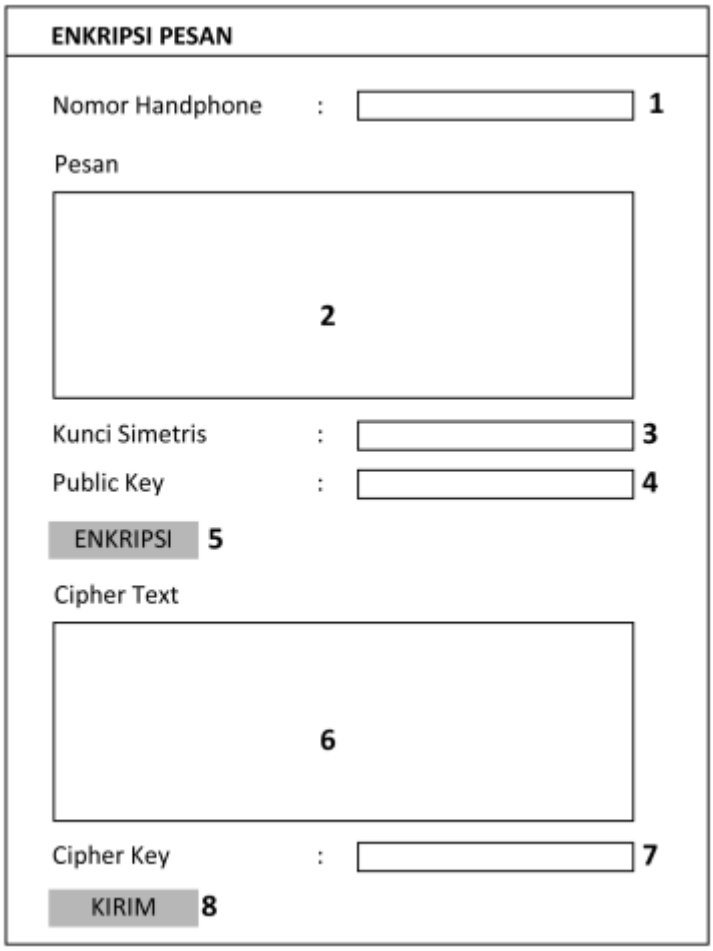
Gambar 3.7 Interface Form Generate Key
Sumber: Oleh Penulis (2019)

Keterangan dari gambar diatas adalah :

1) *Textfield* pengirian nomor selular atau nomor handphone
2) Tombol untuk meng-*generate key*
3) *Textfield* tempat dimana *public key* akan diletakkan
4) *Textfield* tempat dimana *private key* akan diletakkan
5) Tombol untuk mengirim *public key* ke nomor tujuan
6) Tombol untuk menyimpan kunci yang berhasil dibangkitkan

3. Rancangan *Form* Enkripsi

Rancangan interface ini digunakan untuk melakukan enkripsi pesan adalah sebagai berikut:



Gambar 3.8 Interface Enkripsi Pesan
Sumber: Oleh Penulis (2019)

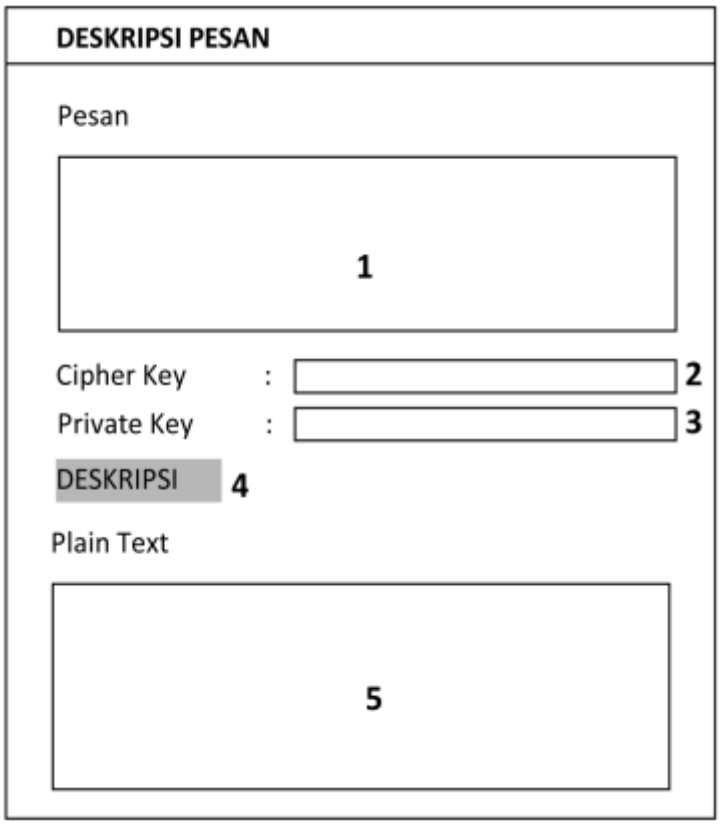
Keterangan dari gambar diatas adalah :

1) *Textfield* pengirian nomor selular atau nomor handphone
2) *Textarea* tempat pesan diketikan
3) *Textfield* tempat dimana kunci simetris ECB otomatis akan diletakkan
4) *Textfield* tempat dimana *public key* akan diletakkan
5) Tombol untuk melakukan enkripsi pesan
6) *Textarea cipher text* hasil enkripsi diletakkan
7) *Textfield* tempat dimana *cipher key* hasil enkripsi kunci simetris ECB diletakkan.

8) Tombol untuk mengirim *cipher text* ke tujuan.

4. Rancangan *Form* Dekripsi

Rancangan interface ini akan digunakan aplikasi untuk melakukan deskripsi pesan.



Gambar 3.9 Interface Deskripsi Pesan
Sumber: Oleh Penulis (2019)

Keterangan dari interface di atas adalah sebagai berikut:

1) *Textarea* tempat pesan yang diterima

2) *Textfield* tempat dimana *cipher key* akan diletakkan

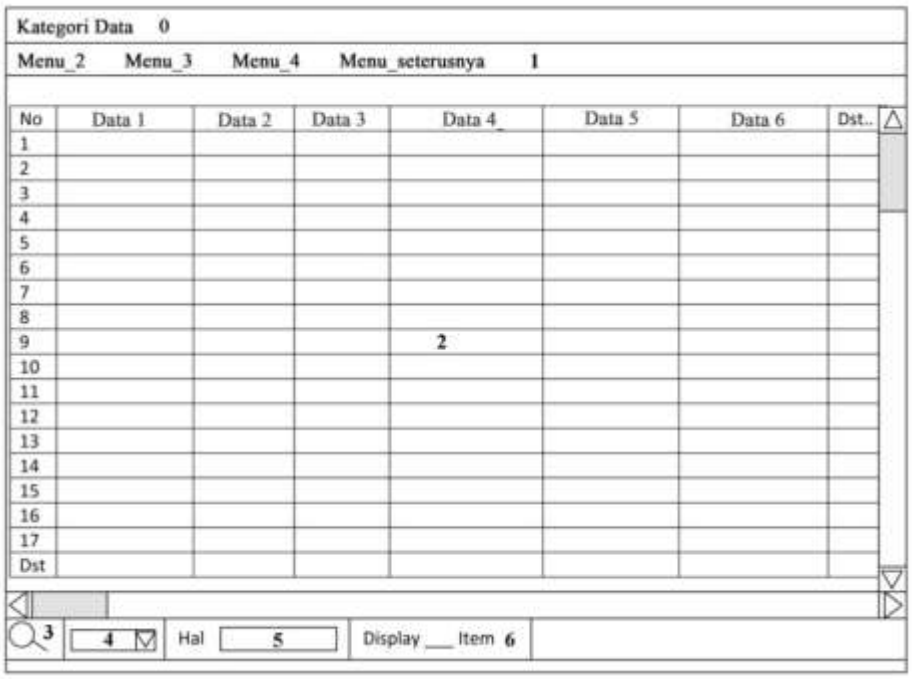
3) *Textfield* tempat dimana *private key* akan diletakkan

4) Tombol untuk melakukan deskripsi pesan

5) *Textarea* dimana *cipher text* pesan akan ditampilkan.

5. Rancangan Penyimpanan SMS

Rancangan ini digunakan untuk menampilkan seluruh data-data yang pesan yang diterima dan pesan yang dikirim serta kunci yang tersimpan.



