PENERAPAN SISTEM ENKRIPSI AES-128 PADA APLIKASI SMS BERBASIS ANDROID

SKRIPSI

Oleh

Fachrur Rijal

NIM 122410101037

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2019

PENERAPAN SISTEM ENKRIPSI AES-128 PADA APLIKASI SMS BERBASIS ANDROID

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan pendidikan di Program Studi Sistem Informasi Universitas Jember dan mendapat gelar Sarjana Sistem Informasi

Oleh

Fachrur Rijal

NIM 122410101037

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS JEMBER

2019

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Allah SWT yang senantiasa memberikan kenikmatan, rahmad serta hidayah-Nya
2. Kedua orang tua: Bapak Abdul Hafid dan Ibu Mutmainnah yang senantiasa mendidik dan mendoakan saya
3. Ketiga saudara kandung saya: Milla Rahma F. Ulfa Nurul B. dan Rifqy Zaidan Z. yang senantiasa memberi dukungan moral
4. Kedua dosen pembimbing, Bapak Drs. Antonius Cahya P, M.App.Sc dan Bapak Yanuar Nurdiansyah, ST, M.Cs yang senantiasa memberikan bimbingan, dukungan dan semangat dalam pengerjaan tugas akhir ini
5. Seluruh dosen Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Jember
6. Keluarga Next Planz : Yudi, Brelly, Affan, Vadil, Candra dan Fariz
7. Keluarga kontrakan Sumatra songo: Bagus Cinta, Riska Icha, Alfat Kopet, Coach RZK, Hilmi Koblo, Arip Panjang
8. Teman-teman dari MIXMEDIA dan Museum Huruf
9. Hofi yang membantu pengerjaan program
10. Aji Mukti yang menyediakan tempat untuk kawan-kawan agar bias mengerjakan tugas akhirnya di akhir-akhir waktu perkuliahan ini
11. Rinaldi sayoga dan juga teman-teman yang lain yang rela meminjami saya laptop agar bias merampungkan tugas akhir saya
12. Seluruh anggota grup H-sidang yang selalu mengingatkan bahwa kita takkan muda lagi
13. Dan pada seluruh orang-orang yang saya temui dan memberi banyak pelajaran dan manfaat pada kehidupan saya

MOTTO

"Tak mungkin menang banyak, jika tak bertaruh besar"

(Seringai)

"What uneasiness lies in being loved"

(Osamu Dazai)

PERNYATAAN

Saya yang bertanda tangan dibawah ini:

Nama : Fachrur Rijal

NIM : 122410101037

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penerapan Sistem Enkripsi AES-128 Pada Aplikasi SMS Berbasis Android", adalah benar-benar hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa adanya tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember, 11 Juli 2019

Yang menyatakan,

Fachrur Rijal

NIM 122410101037

SKRIPSI

PENERAPAN SISTEM ENKRIPSI AES-128 PADA APLIKASI SMS BERBASIS ANDROID

Oleh
Fachrur Rijal
NIM 122410101037

PEMBIMBING

Dosen Pembimbing Utama	: Drs. Antonius Cahya P, M.App.Sc., Ph.D.
Dosen pembimbing Pendamping	: Yanuar Nurdiansyah, ST, M.Cs

PENGESAHAN PEMBIMBING

Skripsi berjudul ” Penerapan Sistem Enkripsi AES-128 Pada Aplikasi SMS Berbasis Android”, telah diuji dan disahkan pada,

hari, tanggal : Kamis, 11 Juli 2019

tempat : Fakultas Ilmu Komputer Informasi Universitas Jember

Disetujui oleh:

Pembimbing I,	Pembimbing II,
Drs. Antonius Cahya P, M.App.Sc., Ph.D.	Yanuar Nurdiansyah, ST, M.Cs
NIP 196909281993021001	NIP 198201012010121004

PENGESAHAN PENGUJI

Skripsi berjudul ” Penerapan Sistem Enkripsi AES-128 Pada Aplikasi SMS Berbasis Android”, telah diuji dan disahkan pada,

hari, tanggal : Kamis, 11 Juli 2019

tempat : Fakultas Ilmu Komputer Informasi Universitas Jember

Disetujui oleh:

Penguji I,	Penguji II,
Prof. Drs. Slamin, M.Comp.Sc., Ph.D.	Tio Dharmawan S.Kom, M.Kom
NIP 196704201992011001	NIP 760016851

Mengesahkan

a.n Dekan,

Wakil Dekan I Fakultas Ilmu Komputer

Drs. Antonius Cahya P, M.App.Sc., Ph.D.

NIP 196909281993021001

RINGKASAN

Penerapan Sistem Enkripsi AES-128 Pada Aplikasi SMS Berbasis Android; Fachrur Rijal, 122410101037; 2019; 85 halaman; Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Jember.

Keamanan dan kerahasiaan suatu informasi yang terdapat pada layanan pesan singkat di telepon genggam kurang begitu diperhatikan oleh para pengguna telepon genggam itu sendiri. Guna mengatasi permasalahan tersebut digunakanlah metode kriptografi untuk mengamankan informasi yang tersebut. Terdapat banyak algoritma kriptografi yang bisa digunakan salah satunya adalah algoritma *Advances Encryption Standard* (AES). Pada proyek ini menggunakan algoritma AES-128 yang diimplementasikan pada sebuah aplikasi layanan pesan singkat berbasis *Android.* Algoritma AES-128 ini diimplementasikan di dalam isi pesan yang dikirimkan oleh pengirim pesan singkat tersebut. Dalam penggunaanya cipher key hanya diketahui oleh pengirim dan penerima. Algoritma AES-128 merubah *plain text* menjadi kode-kode dalam bentuk *hexadecimal* melalui operasi perhitungan XOR dengan menggunakan konversi *plain text* dan *cipher key* melalui sepuluh perputaran. Sehingga membuat isi pesan menjadi lebih aman dari tindakan pembajakan.

PRAKATA

Alhamdulillaah, segala puji bagi Allah SWT yang telah melimpahkan segala karunia, nikmat serta hidayah-Nya kepada saya untuk bias menyelesaikan tugas akhir atau skripsi yang berjudul "Penerapan Sistem Enkripsi AES-128 Pada Aplikasi SMS Berbasis Android". Skripsi ini disusun guna memenuhi syarat untuk menyelesaikan studi saya di Program Studi Sistem Informasi Fakultas Ilmu Komputer Universitas Jember.

Tak lupa juga saya mengucapkan banyak terimakasih kepada pihak-pihak yang telah membantu dalam menyelesaikan tugas akhir ini. Terimakasih secara khusu saya ucapkan kepada:

1. Bapak Drs. Antonius Cahya P, M.App.Sc selaku Dosen Pembimbing Utama dan Bapak Yanuar Nurdiansyah, ST, M.Cs selaku Dosen Pembimbing Anggota yang telah memberikan bimbingan, bantuan serta dukungan dalam proses pembuatan skripsi ini.
2. Bapak dan Ibu dosen serta karyawan Fakulats Ilmu Komputer Universitas Jember.
3. Keluarga tercinta, Ayah saya Abdul Hafid, Ibu saya Mutmainah, kakak Milla Rahma F., adik Ulfa Nurul B., dan adik Rifqy Zaidan Z. yang telah memberikan dukungan moril serta materil kepada saya.
4. Keluarga Next Planz : Yudi, Brelly, Affan, Vadil, Candra dan Fariz.
5. Keluarga kontrakan Sumatra songo: Bagus Cinta, Riska Icha, Alfat Kopet, Coach RZK, Hilmi Koblo, Arip Panjang.
6. Teman-teman dari MIXMEDIA dan Museum Huruf.
7. Hofi Atmajaya yang membantu proses pembuatan program.
8. Aji Mukti yang menyediakan tempat untuk kawan-kawan agar bias mengerjakan tugas akhirnya di akhir-akhir waktu perkuliahan ini.