Gambar 3.10 Interface Penyimpanan Pesan
Sumber: Oleh Penulis (2019)

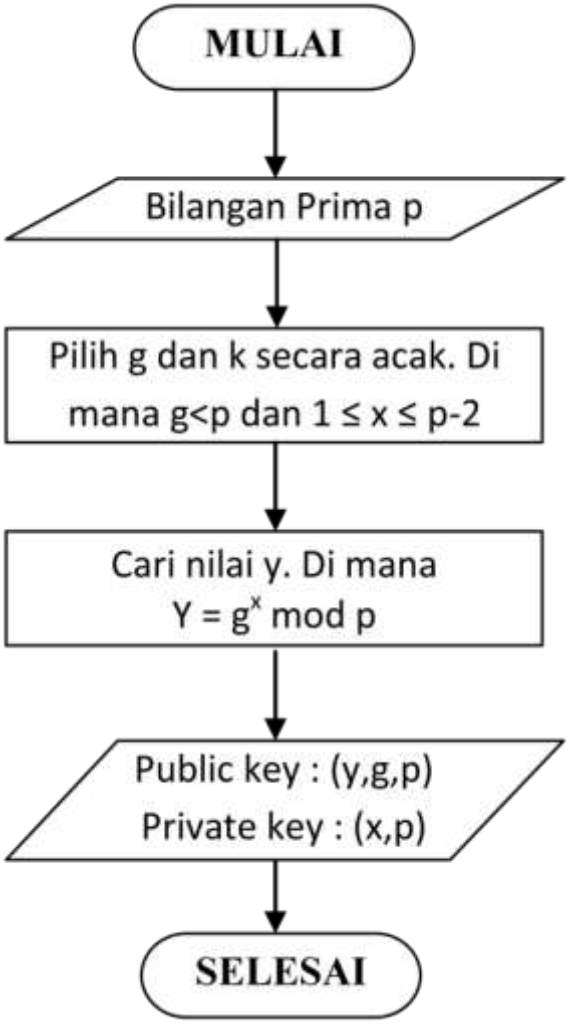
Keterangan dari gambar diatas adalah sebagai berikut :

1) Nama kategori dari data.
2) Daftar menu untuk *user* melakukan manipulasi data.
3) Tempat dimana data-data yang ada akan ditampilkan.
4) Menu untuk mencari data berdasarkan *field* dan kata kunci tertentu.
5) Menu untuk menampilkan jumlah baris data yang akan ditampilkan.
6) Keterangan halaman yang sedang ditampilkan.
7) Keterangan banyaknya jumlah data yang ada.

3.4.4 Flowchart Proses

1. Flowchart Pembangkitan Kunci Elgamal

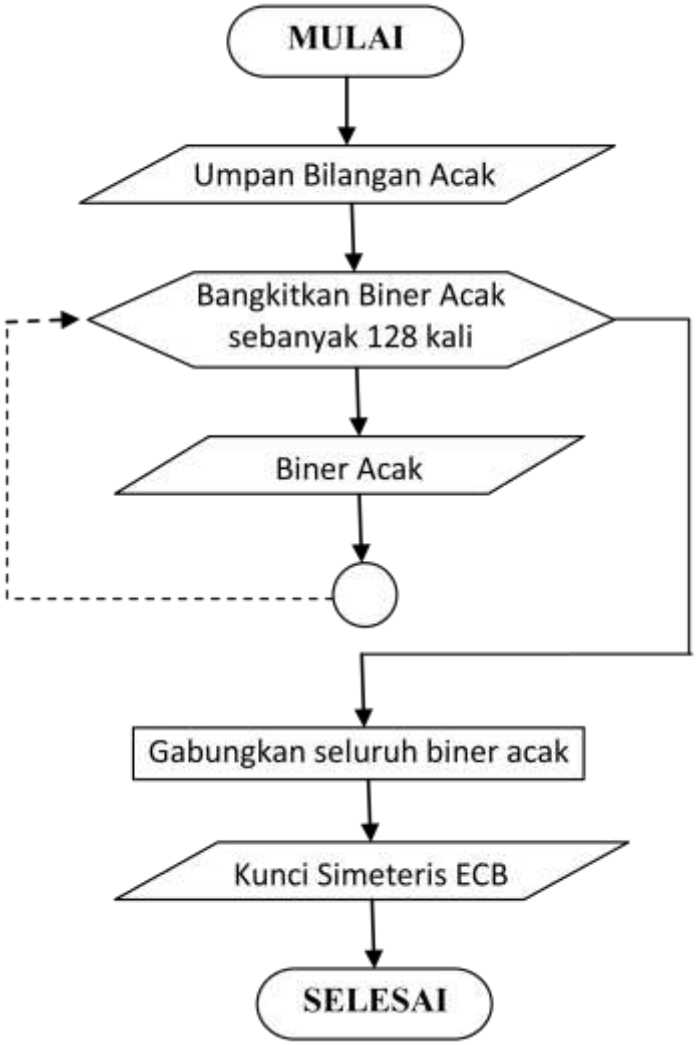
Flowchart ini menggambarkan proses *generate key* Elgamal yang terjadi.



Gambar 3.11 Flowchart Generate Key Elgamal
Sumber: Oleh Penulis (2019)

2. Flowchart *Generate Key* Simetris ECB

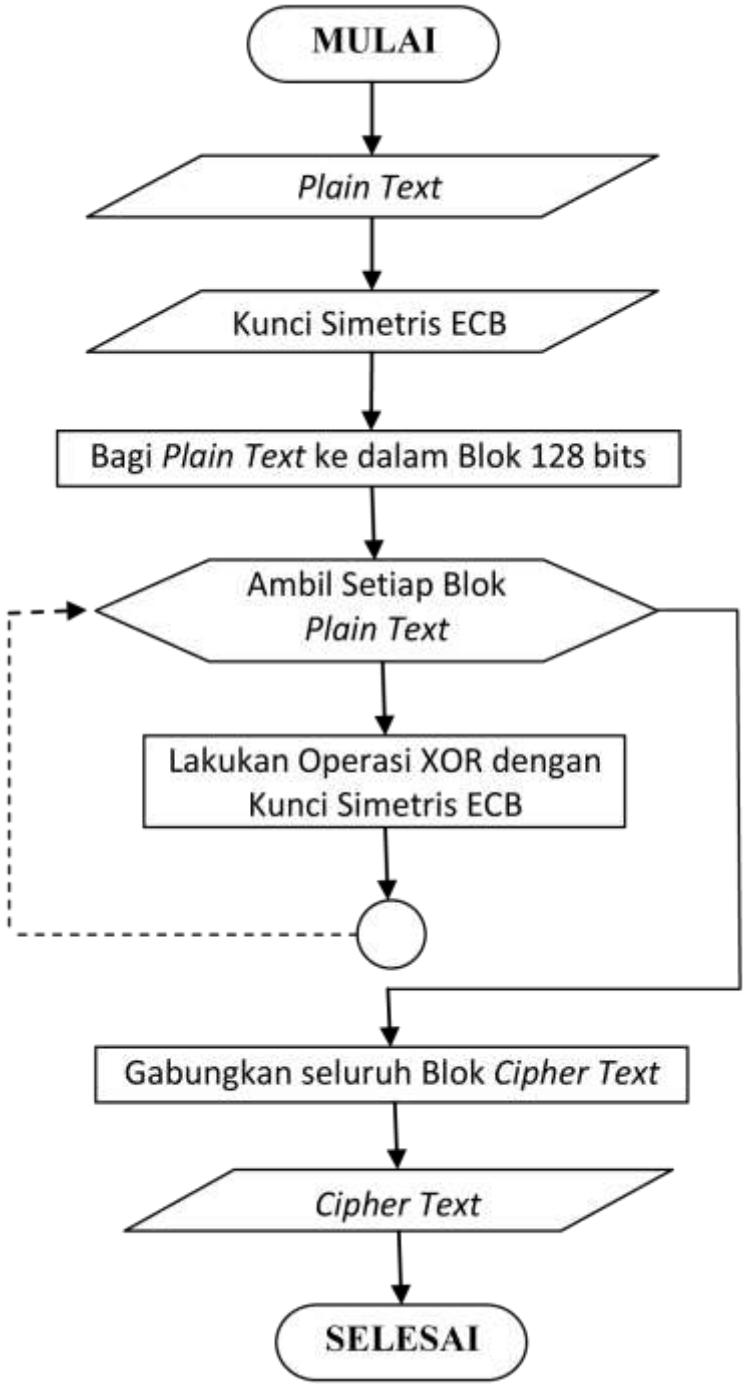
Flowchart berikut menggambarkan proses dari pembangkitan kunci simetris ECB.



Gambar 3.12 Flowchart Generate Key Simetris ECB
Sumber: Oleh Penulis (2019)

3. Flowchart Enkripsi dengan Mode ECB

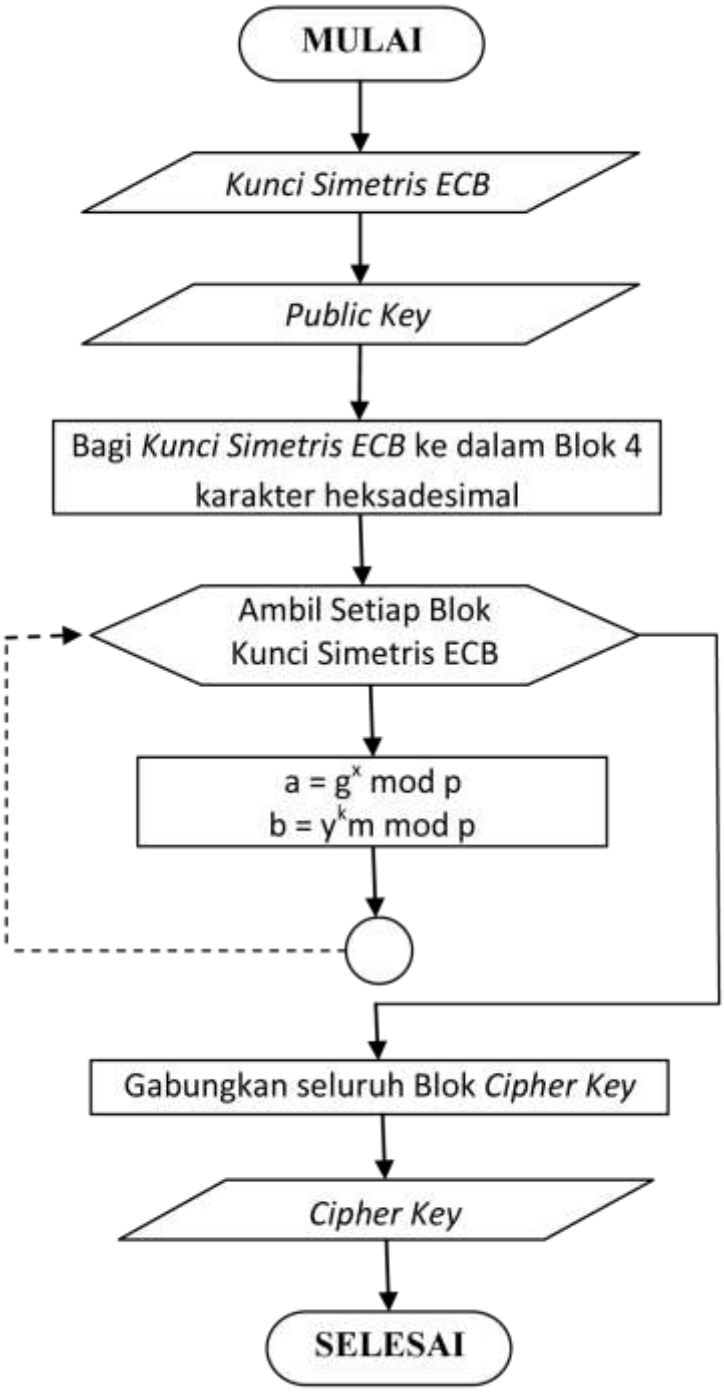
Flowchart berikut menggambarkan proses pengenkripsian *plain text* menjadi *cipher text*.



Gambar 3.13 Flowchart Enkripsi dengan Mode ECB
Sumber: Oleh Penulis (2019)

4. Flowchart Enkripsi dengan Algoritma Elgamal

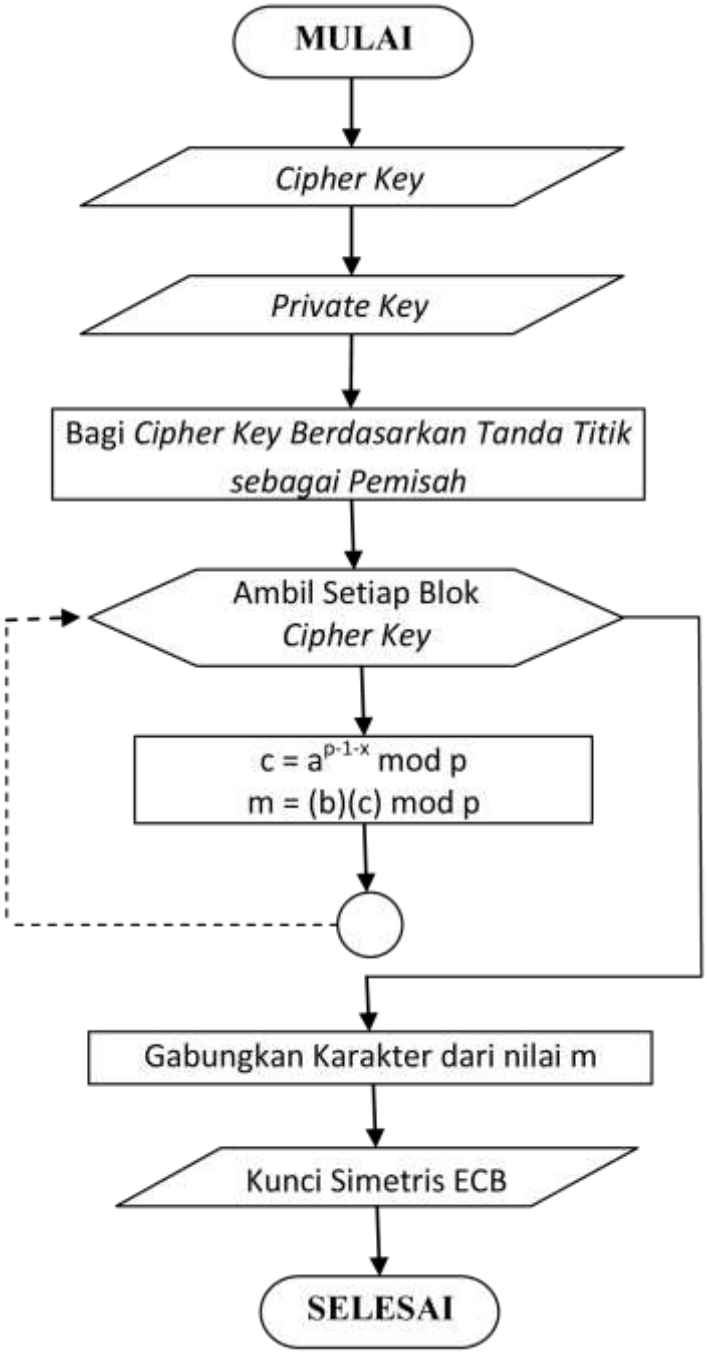
Flowchart berikut menggambarkan proses pengenkripsian kunci simetris ECB menjadi *cipher key*.



Gambar 3.14 Flowchart Enkripsi dengan Algoritma Elgamal
Sumber: Oleh Penulis (2019)

5. Flowchart Dekripsi dengan Algoritma Elgamal

Flowchart berikut menggambarkan proses pendekripsian *cipher key* menjadi kunci simetris ECB.

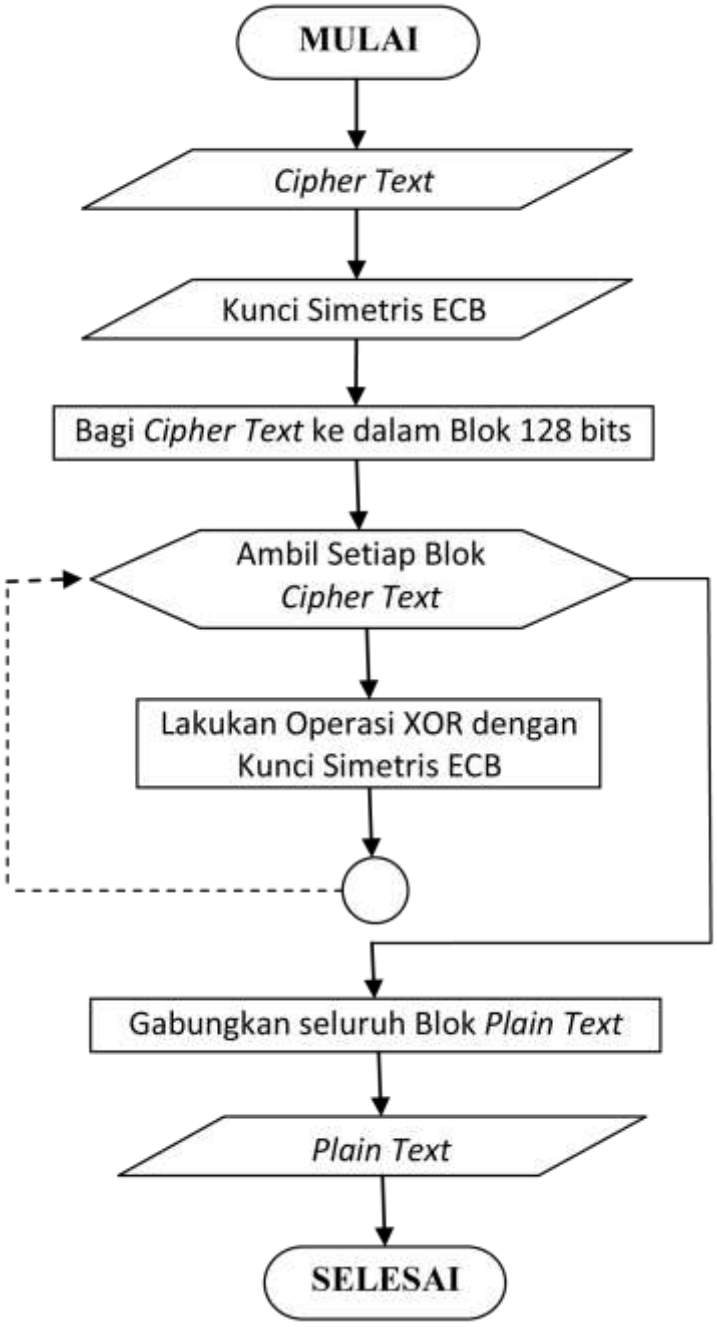


Gambar 3.15 Flowchart Dekripsi dengan Algoritma Elgamal

Sumber: Oleh Penulis (2019)