9. Rinaldi Sayoga dan teman-teman yang lain yang rela meminjami saya laptop agar bias merampungkan tugas akhir saya.
10. Seluruh anggota grup H-sidang yang selalu mengingatkan bahwa kita takkan muda lagi.
11. Dan pada seluruh orang-orang yang yang saya temui dan memberi banyak pelajaran dan manfaat pada kehidupan saya.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1. PENDAHULUAN

Bab ini merupakan bagian awal dari penulisan tugas akhir ini. Bab ini berisi latar belakang, perumusan masalah, tujuan dan manfaat, batasan masalah, metodologi penelitian dan sistematika penulisan.

1.1 Latar Belakang

Smartphone merupakan salah satu hasil perkembangan teknologi dalam hal komunikasi saat ini. *Smartphone* juga dijadikan sebuah kebutuhan bagi sebagian besar manusia yang ada di dunia. Hampir seluruh golongan masyarakat di dunia menggunakan *smartphone* sebagai alat komunikasi yang mereka gunakan. Hal yang mempengaruhi masyarakat untuk menggunakan *smartphone* yakni kemudahan yang ditawarkan melalui fitur-fitur yang terbilang canggih seperti sebuah komputer genggam. Selain hal-hal yang telah disebutkan di atas faktor pendukung yang mempengaruhi masyarakat untuk beralih menggunakan *smartphone* yakni harga yang ditawarkan oleh produsen-produsen *smartphone* yang bisa dijangkau oleh semua golongan masyarakat. Salah satu jenis sistem operasi mobile yang berkembang pesat saat ini adalah *Android*. *Smartphone Android* merupakan *smartphone* yang saat ini paling banyak diminati oleh masyarakat dengan segala fasilitas dan harga yang ditawarkan oleh produsen-produsen *smartphone* yang mengembangkan *Android*. Sebagai salah satu alat komunikasi smartphone Android memiliki fasilitas wajib yang harus ada pada perangkat telepon seluler, salah satunya adalah SMS (*Short Message Service*). Meskipun memiliki banyak fitur canggih, layanan SMS masih digunakan oleh para pengguna *smartphone* karena mudah dan praktis untuk digunakan.

Dari banyaknya penggunaan telepon seluler yang menggunakan layanan SMS sebagai layanan komunikasi pesan singkat kurang diimbangi oleh faktor keamanan yang menjamin kerahasiaan bagi para penggunanya. Proses pengiriman dan penerimaan pesan sangat rentan terhadap upaya pencurian, penyadapan, pembajakan, pemerasan dan banyak hal lain terhadap suatu informasi (Andy,2012). Karena hal-hal tersebut perangkat seluler yang digunakan, khususnya Android

memerlukan sebuah fitur keamanan yang dapat melindungi informasi dalam SMS yang dikirim dan diterima dengan enkripsi pada pesan tersebut.

Enkripsi adalah proses mengamankan suatu informasi dengan membuat informasi tersebut tidak dapat dibaca tanpa bantuan pengetahuan dan atau alat khusus (Anwar, 2015). Enkripsi memungkinkan merubah kode-kode yang dapat dimengerti diubah kedalam kode-kode yang tidak dimengerti dengan menggunakan algoritma khusus. Untuk memecahkan kode yang telah dienkripsi dibutuhkan sebuah proses dekripsi.

Pada penelitian ini akan menggunakan metode enkripsi dengan menggunakan algoritma AES-128. Algoritma AES digunakan pada penelitian ini karena selain dinilai aman dalam proses enkripsi dan dekripsi algoritma AES dinilai cukup baik untuk diterapkan pada berbagai software maupun *hardware*. Penggunaan algoritma AES-128 akan diimplementasikan pada isi pesan SMS yang dikirim dan diterima. Diharapkan pengimplementasian algoritma AES-128 dapat menjaga dan melindungi kerahasiaan dari pengirim dan penerima SMS.

1.2 Rumusan Masalah

Berdasarkan uraian di atas terdapat beberapa permasalahan, maka dapat diambil rumusan masalah, antara lain:

1. Bagaimana implementasu kriptografi AES-128 terhadap isi informasi yang terdapat dalam SMS?
2. Bagaimana rancangan arsitektur dalam membangun aplikasi SMS security?

1.3 Tujuan dan Manfaat

Tujuan

Tujuan dari penelitian ini adalah:

1. Mengetahui proses implementasi kriptografi AES-128 dalam proses merahasiakan dan mengamankan isi informasi dalam SMS.
2. Merancang dan membangun aplikasi SMS security dengan keamanan data yang terlindungi

Manfaat

Manfaat dari penelitian ini adalah:

1. Manfaat akademis

   Hasil dari penelitian ini diharapkan dapat memberikan pengetahuan dan juga sebagai bahan bagi siapa saja yang membutuhkan informasi seputar penelitian ini sebagai bahan referensi.

2. Manfaat bagi peneliti

   Mengetahui cara kerja dan penerapan algoritma kriptografi AES-128 untuk merahasiakan dan mengamankan informasi dalam SMS.

3. Manfaat bagi masyarakat

   Memberikan pengetahuan tentang betapa pentingnya suatu sistem enkripsi untuk pesan SMS agar tak mudah disadap oleh pihak lain, danjuga memberikan pengetahuan tentang algoritma kriptografi AES-128.

1.4 Batasan Masalah

Terdapat beberapa batasan masalah dalam pengerjaan penelitian ini, antara lain:

1. Aplikasi yang dibangun berbasis Android.
2. Output yag dihasilkan berupa aplikasi Android yang mampu mengenkripsi dan juga mendekripsi isi SMS.
3. Teknik kriptorafi yang digunakan adalah algoritma AES-128 yang digunakan untuk merahasiakan dan mengamankan isi dari SMS yang dikirim dan diterima.

BAB 2. TINJAUAN PUSTAKA

Dalam penelitian tentang pengimplementasian algoritma AES-128 yang diterapkan pada aplikasi *SMS Security* dibutuhkan beberapa landasan teori yang berkaitan tentang penelitian ini, antara lain:

2.1 Penelitian Terdahulu

Pada penelitian terdahulu yang membahas tentang enkripsi pada SMS dengan judul "Implementasi Algoritma *Caesar, Chiper Disk* Dan *Scytale* pada Aplikasi Enkripsi Dan Dekripsi Pesan Singkat, LumaSMS" yang dilakukan oleh Yusuf Triyuswoyo, dkk dari Jurusan Manajemen Sistem Informasi Universitas Gunadarma pada tahun 2014. Pada penelitian tersebut dijelaskan penggunaan algoritma kriptografi klasik, yaitu *Caesar, Chiper Disk* Dan *Scytale* yang diimplementasikan dalam pembuatan program aplikasi LumaSMS. Penelitan tersebut bertujuan untuk melindungi isi dari pesan SMS yang dikirim maupun yang diterima dari risiko yang mengancam keamanan pesan SMS, seperti: SMS *spoofing*, SMS *snooping* dan SMS *interception*. Hasil yang telah didapat dari penelitian tersebut adalah aplikasi LumaSMS yang mampu melindungi kerahasiaan dari isi pesan SMS yang dikirimkan melalui jaringan operator seluler (Triyuswoyo,2014).