6. Flowchart Dekripsi dengan Mode ECB

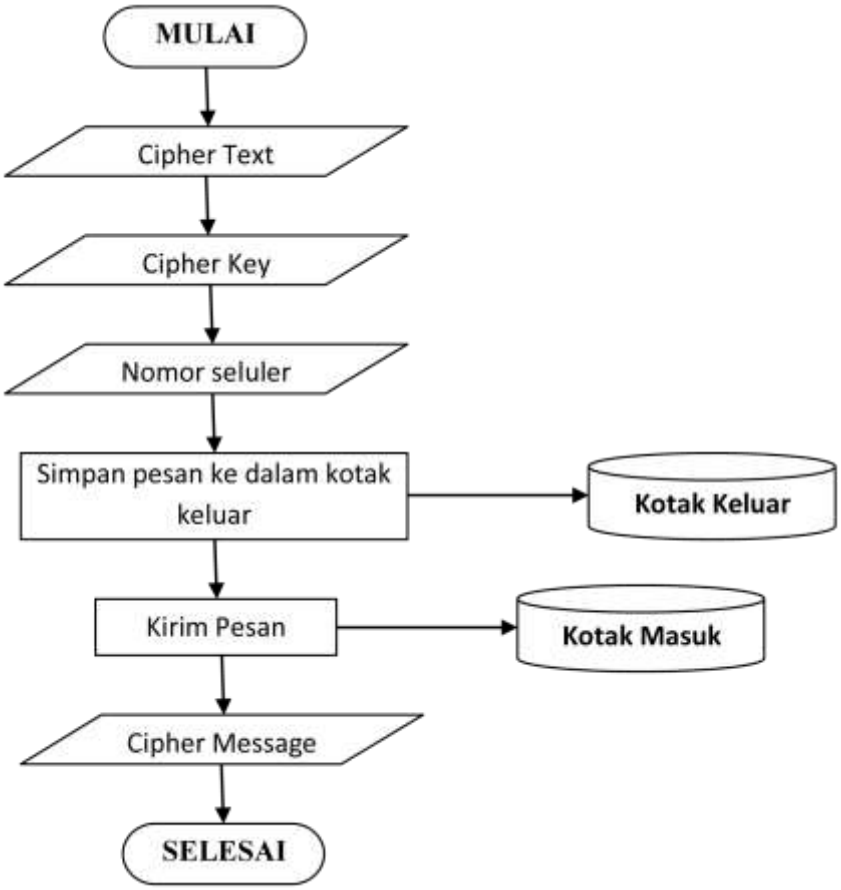
Flowchart berikut menggambarkan proses pendekripsian *cipher text* menjadi *plain text*.



Gambar 3.16 Flowchart Dekripsi dengan Mode ECB
Sumber: Oleh Penulis (2019)

7. Flowchart Pengiriman Pesan Melalui SMS

Flowchart berikut menggambarkan proses pengiriman pesan melalui SMS.



Gambar 3.17 Flowchart Pengiriman Pesan Melalui SMS
Sumber: Oleh Penulis (2019)

BAB II
LANDASAN TEORI

2.1 *Short Message Service* (SMS)

Short Message Service yang lebih dikenal dengan sebutan SMS merupakan sebuah teknologi yang memungkinkan untuk menerima maupun mengirimkan pesan antar telepon bergerak atau ponsel. Mekanisme cara kerja sistem *Short Message Service* (SMS) adalah melakukan pengiriman pesan singkat dari satu terminal ke terminal yang lain. *Short Message Service* (SMS) digunakan sebagai media penyampai pesan atau perantara dari user ke modem wavecom. Yang berfungsi sebagai penerima pesan (Ardiansyah, et al, 2015).

2.1.1 Keuntungan Memakai SMS

Adapun keuntungan dari menggunakan *Short Message Service* (SMS) dapat dikatakan yang paling sukses di dunia. Hal ini terbukti dengan jumlah transaksi *Short Message Service* (SMS) yang "Ter Besar" dalam jumlah perhari saja. Apakah yang membuat *Short Message Service* (SMS) menjadi special dan menjadikan *Short Message Service* (SMS) begitu popular didunia (Sunardi, et al,2009) ?

1. *Short Message Service* (SMS) dapat dikirim dan dibaca kapanpun
2. *Short Message Service* (SMS) dapat dikirimkan pada *Offline Mobile Phone*
3. *Short Message Service* (SMS) tidak begitu mengganggu

4. *Short Message Servic)* (SMS) didukung penuh oleh antar *GSM* dan *Wireless* lainnya

2.1.2 *Short Message Service Center* (SMSC)

Short Message Service Center (SMSC) adalah yang bertanggung jawab atas operasi *Short Message Service* (SMS) dari sebuah nirkabel. Ketika *Short Message Servic* (SMS) dikirim, penerima tidak menerima secara langsung *Short Message Service* (SMS) tersebut, namun *Short Message Service* (SMS) akan diterima oleh *Short Message Service Center* (SMSC) terlebih dahulu kemudian diteruskan kepada penerima. Hal ini akan memberikan kenyamanan si pengirim, dalam arti saat penerima sedang tidak aktif, *Short Message Service* (SMS) yang dikirimkan akan terus di coba untuk disampaikan ke penerima oleh *Short Message Service Center* (SMSC) selama periode aktif (Sunardi, et al,2009)

2.2 Kriftografi

2.2.1 Pengertian Kriftografi

Kemajuan dan perkembangan teknologi informasi saat ini telah berpengaruh pada hampir semua aspek kehidupan manusia, tidak terkecuali dalam hal berkomunikasi. sehingga membuat setiap elemen masyarakat harus lebih menjamin tentang keamana berkomunikasi yang mereka lakukan seperti keamanan dalam mengaman kan suatu pesan *Short Message Service* (SMS) untuk melindungi privasi atau hal yang dianggap rahasia. Untuk melakukan keamanan pesan *Short Message Service* (SMS) agar dapat melindungi hal yang

dianggap rahasia, maka yang diperlukan adalah dengan menyandikan isi pesan *Short Message Service* (SMS) menjadi suatu kode-kode yang tidak dimengerti sehingga apabila disadap maka akan kesulitan untuk mengetahui isi informasi *Short Message Service* (SMS) yang sebenarnya, dan hal yang diperlukan yaitu kriftografi.

Kriftografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikannya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya. Dalam ilmu kriptografi, terdapat dua buah proses yaitu melakukan enkripsi dan dekripsi. Pesan yang akan dienkripsi disebut sebagai teks biasa atau *plaintext*. Disebut demikian karena informasi ini dengan mudah dapat dibaca dan dipahami oleh siapa saja. Algoritma yang dipakai untuk mengenkripsi dan mendekripsi sebuah *plaintext* melibatkan penggunaan suatu bentuk kunci. Pesan *plaintext* yang telah dienkripsi atau dikodekan dikenal sebagai *ciphertext* (Pabokory, et al,2015).

Dalam jurnal yang berjudul "Implementasi Kriptografi Pengamanan Data Pada Pesan Teks, Isi File Dokumen, Dan File Dokumen Menggunakan Algoritma Advanced Encryption Standard*"* juga menjelaskan bahwa dalam kriptografi kita akan sering menemukan berbagai istilah atau *terminology* (Pabokory, et al,2015). Beberapa istilah yang harus diketahui yaitu:

1. Pesan, *Plainteks*, dan *Cipherteks*

Pesan *(message)* adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan adalah *plaintex)* atau teks jelas *cleartext*.

2. Pengirim dan Penerima

   Komunikasi data melibatkan pertukaran pesan antara dua entitas. Pengirim *(sender)* adalah entitas yang mengirim pesan kepada entitas lainnya. Penerima *(receiver)* adalah entitas yang menerima pesan.

3. Enkripsi dan dekripsi

   Proses menyandikan *plainteks* menjadi *cipherteks* disebut enkripsi *(encryption)* atau *enciphering* (standard nama menurut ISO 7498-2). Sedangkan proses mengembalikan cipherteks menjadi plainteks semula disebut dekripsi *(decryption)* atau *deciphering* (standard nama menurut ISO 7498-2).