Penelitian sebelumnya yang pernah dilakukan dan mengimplementasikan algoritma AES-128, yakni dengan judul "Implementasi Algoritma AES-128 Pada *Mobile Learning* Universitas Jember" yang dilakukan oleh Ragilliyandi Erick Putra I, dkk dari Program Studi Sistem Informasi Universitas Jember pada tahun 2014. Pada penelitan tersebut dijelaskan penggunaan algoritma AES-128 yang diterapkan pada *mobile learning* Universitas Jember dan diimplementasikan pada data pengguna *mobile learning* Universitas Jember, antara lain nama mahasiswa, NIM, *password*, mata kuliah yang diikuti, daftar kegiatan yang diikuti dan lain sebagainya. Hasil yang telah didapatkan dari penelitian tersebut adalah proses komunikasi data untuk validasi data login yang telah diamankan dengan menggunakan algoritma AES-128 tidak dapat diterjemahkan secara langsung dan keamanan data dari pengguna aplikasi *mobile learning* Universitas Jember dapat terjaga kerahasiaanya (I, 2014).

Pada penelitian sebelumnya yang juga membahas tentang keamanan pada pesan pribadi atau SMS yang dilakukan oleh Hasan Jindan, dkk dari Prgram Studi Sistem Informsi Uniersitas Jember pada tahun 2015, dengan judul penelitiannya," Pesan Rahasia Dengan Metode Kriptografi Elgamal pada Perangkat *Android Mobile*". Pada penelitian tersebut penulis menjelaskan tentang penggunaan metode enkripsi dan dekripsi Elgamal yang diterapkan pada aplikasi pesan singkat atau SMS berbasis *Android.* Proses pembentukan kunci yang digunakan sebagai *public key* dan *private key* membutuhkan bilangan prima yang bernilai besar agar isi pesan semakin aman dari pembajakan yang dilakukan olh pihak lain yang tidak bertanggungjawab (Jindan, 2015).

2.2 Kriptografi

Kriptografi adalah ilmu mengenai teknik enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi (Kromodimoeldjo, 2010). Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci).

2.3 Algoritma AES-128

AES (*Advanced Encryption Standard*) merupakan standar enkripsi yang diperkenalkan oleh NIST (*National Institute of Standard and Technology*) pada tahun 2001 sebagai pengganti dari algoritma DES. Algoritma AES yang disebut juga dengan algoritma Rijndael menggunakan teknik enkripsi *block cipher* dengan menggunakan subtitusi terhadap tabel *S-Box* secara langsung terdhadap naskah (*plaintext*).

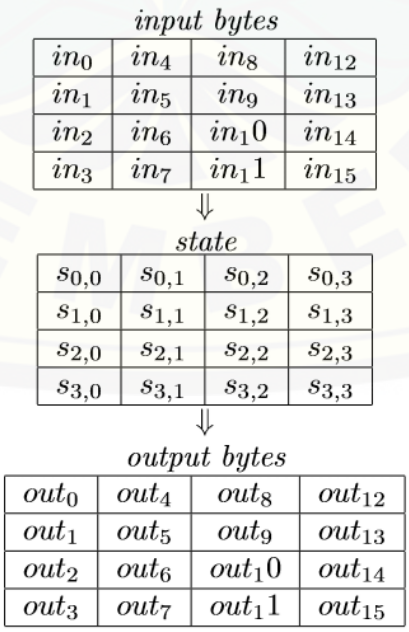
Input dan *output* dari algoritma AES terdiri dari urutan data sebesar 128 bit. Urutan data tersebut juga disebut sebagai blok atau *plaintext.* Sedangkan *cipher key* dari AES terdiri dari panjang kunci dengan nilai besaran dalam bit sebesat 128, 196

dan 256. Besaran bit yang dipakai pada kunci mempengaruhi pada jumlah putaran yang diterapkan pada algoritma AES. Proses perputaran (*Round*) enkripsi pada algoritma AES-128 dikerjakan sebanyak 10 kali (Nr=10), dengan besar kunci (Nk=4) dan besar blok (Nb=4) (Kromodimoeldjo, 2010). Jumlah putaran dalam algoritma AES digambarkan pada tabel 2.1.

Tabel 2. 1 Tabel jumlah perputaran AES (Kromodimoeljo, 2010)

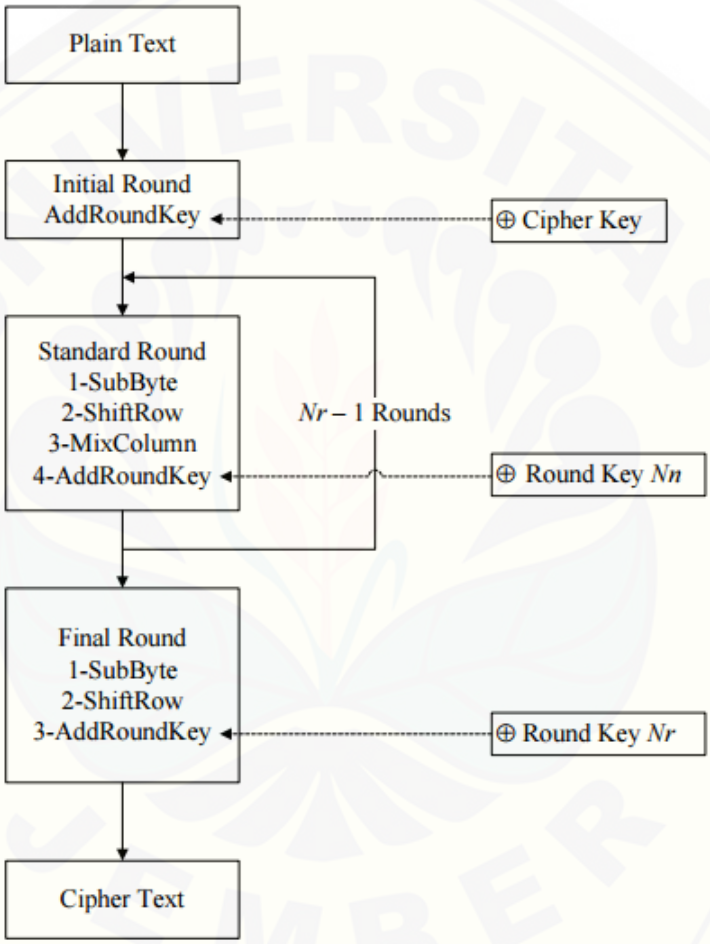
	Besar Kunci (Nk)	**Besar Blok (Nb)**	**Jumlah Putaran (Nr)**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Algoritma enkripsi AES beroperasi terhadap *state* dari teks yang dipandang sebagai matrik yang terdiri dari 16 byte (Kromodimoeldjo, 2010). Setiap kolom matrik mempresentasikan satu *word*, dengan $8_{0,i}$ sebagai *most significant byte* untuk kolom i. Gambar 2.1 menunjukkan bagaimana *state* didapat dari *input* dan bagaimana *state* dijadikan *output*. *Transformasi AddRoundKey, Subtitute Byte, ShiftRows* dan *MixColumns* semua dilakukan terhadap *state*.