4. *Cipher* dan kunci

   Algoritma kriptogarfi disebut juga *cipher*, yaitu aturan untuk enkripsi dan dekripsi, atau fungsi matematika yang digunakan untuk enkripsi dan dekripsi. Beberapa *cipher* memerlukan algoritma yang berbeda untuk enkripsi dan dekripsi. Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara dua buah himpunan yang berisi elemen-elemen plainteks dan himpunan yang berisi *cipherteks*. Enkripsi dan dekripsi merupakan fungsi yang

memetakan elemen-elemen antara dua himpunan tersebut. Misalnya P menyatakan *plainteks* dan C menyatakan *cipherteks*, maka :

E(P) = C fungsi enkripsi E memetakan P ke C . . . . . (1)

D(C) = P fungsi dekripsi D memetakan C ke P . . . . . .(2)

Karena proses enkripsi kemudian dekripsi mengembalikan pesan ke pesan asal, maka persamaan D(E(P)) = P harus benar.

2.2.2 Tujuan Kriftografi

Ada empat tujuan yang mendasar dari ilmu kriptograpi yang juga merupakan aspek keamanan informasi yaitu (Efrandi, et al, 2014):

1. Kerahasiaan (*Secrecy*)

   Merupakan layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka atu mengupas informasi yang telah disandi. Integritas data adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah.

2. integritas data (*Integrity date*)

   sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak- pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubtitusian data lain kedalam data yang sebenarnya.

3. Autentikasi (*Authentication*)

   Merupakan yang berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasiitu sendiri. Dua pihak yang

saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian isi datanya, waktu pengiriman dan lain-lain.

4. Non- repudiasi atau penyangkalan

   Merupakan usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau yang membuat.

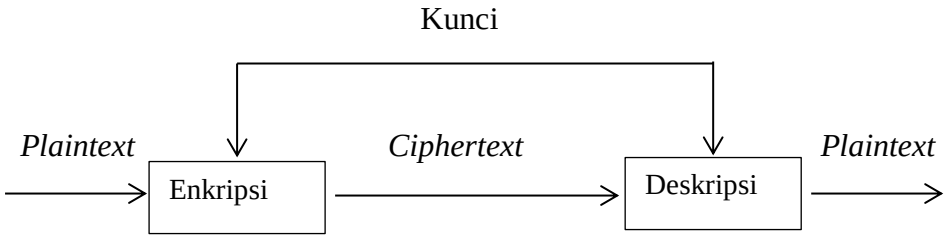
2.2.3 Komponen Kriftografi

1. *Plaintext* yaitu pesan yang dapat dibaca.
2. *Ciphertext* yaitu pesan sandi/pesan acak yang tidak bisa dibaca.
3. *Key* yaitu kunci untuk melakukan teknik kriptografi.
4. Algoritma yaitu metode untuk melakukan enkripsi dan dekrips

2.2.4 Algoritma Kriptografi Kunci Simetris

Algoritma simetris (algoritma kriptografi konvensional) adalah algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan proses deskripsi. Algoritma kriptografi simetris dibagi menjadi dua kategori yaitu algoritma aliran *(Stream Ciphers)* dan algoritma blok *(Block Ciphers)*. Dimana pada algoritma aliran, proses penyandiannya akan beriorientasi pada satu bit atau byte data. Sedangkan pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau byte data (per blok). Adapun contoh algoritma kunci simetris adalah *Data Encryption Standard* (DES), Blowfish, Twofish, MARS, IDEA,

3DES (DES diaplikasikan 3 kali), *Advanced Encryption Standard* (AES) yang bernama asli Rijndael (Irham Mu'alimin Arrijal, et al,2016)



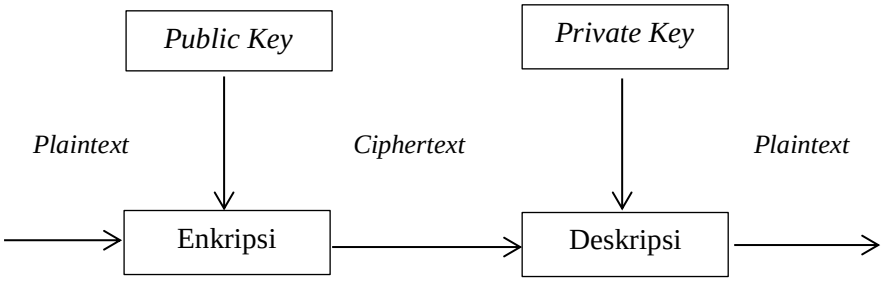
Gambar 2.1 Skema Sistem Kriptografi Kunci Simetris
Sumber : Suhardi (2016)

1. Kelebihan Algoritma Kriptografi Kunci Simetris :
   a. Masalah keamanan pada distribusi kunci dapat lebih baik
   b. Masalah manajemen kunci yang lebih baik karena jumlah kunci yang lebih sedikit
2. Kelemahan Algoritma Kriptografi Kunci Simetris:
   a. Kecepatan yang lebih rendah bila dibandingkan dengan algoritma simetris
   b. Untuk tingkat keamanan sama, kunci yang digunakan lebih panjang dibandingkan dengan algoritma simetris.

2.2.5 Kriptografi Kunci Publik (Asimetris)

Chandra (2016) kriptografi kunci publik adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma asimetris karena kunci untuk enkripsi dibuat

umum *(public key)* atau dapat diketahui oleh setiap orang, tapi kunci untuk dekripsi hanya diketahui oleh orang yang berwenang, yang mengetahui data yang disandikan atau sering disebut kunci pribadi *(private key)*. Contoh algoritma terkenal yang menggunakan kunci publik adalah *Riverst Shamir Adleman* (RSA) dan *Elliptic Curve Cryptography (ECC)*.



Gambar 2.2 Skema Algoritma Kriptografi Kunci Asimetris
Sember : Suhardi (2016)

2.3 Elektronic Code Book (ECB)

2.3.1 Pengertian Elektronic Code Book (ECB)

Cipher blok merupakan algoritma kriptografi yang beroperasi dalam bentuk blok bit. Proses enkripsi dilakukan terhadap blok bit *plainteks* dengan menggunakan kunci yang berukuran sama dengan ukuran blok *plainteks*. Algoritma ini akan menghasilkan *cipherteks* yang sama panjang dengan blok *plainteks*. Proses dekripsi terhadap *cipherteks* berlangsung dengan cara serupa seperti enkripsi. Hanya saja pada proses dekripsi, operasi berjalan kebalikan dari proses enkripsi (Chumaidi Rahma, et al, 2009).

Proses enkripsi dengan kunci K dinyatakan secara formal dengan persamaan

$$EK(P) = C \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

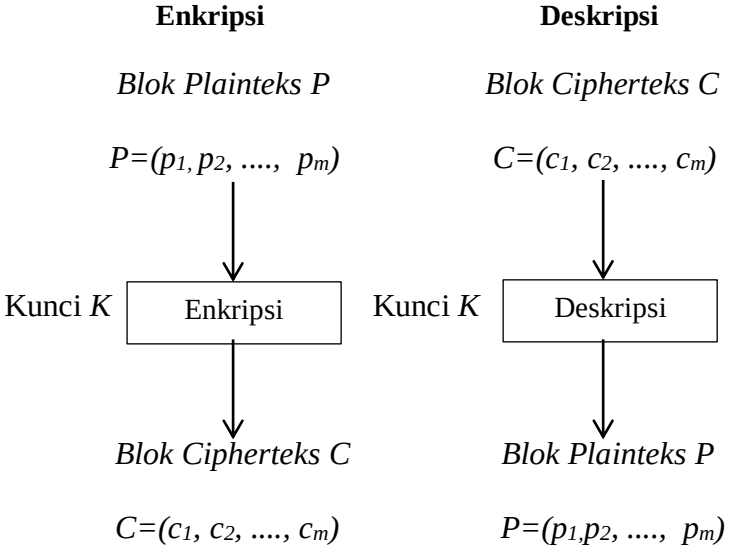
Sedangkan persamaan yang menyatakan proses *dekripsi* dengan kunci K adalah

$$DK(C) = P \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Fungsi E yang digunakan dalam proses enkripsi harus merupakan fungsi yang berkoresponden satu-ke-satu, sehingga

$$E^{-1} = D \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

Skema enkripsi dan dekripsi cipher blok dapat dilihat pada Gambar.



Gambar 2.3 Ilustrasi Enkripsi dan Dekripsi
Sumber : Chumaidi Rahman (2009)

Pada cipher blok *plainteks* dibagi menjadi beberapa blok dengan panjang tetap yaitu sepanjang kunci yang dimasukkan. Karena itulah terdapat kemungkinan panjang *plainteks* tidak habis dibagi dengan panjang ukuran blok yang ditetapkan atau panjang kunci. Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada ukuran blok-blok lainnya. Solusi permasalahan ini adalah dengan padding, yaitu dengan menambahkan blok terakhir dengan pola

bit yang teratur hingga panjang blok sama dengan ukuran blok yang ditetapkan. Pola bit teratur ini misalnya ditambahkan bit 0 semua, atau bit 1 semua, atau bit 0 dan 1 secara bergantian. Setelah proses dekripsi hapus kembali padding, agar hasil dekripsi kembali menjadi plainteks. Pada algoritma kriptografi yang beroperasi pada mode blok ini dikenal beberapa mode operasi. Empat mode operasi yang lazim digunakan pada cipher blok adalah:

1. *Electronic Code Book* (ECB)
2. *Cipher Block Chaining* (CBC)
3. *Cipher Feedback* (CFB)
4. *Output Feedback* (OFB)

Penggunaan mode-mode operasi tersebut tidak merubah fungsi enkripsi dan dekripsi yang telah didefinisikan.