Gambar 2. 1 *Input, State, Output* (Kromodimoeldjo, 2010)

Proses enkripsi pada algoritma AES terdiri dari 4 jenis transformasi *bytes*, yaitu *Subtitute Byte*, *ShiftRows, Mixcolumns*, dan *AddRoundKey* (Ilyas, 2014). Pada awal proses enkripsi, *input* (kunci) yang telah disalin ke dalam *state* akan mengalami transformasi *byte AddRoundKey*, dan hasil dari transformasi tersebut yang akan digunakan dalam operasi perhitungan XOR terhadap *plain text*. Ilustrasi proses enkripsi AES dapat digambarkan seperti pada gambar 2.2.



Gambar 2. 2 Diagram Alir Enkripsi AES-128 (Ilyas, 2014)

Dari gambar 2.2 dijelaskan bahwa putaran yang dilakukan dalam algoritma AES-128 dilakukan dalam Nr=10 putaran, yakni sebagai berikut:

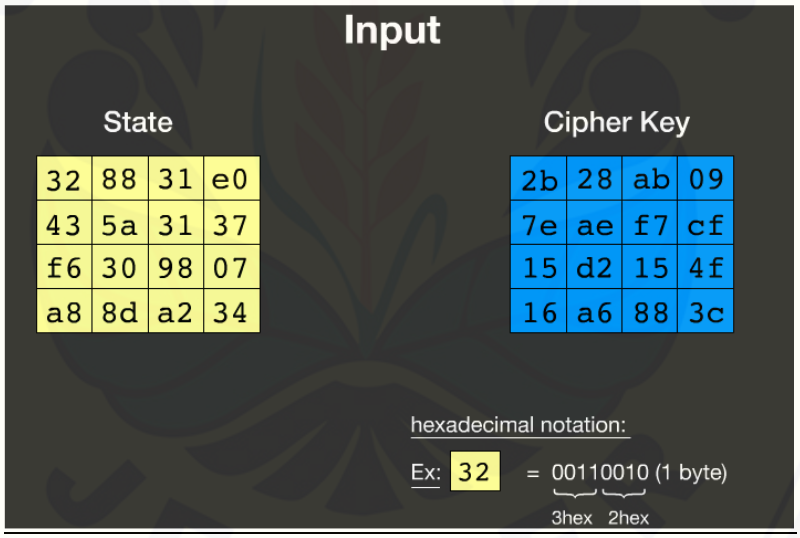
a. Transformasi *AddRoundkey* sebanyak 10 kali

Tahapan yang pertama ini juga disebut sebagai tahap *Key Schedule*, pada tahapan ini kunci yang telah ditentukan mengalami proses transformasi terlebih dahulu sebanyak 10 kali sebelum kunci dijumlahkan dengan *plaintext.*

b. Perputaran sebanyak Nr-1 kali, dan proses yang dilakukan dalam proses ini yakni: *SubByte, ShiftRow, MixColumn,* dan *AddRoundkey*.

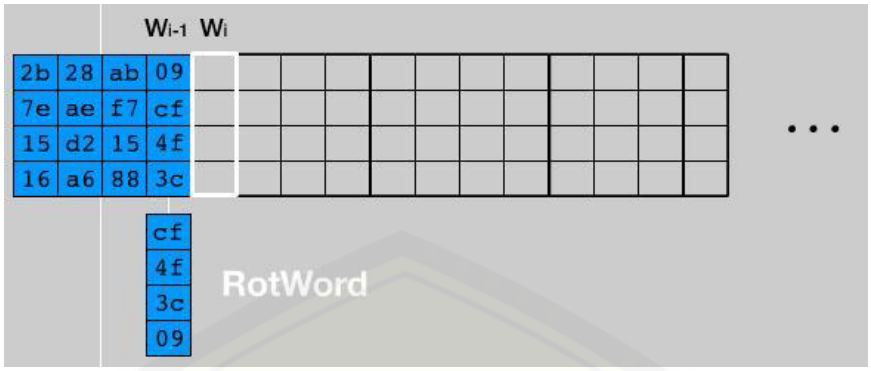
c. Proses putaran terakhir dengan tahapan *SubByte, ShiftRow,* dan *AddRoundkey*.

Berikut merupakan contoh penerapan algoritma AES-128 terhadap sebuah naskah (*plaintext*) yang telah dirubah terlebih dahulu dalam bentuk bilangan *hexadecimal* yang disusun menjadi matriks berordo 4x4 beserta kuncinya yang juga telah mengalami konversi seperti pada *plaintext* yang digambarkan dalam gambar 2.3.



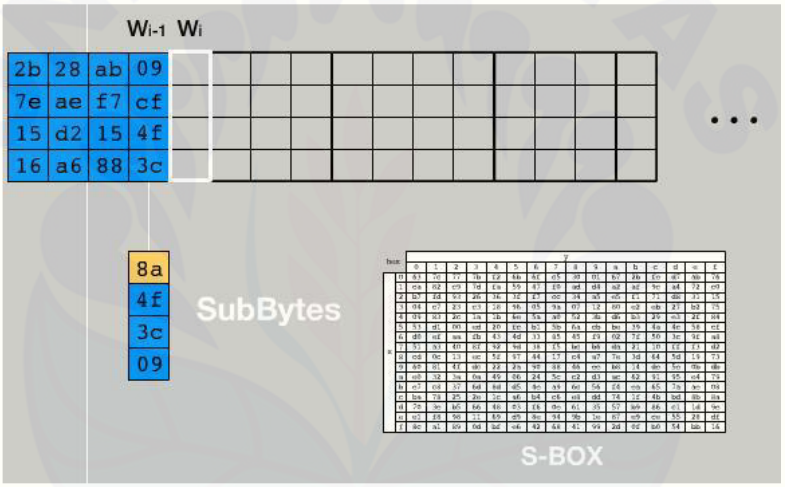
Gambar 2. 3 *Plaintext* dan *cipher key* dalam bentuk matriks 4x4

Pada tahap selanjutnya adalah tahap *key schedule* sebanyak 10 kali perputaran. Langkah pertama yang dilakukan adalah merubah posisi dari kolom ke-4, pada kolom ke-4 bilangan (09, cf, 4f, 3c) dirubah menjadi (cf, 4f, 3c, 09). Perubahan posisi tersebut disebut *RotWord.* Ilustrasi dari perubahan posisi tersebut terdapat pada gambar 2.4.



Gambar 2. 4 Perubahan posisi pada kolom terakhhir

Hasil dari *RotWord* masing-masing disubtitusikan terhadap tabel *S-Box* (lampiran B). Proses ini dinamakan proses *SubBytes* seperti pada gambar 2.5.

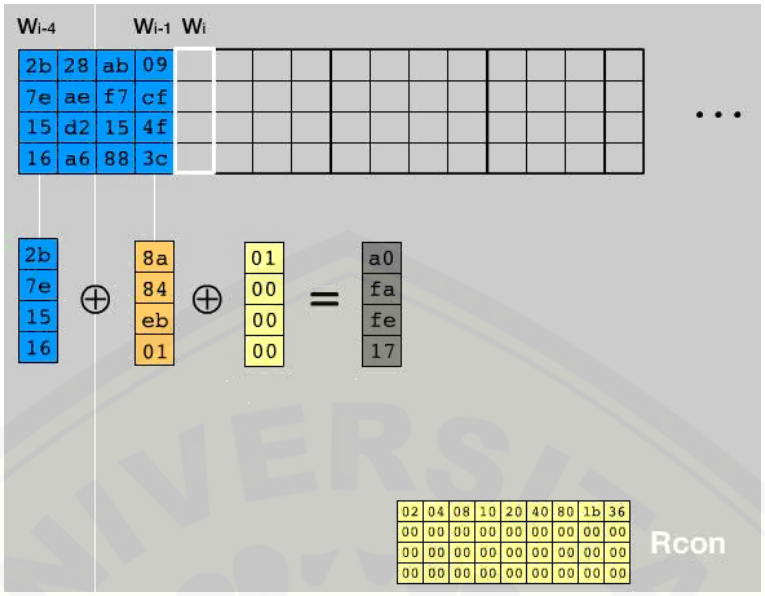


Gambar 2. 5 Proses subtitusi terhadap tabel *S-Box*

Setelah *Rotword* disubtitusikan terhadap tabel *S-Box*, maka kolom pertama dari array kunci dijumlahkan menggunakan operasi bilangan XOR dengan hasil *SubBytes* dan Rcon.

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Gambar 2. 6 Tabel Rcon

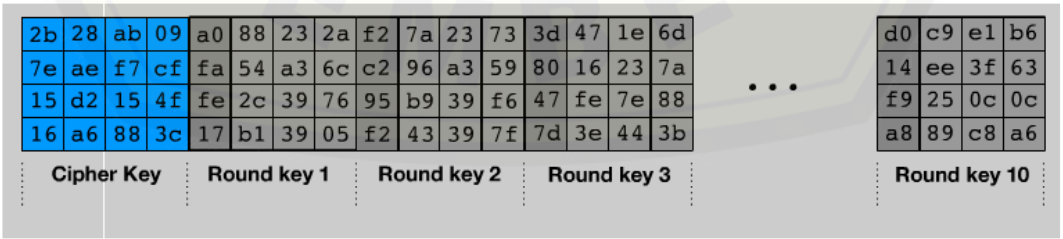


Gambar 2. 7 Proses operasi XOR

Hasil dari operasi perhitungan XOR ditempatkan pada kolom pertama dari *RoundKey 1*, seperti pada gambar 2.8.

Gambar 2. 8 Hasil *RoundKey 1*

Setelah mendapatkan hasil dari *RoundKey 1* dilakukan urutan proses yang sama dari *RotWord*, *SubBytes* dan operasi perhitungan XOR untuk mencari *RoundKey 2* sampai dengan *RoundKey 10.*



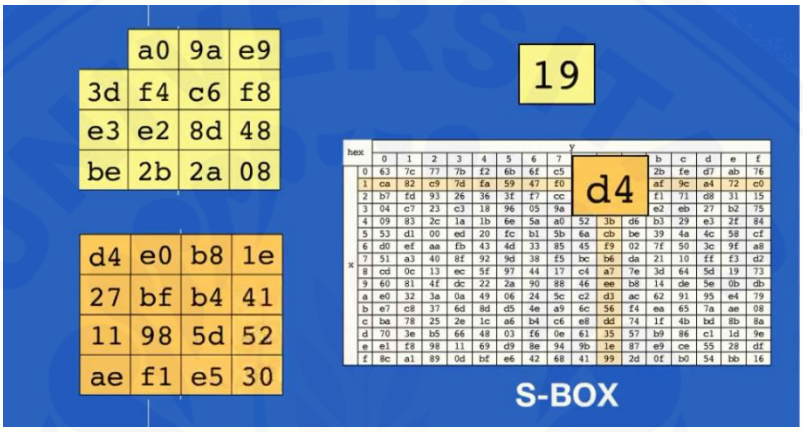
Gambar 2. 9 Hasil *key schedule*

Setelah proses *key schedule* selesai dilakukan dan menghasilkan 10 array *RoundKey*, maka dilakukan proses selanjutnya yaitu proses transformasi naskah

atau *plaintext* sehingga membentuk array bilangan *hexadecimal* yang acak. Beberapa proses yang dilakukan antara lain *SubBytes*, *ShiftRows*, *MixColumn* dan *AddRoundKey*.

- *SubBytes*

Pada proses *SubBytes* kali ini naskah atau *plaintext* yang sebelumnya sudah dikonversi dalam bilangan *hexadecimal* masing-masing disubtitusikan dengan menggunakan tabel *S-Box* seperti pada gambar 2.10.



Gambar 2. 10 Proses transformasi *SubBytes*

- *ShiftRows*

Setelah naskah disubtitusikan dengan tabel *S-Box* tahap selanjutnya adalah *ShiftRows*. Array matriks hasil dari transformasi SubBytes ditransformasikan dengan cara menggeser baris ke-2, ke-3 dan ke-4. Baris ke-2 bergeser sejauh 1 *byte*, baris ke-3 bergeser sejauh 2 *byte* dan baris ke-4 bergeser sejauh 3 *byte*. Gambar 2.11 menjelaskan tentang transformasi *ShiftRows*.

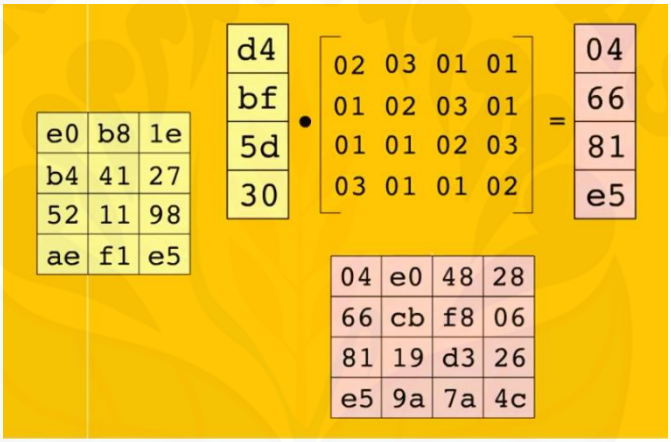
Gambar 2. 11 Proses transformasi *ShiftRows*

- *MixColumn*

Setelah didapatkan hasil dari transformasi *ShiftRows* maka dilakukan proses transformasi selanjutnya yakni *MixColumn*. Pada proses ini dilakukan operasi perkalian matriks pada setiap kolom dari hasil transformasi *ShiftRows* dengan polinom *a(x)* mod (x^4+1). *a(x)* yang ditetapkan adalah *a*(*x*) = {03}*x*3 + {01}*x*2 + {01}*x* + {02}.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

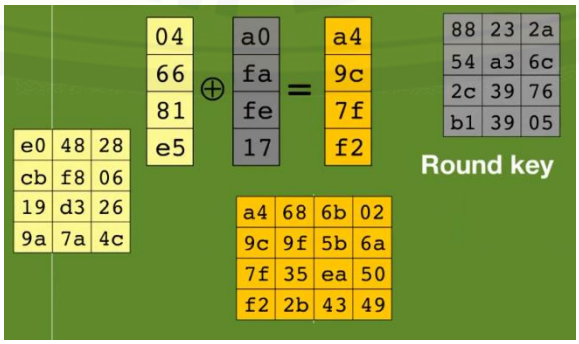
Implementasi *MixColumn* dijelaskan pada gambar 2.12.