Electronic Code Book (ECB) adalah salah satu teknik yang digunakan pada Cipher Blok. Pada mode ini, setiap blok *palainteks* (Pi) dienkripsi secara individual dan independen menjadi blok *cipherteks* ($C_{i)}$. Mode ini tidak akan mempengaruhi blok-blok lainnya. Istilah *"code book"* di dalam ECB muncul dari fakta bahwa karena blok *plainteks* yang sama selalu dienkripsi menjadi blok *cipherteks* yang sama, maka secara teoritis dimunkinkan membuat buku kode *plainteks* dan *cipherteks* yang berkoresponden. Namun, semakin besar ukuran blok, semakin besar ukuran buku kodenya (Ervyn Yoga Indra Kurniawan, 2011).

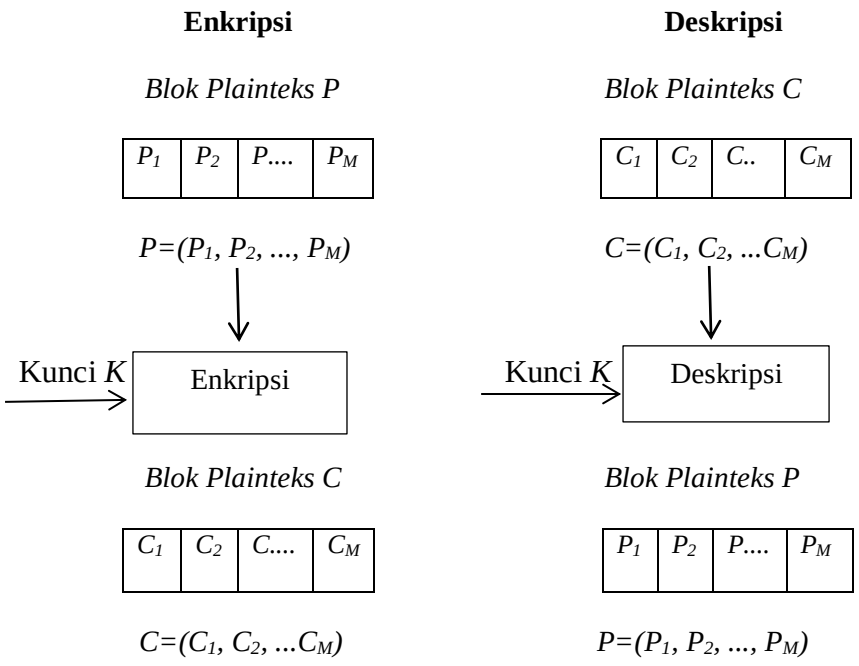
Pada mode ECB, setiap mode *plainteks* ($P_{i)}$ dienkripsi secara individual dan independen menjadi blok *cipherteks* ($C_{i)}$. Secara matematis proses enkripsi dengan mode ECB dapat dinyatakan sebagai berikut:

$$Ci = EK(Pi) \quad . . . . . (6)$$

dan proses dekripsi sebagai berikut:

$$Pi= DK(Ci) \quad . . . . . (7)$$

Dalam hal ini, Pi dan Ci merupakan blok *plainteks* dan *cipherteks* ke-i. Skema enkripsi dan dekripsi dengan mode ECB dapat dilihat pada gambar.



Gambar 2.4 Ilustrasi Enkripsi dan Dekripsi ECB
Sumber : Chumaidi Rahman (2016)

Mode ini merupakan mode termudah dari keempat mode yang telah disebutkan di atas. Setiap blok *plainteks* dienkripsi secara independen, sehingga

proses enkripsi tidak harus berlangsung secara linear atau proses enkripsi dapat dilakukan pada blok-blok secara tidak berurutan. Keuntungan mode ECB lainnya adalah kesalahan satu bit pada satu blok hanya akan mempengaruhi blok *cipherteks* yang berkoresponden pada proses dekripsi.

2.4 Algoritma Elgamal

2.4.1 Pengertian Algoritma Elgamal

Perkembangan teknologi yang semangkin maju membawa sebuah dampak pada keseluruh aspek kehidupan yang dijalani manusia, seperti dalam mengamankan sebuah keamana data yang dirahasiakan atau dilindungi. Berbagai hal telah dilakukan untuk mendapatkan jaminan keamanan data tersebut, Salah satu cara yang digunakan adalah dengan membangun sebuah keamanan data. Algoritma Elgamal merupakan salah satu dari algoritma kunci. Algoritma ini dikembangkan pertama kali oleh *Taher Elgamal* pada tahun 1985. Sampai saat ini, algoritma Elgamal masih dipercaya sebagai metode penyandian, seperti aplikasi PGP dan GnuPG yang dapat digunakan untuk pengamanan e-mail dan tanda tangan digital. Pada tahun 1994 pemerintah Amerika Serikat mengadopsi Digital *Signature Standard*, sebuah mekanisme penyandian yang berdasar pada algoritma Elgamal.

Algoritma Elgamal merupakan algoritma dalam kriptografi yang termasuk dalam kategori algoritma asimetris. Keamanan algoritma Elgamal terletak pada kesulitan penghitungan logaritma diskret pada bilangan modulo prima yang besar sehingga upaya untuk menyelesaikan masalah logaritma ini menjadi sangat sukar.

Algoritma Elgamal mempunyai kunci publik berupa tiga pasang bilangan dan kunci rahasia berupa satu bilangan. Algoritma ini mempunyai kerugian pada *cipherteks* yang mempunyai panjang dua kali lipat dari *plainteks*. Akan tetapi, algoritma ini mempunyai kelebihan pada enkripsi. Untuk *plainteks* yang sama, algoritma ini memberikan *cipherteks* yang berbeda (dengan kepastian yang dekat) setiap kali *plainteks* di enkripsi (Mukhammad Ifanto, 2009)

2.5 Exlusive-OR (XOR)

2.5.1 Pengertian Exlusive-OR (XOR)

Dalam upaya untuk meningkatkan keamanan SMS untuk merahasiakan segala data atau informasi yang dianggap penting, kriptografi mentransformasikan data awal *(plainteks)* ke dalam bentuk data sandi *(cipherteks)* yang tidak dapat dikenali. *Cipherteks* inilah yang kemudian dikirimkan oleh pengirim *(sender)* kepada penerima *(receiver)*. Setelah sampai di penerima, *cipherteks* tersebut ditranformasikan kembali ke dalam bentuk *plainteks* agar dapat dikenali. Dengan menggunakan kriptografi ini diharapkan hanya orang yang berkepentingan dan punya kunci yang bisa mengakses data yang dikirimkan. Kekuatan suatu teknik kriptografi adalah pada kerumitan kunci dan algoritmanya, sehingga pesan yang dikirimkan tidak mudah ditebak oleh orang lain sehingga data yang dikirimkan terjaga kerahasiaannya. Ada berbagai macam jenis algoritma kriptografi yang sekarang ini telah ada dan sedang dikembangkan, namun yang akan dibahas pada penelitian ini adalah algoritma Exlusive-OR (XOR).

Algoritma XOR adalah salah satu algoritma kriptografi modern dengan meng-XOR kan *plainteks* (P) dengan kunci (K) menghasilkan *cipherteks*. Pada penelitian ini akan diuraikan bagaimana mengimplementasikan algoritma Excusive-OR menggunakan pesan teks atau data sederhana ke dalam pemrograman dan menjelaskan bagaimana langkah–langkah proses enkripsi dan dekripsi data sederhana menggunakan algoritma Exclusive-OR (Suhardi, 2016)

2.5.2 Operasi Logika Exclusive OR (XOR)

Operator biner yang sering digunakan dalam cipher yang yang beroperasi dalam mode bit adalah XO. Notasi matematis untuk operator XOR adalah " ⊕ " . Operator XOR diperasikan pada dua bit dengan aturan sebagai berikut:

Tabel 2.1 Aturan Operasi XOR

A	**B**	**A ⊕ B**
0	0	0
0	1	1
1	0	1
1	1	0

Sumber : Suhardi (2016)

Contoh: jika 10011 ⊕ 11001 = 01010 hasilnya yang diperoleh adalah sebagai berikut:

$$\begin{array}{rrrrrrrrrrc} & 1 & & 0 & & 0 & & 1 & & 1 & \\ & 1 & & 1 & & 0 & & 0 & & 1 & \oplus \\ \hline 1 \oplus 1 & & 0 \oplus 1 & & 0 \oplus 0 & & 1 \oplus 0 & & 1 \oplus 1 & & \\ = 0 & & 1 & & 0 & & 1 & & 0 & & \end{array}$$