Gambar 2. 12 Proses transformasi *MixColumn*

- *AddRoundKey*

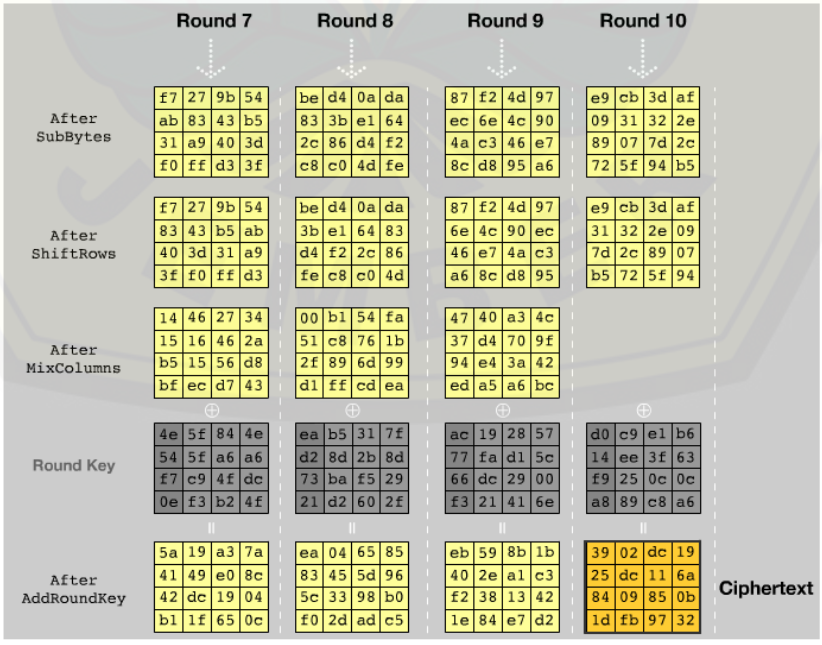
Proses selanjutnya yakni *AddRoundKey*. Pada tahap ini dilakukan sebuah operasi perhitungan XOR terhadap hasil dari transformasi *MixColumn* dengan *round key*. Proses transformasi *AddRoundKey* diperlihatkan pada gambar 2.13.



Gambar 2. 13 Proses transformasi *AddRoundKey*

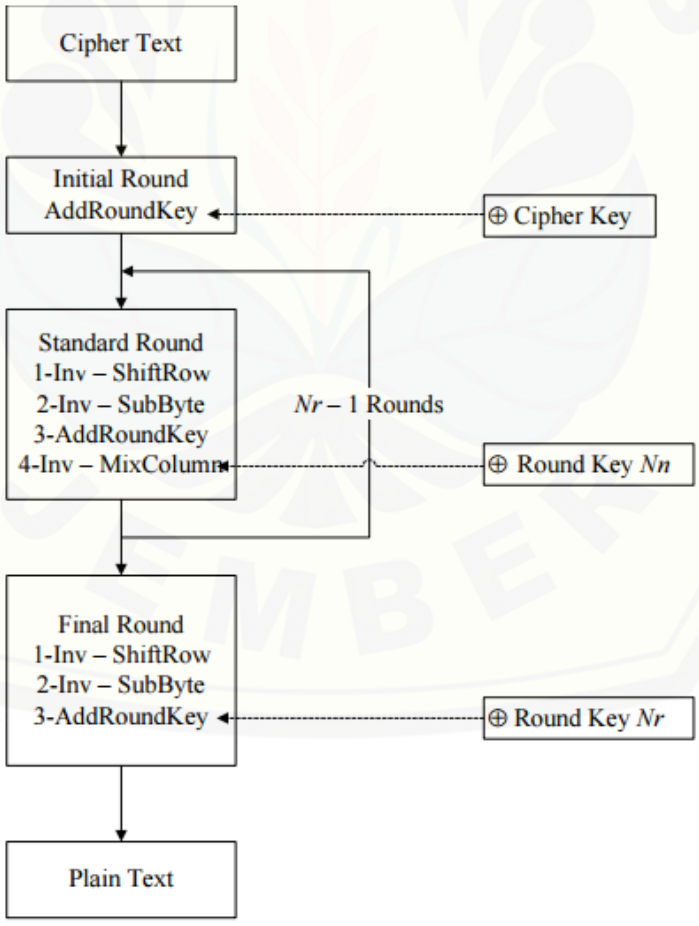
Tahapan-tahapan *SubBytes*, *ShiftRows*, *MixColumn* dan *AddRoundKey* diterapkan juga pada *round* ke-2 sampai dengan *round* ke-9. Sedangkan pada *round* ke-10 tahapan yang dilakukan adalah *SubBytes*, *ShiftRows* dan *AddRoundKey*. Hasil dari *round* ke-2 sampai dengan *round* ke-10 diperlihatkan pada gambar 2.14 dan gambar 2.15.

Gambar 2. 14 Hasil dari *round* ke-2 sampai dengan *round* ke-6



Gambar 2. 15 Hasil dari *round* ke-7 sampai dengan *round* ke-10

Pada proses dekripsi menggunakan algoritma AES-128, dekripsi dapat dilakukan menggunakan *inverse* transformasi dengan menggunakan *inverse* dari urutan transformasi pada proses enkripsi, karena setiap transformasi yang dilakukan dalam enkripsi AES mempunyai *inverse* (Kromodimoeldjo, 2010). Dekripsi dilakukan dari *state* pertama yaitu *state AddRounkey*, pada tahap ini *cipher key* mengalami perputaran sebanyak 10 kali seperti yang terjadi pada proses enkripsi, perbedaannya proses perputaran ini menghasilkan hasil yang berkebalikan dari proses enkripsi. Setelah itu, *state* hasil proses *AddRoundkey* ditransformasikan terhadap *inverse ShiftRows*, *inverse SubBytes*, *AddRoundkey*, dan *inverse MixColumns.* Proses tersebut berlangsung selama 9 perputaran. Sementara pada perputaran kesepuluh proses tersebut dilakukan tanpa adanya tranformasi *inverse MixColumns.* Proses dekripsi AES-128 dijelaskan pada gambar 2.3.



Gambar 2. 16 Diagram Alir Dekripsi AES-128 (Ilyas, 2014)

2.4 Android

Android adalah sistem operasi berbasis linux yang digunakan pada peragkat mobile. Android merupakan *open source* yang memungkinkan para pengembang menciptakan aplikasi mereka secara gratis pada sistem operasi ini.