2.5.3 Analisis Proses Enkripsi dan Deskripsi metode *XOR*

Algoritma enkripsi menggunakan XOR adalah dengan meng-*XOR*-kan *plainteks* (P) dengan kunci (K) menghasilkan *cipherteks* (C):

$$C = P \oplus K > \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (8)$$

Algoritma dekripsi menggunakan *XOR* adalah dengan meng-*XOR*-kan *ciphertext* (C) dengan kunci (K) menghasilkan *plainteks* (P):

$$P = C \oplus K \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

Misalkan kita ingin mengirim pesan *(plainteks)* "UNPAB" dengan kunci "8", tahapan penghitungan manualnya adalah sebagai berikut :

1. Buka tabel ASCII, kemudian cari nilai "UNPAB" dan "8", sehingga kita temukan nilai U = 85, N=78, P=80, A=65, B=66 dan 8=56
2. Ubah angka 85, 78, 80, 65, 66 dan 56 kedalam bilangan biner, sehingga kita dapatkan nilai 85= 01010101, 78=01001110, 80=01010000, 65=01000001, 66=01000010 dan key 8=00111000.
3. Lakukan proses enkripsi dengan metode XOR (C = P ⊕ K) seperti berikut :

   a. Plainteks 01010101 (karakter 'U')
      Kunci 00111000 ⊕ (karakter '8')
      Cipherteks 01101101 (karakter 'm')

   b. Plainteks 01001110 (karakter 'N')
      Kunci 00111000 ⊕ (karakter '8')
      Cipherteks 01110110 (karakter 'v')

c. Plainteks 01010000 (karakter ‘P’)
Kunci 00111000 ⊕ (karakter ‘8’)
Cipherteks 01101000 (karakter ‘h’)

d. Plainteks 01000001 (karakter ‘A’)
Kunci 00111000 ⊕ (karakter ‘8’)
Cipherteks 01111001 (karakter ‘y’)

e. Plainteks 01000010 (karakter ‘B’)
Kunci 00111000 ⊕ (karakter ‘8’)
Cipherteks 01111010 (karakter ‘z’)

f. Cipherteks dari pesan “UNPAB” dengan kunci “8” adalah “mvhyz”. Karakter “mvhyz” didapatkan dari nilai :

01101101 *(biner)* =109 *(decimal)* =karakter ‘m’ (Tabel ASCII)

01110110 *(biner)* =118 *(decimal)* =karakter ‘v’ (Tabel ASCII)

01101000 *(biner)* =104 *(decimal)* =karakter ‘h’ (Tabel ASCII)

01111001 *(biner)* =121 *(decimal)* =karakter ‘y’ (Tabel ASCII)

01111010 *(biner)* =122 *(decimal)* =karakter ‘z’ (Tabel ASCII)

4. Untuk melakukan proses dekripsi dengan metode XOR

(P = C ⊕ K) seperti berikut :

a. *Cipherteks* 01101101 (karakter ‘m’)
Kunci 00111000 ⊕ (karakter ‘8’)
Plainteks 01010101 (karakter ‘U’)

b. *Cipherteks* 01110110 (karakter ‘v’)
Kunci 00111000 ⊕ (karakter ‘8’)
Plainteks 01001110 (karakter ‘N’)

c. *Cipherteks* 01101000 (karakter ‘h’)
Kunci 00111000 ⊕ (karakter ‘8’)
Plainteks 01010000 (karakter ‘P’)

d. Cipherteks 01111001 (karakter ‘y’)
Kunci 00111000 ⊕ (karakter ‘8’)
Plainteks 01000001 (karakter ‘A’)

e. Cipherteks 01111010 (karakter ‘z’)
Kunci 00111000 ⊕ (karakter ‘8’)
Plainteks 01000010 (karakter ‘B’)

f. Plainteks dari pesan “mvhyz” dengan kunci “8” adalah “UNPAB”. Karakter “mvhyz” didapatkan dari nilai seperti yang telah dijelaskan diatas.

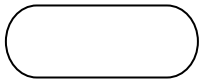
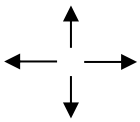
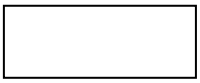
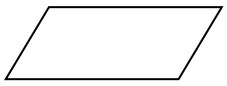
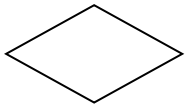
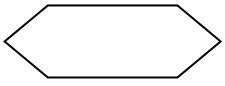
2.6 *Flowchart*

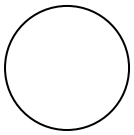
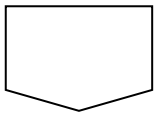
Flowchart yang juga disebut dengan diagram alir merupakan kumpulan simbol-simbol atau skema yang menunjukkan atau menggambarkan rangkaian kegiatan-kegiatan program dari awal hingga akhir. *Flowchart* ini merupakan penggambaran dari urutan langkah-langkah pekerjaan dari suatu algoritma.

Flowchart merupakan gambaran berbentuk suatu grafik yang disertai langkahlangkah dan urutan suatu prosedur dari suatu program. Flowchart dapat membantu proses analisis, perancangan dan pengkodean untuk memecahkan masalah kedalam bagian-bagian yang lebih kecil untuk pengoperasiannya. *Flowchart* biasanya mempermudah penyelesaian suatu masalah pada evaluasi lebih lanjut (Malabay, 2016).

Adapun simbol-simbol pada *flowchart akan* lihat pada tabel sebagai berikut:

Tabel 2.2 Simbol-Simbol Flowchart

No	Simbol	Nama Simbol	Fungsi
1.		Terminal	Permulaan atau akhir dari program
2.		Garis Alir (Flow Line)	Arah aliran Program
3		*Preparation*	Proses inisialisasi atau pemberian harga awal
4		Proses	Suatu proses perhitungan atau suatu proses pengolahan data
5		*Input-Output*	Suatu proses input/output data, parameter, informasi
6.		*Predefined Process* (Sub program)	Permulaan Sub program atau proses menjalankan sub program
7		*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya
8.		*On Page Preparation*	suatu simbol yang menyediakan tempat pengolahan

No	Simbol	Nama Simbol	Fungsi
9.		*Connector*	Penghubung bagian-bagian flowchart yang berada pada satu halaman
10.		*Off-Page Connector*	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

Sumber : Opik Taupik K (2013)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pesatnya perkembangan teknologi informasi dan komunikasi saat ini membuat kemudahan bagi setiap elemen masyarakat untuk berkomunikasi dan mendapatkan suatu informasi dengan mudah dan cepat, dengan berkembangnya suatu teknologi komunikasi setiap orang dapat menerima segala bentuk pesan komunikasi dimana saja dan kapan saja tanpa mengenal batas tempat, ruang dan waktu. Dan fasilitas itu bisa didapatkan melalui telepon seluler yang banyak dipakai saat ini yaitu SMS *(Short Message Service)* yang dapat mempermudahkan setiap elemen masyarakat saat ini, Namun dalam pengiriman suatu pesan komunikasi memiliki banyak hambatan yang membuat suatu kendala dalam pengiriman suatu pesan, kendala itu dapat berupa lemahnya suatu keamanan data yang membuat pesan yang dirahasiakan dapat diketahui oleh orang lain dengan cara menyadap atau memanipulasi pesan rahasia yang dikirim. Maka dari itu diperlukan suatu rumusan yang dapat meningkatkan keamanan suatu pesan komunikasi pada SMS *(Short Message Service)*.

Namun, setiap harinya perkembangan teknologi semangkin canggih dan berkembang begitu pesat dengan seiring waktu dan berkembangnya zaman, maka muncullah beberapa kekurangan yang ada pada SMS *(Short Message Service)*. Salah satunya yang paling disorot adalah tingkat keamanan yang terdapat pada SMS *(Short Message Service)* tersebut. Karena tidak dapat dipungkiri bahwa pada zaman sekarang SMS *(Short Message Service)* digunakan untuk

berkomunikasi sebagai pertukar pesan informasi yang bersifat rahasia yang digunakan para Pengusaha, Pejabat, Petinggi Dan Pemerintahan. Sehingga ada banyak terjadi suatu pencurian data SMS *(Short Message Service)* atau yang sering disebut suatu penyadapan yang dapat membuat tingkat keamanan SMS *(Short Message Service)* perlu lebih ditingkatkan lagi.