Android merupakan *platform* terbuka, artinya *platform* yang memberikan kebebasan bagi para pengembang untuk melakukan pengembangan sesuai dengan yang dinginkan oleh pihak pengembang *platform* android tersebut. Bagi para pengembang mengunakan *tools* yang biasa disebut dengan IDE, beberapa IDE yang sering digunakan adalah *Eclipse* dan *Android Studio*.

2.5 SMS (*Short Message Service*)

SMS adalah layanan pesan teks yang merupakan komponen dari sistem komunikasi *mobile*, menggunakan protokol komunikasi yang terstandar dan memungkinkan pertukaran pesan singkat antar pengguna perangkat telepon seluler (Nishika, 2015).

BAB 3. METODOLOGI PENELITAN

Pada bab ini berisikan tentang pembahasan alur penelitian yang diterapkan pada pembuatan aplikasi *SMS Security* serta pengimplemetasian algoritma AES-128 pada aplikasi *SMS Security*.

3.1 Tahap Pengumpulan Data

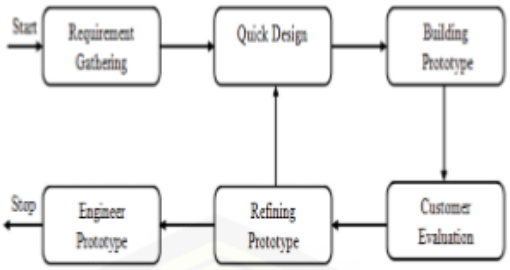
Tahap pengumpulan data dilakukan dengan cara mencari data primer dan sekunder yang dibutuhkan untuk mengimplementasi algoritma AES-128 pada aplikasi SMS security. Data primer diperoleh dari hasil wawancara dari beberapa sampel orang yang menggunakan *Android Mobile* sebagai alat telekomunikasi. Sedangkan data sekunder diperoleh dengan cara studi literatur dari penelitian-penelitian terdahulu di berbagai jurnal, buku, skripsi dan *e-book*. Studi literatur dibutuhkan untuk menunjang pemahaman dan pengetahuan penulis tentang materi, konsep, teori, dan metode apa yang diperlukan dalam proses pengerjaan tugas akhir ini.

3.2 Jenis Penelitian

Pada penelitian ini jenis penelitian yang digunakan ialah penelitian pengembangan. Dikarenakan pada penelitian ini membuat dan mengembangkan suatu produk, dan penelitian ini bukan dimaksudkan untuk menemukan suatu teori atau mengujikan teori tersebut dalam bentuk eksperimen.

3.3 Tahap Perancagan

Metode perancangan aplikasi yang digunakan dalam penelitian ini yakni SDLC (*Software Development Life Cycle*) dengan model *prototype*. Model *prototype* memungkinkan *user* mengevaluasi *software* yang dibuat dengan *prototype* dengan mencobacanya secara langsung daripada dengan penjelasan dengan hanya mengira-ngira produk yang dihasilkan dari penggambaran desain di atas kertas (Kumar, 2013). *Prototyping* memiliki beberapa keuntungan diantaranya : *designer* dan *developer* dapat mendapatkan *feedback* dari *user* pada saat penggarapan *project* berlangsung. *Client* dan *developer* dapat membandingkan apakah *software* yang dibuat cocok dengan spesifikasi awal yang telah disetujui. Alur perancangan model *prototype* dapat dilihat pada gambar 3.1.



Gambar 3. 1 *Prototype Model* (Kumar, 2013)

3.3.1. Desain Perancangan Sistem

Pembuatan desain sistem pada penelitian ini menggunakan *Unified Modeling Language* (UML) yang dirancang dengan konsep *Object Oriented Design*. Pemodelan UML yang digunakan sebagai berikut:

1. *Use Case Diagram*

Use case adalah model yang menggambarkan fungsi atau tugas yang dilakukan oleh *user*. *Use case* dapat digunakan untuk menggambarkan *job specification* dan *job description* serta hubungan antar *job*.

2. *Scenario*

Scenario berfungsi untuk menjelaskan alur sistem dari fitur yang ada pada *job specification* dan *job description* yang ada pada *use case diagram*. *Scenario diagram* menjelaskan alur sistem dan juga keadaan yang akan terjadi ketika terjadi suatu *event* yang dilakukan oleh aktor.

3. *Activity Diagram*

Activity diagram digunakan untuk menggambarkan suatu aktivitas dalam sebuah sistem. *Activity diagram* memiliki fungsi yang sama dengan *scenario* akan tetapi *activity diagram* digambarkan dalam sebuah diagram alir.

4. *Sequence Diagram*

Digunanakan untuk menggambarkan aliran logika antar objek yang mengindikasikan komunikasi antar objek di dalam sistem yang tersusun sesuai dengan rangkaian waktu.

5. *Class Diagram*

Menggambarkan struktur dan deskripsi class serta hubungan antar *class*, sehingga memudahkan dalam proses *coding.*

3.3.2. Pembuatan Prototype

Pada tahap ini mengimplementasikan desain yang akan menjadi sebuah *prototype* aplikasi berbasis android untuk pengamanan saat bertukar pesan di SMS dan mengimplentasikan algoritma kriptografi AES-128. Hal-hal yang dilakukan pada tahapan ini adalah menuliskan kode pada IDE *Android Studio* sebagai *tool* pembuatan aplikasi *SMS security*.

3.3.3. *Evaluation*

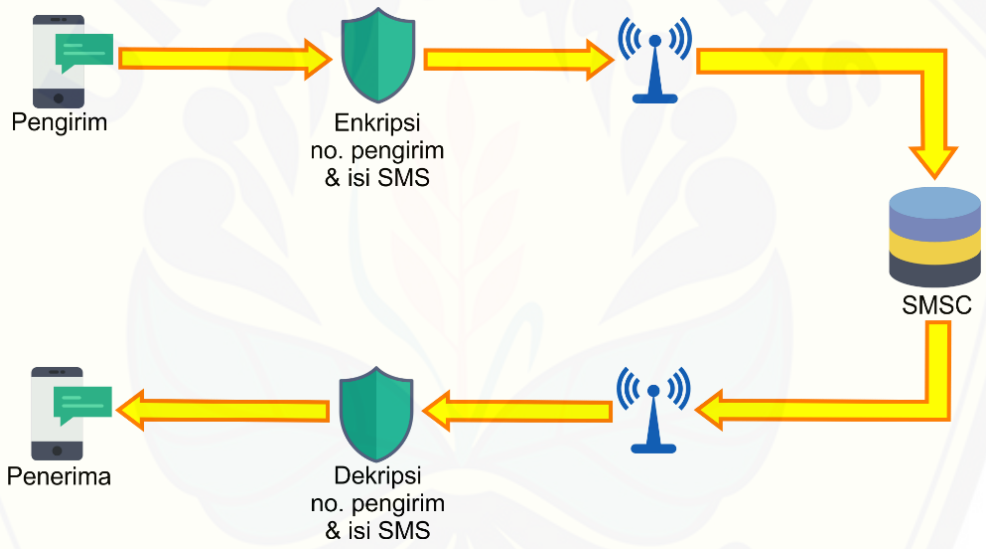
Pada tahap ini dilakukan pengujian terhadap aplikasi yang dibuat menggunakan metode *black box testing*, dimana pengujian ini menguji fitur-fitur fungsional yang terdapat pada aplikasi atau sistem yang dibuat. Dalam metode pengujian ini *tester* mengerti apa saja yang dilakukan oleh program yang sedang diuji (Kurniawati, 2018).