Dalam mengatasi suatu masalah pada sistem tingkat keamanan pada SMS *(Short Message Service)*, maka akan merancang dan membuatkan sebuah aplikasi enkripsi dan dekripsi yang akan diimplementasikan untuk aplikasi SMS pada dua buah komputer atau laptop. Metode yang digunakan untuk mengamankan pesan SMS *(Short Message Service)* yaitu dengan Metode Algoritma Elgamal dan Mode Operasi Blok Cipher Electronik Code Book (ECB). Cipher Elektronik Book digunakan karena *Enkripsi* dan *dekripsi* yang sifatnya acak dan cocok untuk mengenkripsi file yang diakses secara acak karena tiap blok *plaintext* dienkripsi dengan secara independen (Arif Kurnia Rachman, 2010). Maka dibuatlah sebuah laporan Skripsi yang berjudul "PENERAPAN KEAMANAN SMS (*SHORT MASSAGE SERVICE*) DENGAN ALGORITMA ELGAMAL MENGGUNAKAN METODE *ELECTRONIK CODE BOOK* (ECB)" yang mana dapat membantu untuk meningkatkan pengamanan suatu pesan SMS (*Short Massage Service*) yang dianggap penting atau pesan yang dirahasiakan agar pesan tersebut tetap terlindungi kerahasiaannya.

1.2 Rumusan Masalah

Adapun Rumusan Masalah yang terdapat pada penulisan skripsi ini meliputi :

1. Bagaimana meningkatkan penerapan keamanan pesan SMS *(Short Message Service)* Menggunakan Algoritma Elgamal dengan Moetode ECB (Electronik Code Book) ?
2. Bagaimana Mengimplementasikan Keamanan Pesan SMS *(Short Message Service)* dengan Algoritma Elgamal dan ECB (Elektronik Cipher Book) dalam bentuk aplikasi ?

1.3 Batasan Masalah

Adapun Batasan Masalah yang terdapat pada penulisan skripsi ini meliputi:

1. Algoritma kunci publik yang digunakan adalah Algoritma Elgamal dan Mode Operasi Algoritma Simetris yang digunakan adalah Metode Operasi Electronik Code Book (ECB) dengan panjang 128 bits dan menggunakan dua buah laptop
2. Teknik *enkripsi* yang digunakan adalah teknik XOR dan bahasa Pemograman yang digunakan untuk membangun aplikasi adalah bahasa pemrograman PHP

1.4 Tujuan Penulisan

Adapun tujuan dari penulisan yang terdapat pada laporan skripsi ini meliputi:

1. Dapat Meningkatkan Suatu Keamana Pesan SMS *(Short Message Service)* dengan Menggunakan Algoritma Elgamal dan Metode Operasi Blok Cipher ECB (Electronik Code Book)

2. Membuat suatu aplikasi untuk mempermudah pengimplementasian Mengamankan Pesan SMS *(Short Message Service)*

1.5 Manfaat Penulisan

Adapun manfaat dari penulisan ini yaitu agar dapat meningkatkan keamanan pesan SMS *(Short Message Service)* yang dirahasiakan atau yang dianggap penting dari orang-orang yang ingin melakukan suatu kejahatan seperti menyadap pesan SMS *(Short Message Service)* yang dapat merugikan pengirim dan penerima pesan.

DAFTAR PUSTAKA

Andrian, Yudhi, and Purwa Hasan Putra. "Analisis Penambahan Momentum Pada Proses Prediksi Curah Hujan Kota Medan Menggunakan Metode Backpropagation Neural Network." *Seminar Nasional Informatika (SNIf)*. Vol. 1. No. 1. 2017.

Ardiansyah., Beni Irawan., Tedy Rismawan. (2015).Rancang Bangun Sistem Keamanan Kendaraan Bermotor Dengan Sms Gateway Berbasis Mikrokontroler Dan Android. *Jurnal Coding, Sistem Komputer UntanVolume 03, No. 1 (2015),* hal 11-19ISSN : 2338-493x,. Diakses dari file:///C:/Users/Acer/Downloads/9672-31025-2-PB%20(1).pdf

Aryza, S., Irwanto, M., Lubis, Z., Siahaan, A. P. U., Rahim, R., & Furqan, M. (2018). A Novelty Design Of Minimization Of Electrical Losses In A Vector Controlled Induction Machine Drive. In IOP Conference Series: Materials Science and Engineering (Vol. 300, No. 1, p. 012067). IOP Publishing.

Arif Kurnia Rachman. (2010). Perbandingan Mode Chiper Electronic Code Book Dan Chiper Block Chaining Dalam Pengamanan Data. diakses dari http://jurtek.akprind.ac.id/sites/default/files/84-89_rahman.pdf

Batubara, Supina. "Analisis perbandingan metode fuzzy mamdani dan fuzzy sugeno untuk penentuan kualitas cor beton instan." IT Journal Research and Development 2.1 (2017): 1-11.

Batubara, Supina, Sri Wahyuni, and Eko Hariyanto. "Penerapan Metode Certainty Factor Pada Sistem Pakar Diagnosa Penyakit Dalam." Seminar Nasional Royal (SENAR). Vol. 1. No. 1. 2018.

Chandra. (2016). Keamanan Data Dengan Metode Kriptografi Kunci Publik.http://ejournal.stmiktime.ac.id/index.php/jurnalTIMES/article/view/548

Chumaidi Rahman., Isbat Uzzin Nadhori., Kholid Fathoni. (2009). Implementation And Study Blowfish Algotythm For Email Encryption. http://docplayer.info/186512-Makalah-tugas-akhir-studi-dan-implementasi-algoritma-blowfish-untuk-enkripsi-email.html

Efrandi., Asnawati., Yupiyanti. (2014). Aplikasi Kriptografi Pesan Menggunakan Algoritma Vigenere Cipher. *Jurnal Media Infotama Vol. 10 No. 2, September 2014,. ISSN 1858 – 2680* Diakses dari https://jurnal.unived.ac.id/index.php/jmi/article/view/242/220

Ervyn Yoga Indra Kurniawan. (2011). Penerapan Teori Chaos Pada Kriptografi Menggunakan Algoritma Stream Cipher Dan Electronic Code Book (Ecb) Untuk Keamanan Pesan Teks. http://mahasiswa.dinus.ac.id/docs/skripsi/jurnal/13608.pdf

Fachri, barany. Perancangan sistem informasi iklan produk halal mui berbasis mobile web menggunakan multimedia interaktif. Jurasik (jurnal riset sistem informasi dan teknik informatika), 2018, 3: 98-102.

Fachri, barany. "aplikasi perbaikan citra efek noise salt & papper menggunakan metode contraharmonic mean filter." seminar nasional royal (senar). Vol. 1. No. 1. 2018.

Fresly Nandar Pabokory., Indah Fitri Astuti., Awang Harsa Kridalaksana. (2015). Implementasi Kriptografi Pengamanan Data Pada Pesan Teks, Isi File Dokumen, Dan File Dokumen Menggunakan *Algoritma Advanced Encryption Standard.* Di akses dari http://e-journals.unmul.ac.id/index.php/JIM/article/view/23

Ginting, G., Fadlina, M., Siahaan, A. P. U., & Rahim, R. (2017). Technical approach ofTOPSIS in decision making. Int. J. Recent Trends Eng. Res, 3(8),58-64

Irham Mu'alimin Arrijal., Rusdi Efendi., Boko Susilo. (2016). Penerapan Algoritma Kriptografi Kunci Simetris Dengan Modifikasi Vigenere Cipher Dalam Aplikasi Kriptografi Teks. Jurnal Pseudocode, *Volume III Nomor 1, Februari 2016, ISSN 2355 – 5920* https://ejournal.unib.ac.id/index.php/pseudocode/article/view/835

Malabay. (2016). Pemanfaatan Flowchart Untuk Kebutuhan Deskripsi Proses Bisnis

Mayasari, Nova. "Comparison of Support Vector Machine and Decision Tree in Predicting On-Time Graduation (Case Study: Universitas Pembangunan Panca Budi)." *Int. J. Recent Trends Eng. Res* 2.12 (2016): 140-151.\

Mukhammad Ifanto. (2009). Metode Enkripsi Dan Dekripsi Dengan Menggunakan\ Algoritma Elgamal. http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/20092010/Makalah0910/MakalahStrukdis0910-096.pdf

Putera, A., Siahaan, U., & Rahim, R. (2016). Dynamic key matrix of hill cipher using genetic algorithm. Int. J. Secur. Its Appl, 10(8), 173-180

Puspita, Khairani, and Purwa Hasan Putra. "Penerapan Metode Simple Additive Weighting (SAW) Dalam Menentukan Pendirian Lokasi Gramedia Di Sumatera Utara." Seminar Nasional Teknologi Informasi Dan Multimedia, ISSN. 2015.

Suhardi. (2016). Aplikasi Kriptografi Data Sederhana Dengan Metode Exlusive (Xor). https://ejurnal.plm.ac.id/index.php/Teknovasi/article/view/67

Sunardi, Hari aMurti, Hersatoto Listiyono. (2009). Aplikasi SMS Gatewa. *Jurnal Teknovasi Volume 03, Nomor 2, 2016, 23 – 31ISSN : 2355-701X.,* https://www.unisbank.ac.id/ojs/index.php/fti1/article/view/88

Suhardi. (2016). Aplikasi Kriptografi Data Sederhana Dengan Metode Exlusive(Xor).https://ejurnal.plm.ac.id/index.php/Teknovasi/article/view/67