BAB 4. DESAIN DAN PERANCANGAN SISTEM

Pada bab ini akan menjelaskan tentang alur dan proses perancangan serta pengimplementasian algoritma AES-128 dalam pembuatan aplikasi *SMS Security*.

4.1 Gambaran Umum Sistem

Pada penelitian ini sistem yang akan dibangun berbasis android. Sistem ini menerapkan algoritma AES-128 sebagai pengamanan data dalam SMS yang dikirim maupun diterima. Pada sistem ini algoritma AES-128 mengenkripsi isi SMS yang dikirim maupun diterima. Arsitektur dari sistem yang akan dibangun sepert gambar 4.1.

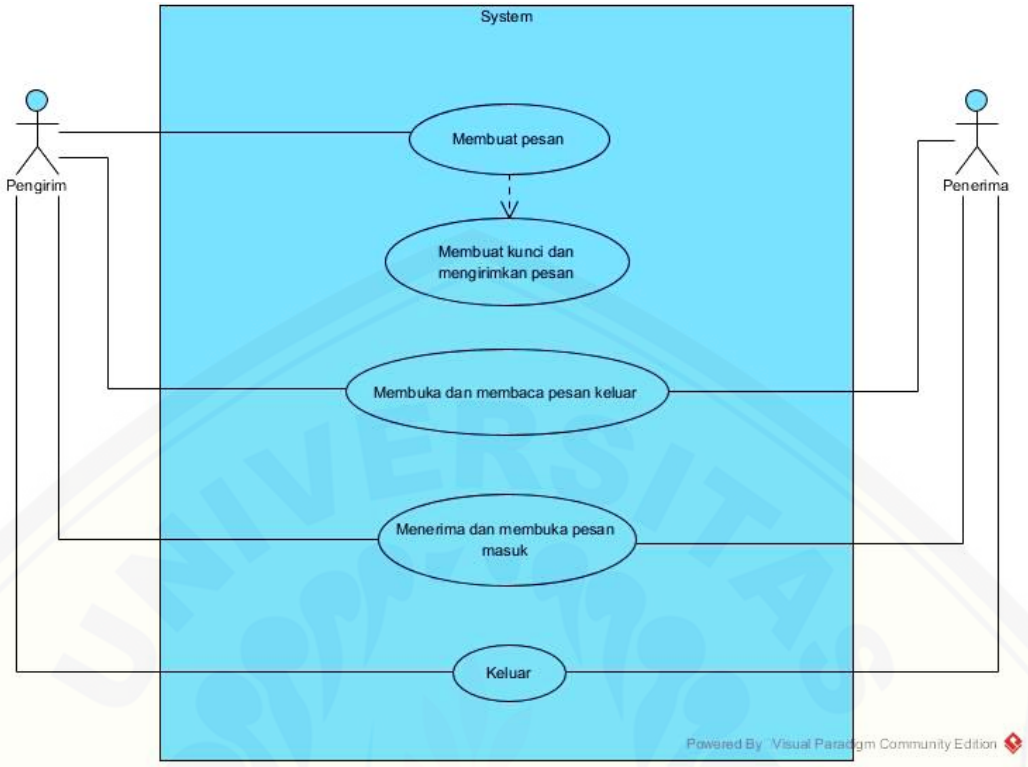


Gambar 4. 1 Arsitektur Sistem SMS Security

4.2 *Usecase Diagram*

Usecase diagram pengimplementasian algoritma AES-128 pada aplikasi *SMS Security* berbasis *Android mobile* berfungsi menggambarkan fitur apa saja yang terdapat pada SMS Security saat dijalankan atau dioperasikan oleh para user, yang terdapat algoritma AES-128 di dalamnya yang berfungsi sebagai pengaman data SMS antara pengirim dan penerima SMS.

Berikut adalah *usecase diagram* yang digunakan pada aplikasi *SMS Security* yang digambarkan pada gambar 4.2.



Gambar 4. 2 *Usecase diagram SMS Security*

Tabel 4. 1 Definisi *usecase SMS Security*

No	*Usecase*	Definisi
1.	Membuat pesan dan membuat kunci	Pengirim membuat pesan yang akan dikirimkan kepada penerima sekaligus membuat kunci pada pesan yang an dikirimkan
2.	Membuka dan membaca pesan keluar	Proses dimana pengguna membuka dan membaca pesan yang telah dikirimkan ke penerima
3.	Membuka dan membaca pesan masuk	Proses dimana pengguna membuka dan juga membaca pesan masuk yang telah diterima dari pengirim
4.	Keluar	Pengguna aplikasi SMS Security keluar dari aplikasi dan mengakhiri proses

Tabel 4. 2 Definisi aktor *usecase SMS Security*

No.	Aktor	Definisi
1.	Pengirim	Pengguna aplikasi *SMS Security* yang bertugas untuk mengirimkan pesan yang telah terenkripsi
2.	Penerima	Pengguna aplikasi *SMS Security* yang bertugas untuk menerima pesan yang telah terenkripsi

4.3 *Usecase Scenario*

Usecase Scenario adalah uraian dari *Usecase Diagram*, dimana setiap *usecase* memiliki fungsi-fungsi tersendiri yang berjalan pada sistem. Adapun *usecase scenario* yang ada pada aplikasi *SMS Security* seperti berikut:

a. Membuat pesan dan membuat kunci

Pada *usecase scenario* "Membuat pesan dan membuat kunci" ini menjelaskan tentang alur dimana *sender* akan membuat pesan yang akan dikirimkan oleh calon penerima pesan, dimana pesan tersebut juga akan dimasukkan sebuah kata sandi atau kunci untuk mengenkripsi isi dari pesan yang akan dikirimkan tersebut. Berikut merupakan *usecase scenario* dari *usecase* "Membuat pesan dan membuat kunci".

Tabel 4. 3 *Usecase scenario* membuat pesan dan membuat kunci

Nama	Membuat pesan dan membuat kunci
Aktor	Pengirim
Entry Condition	Sistem menampilkan form membuat pesan
Exit Condition	Sistem mengirimkan pesan yang telah terenkripsi kepada penerima
Skenario Normal	
1. Pengirim memilih tombol "Buat Pesan"	
	2. Sistem menampilkan halaman *form* "Buat Pesan"

3. Pengirim memilih kontak dengan menekan tombol “Kontak”	
\	4. Sstem me-*redirect* pada halaman kontak
5. Pengirim memilih kontak pada *list* kontak	
	6. Sistem menampilkan kembali halaman *form* “Buat Pesan”
7. Pengirim memasukka isi pesan dan juga memasukkan kunci	
8. Pengirim menekan tombol “Kirim”	
	9. Sistem megirimkan pesan terenkripsi pada penerima
Skenario Alternatif (Megirim Pesan)	
8. Pengirim menekan tombol “Kirim” dan tidak memasukkan apapun ke dalam *field*	
	9. Sistem menampilkan *toasr message*

b. Membuka dan membaca pesan keluar

Pada *usecase scenario* yang kedua ini menjelaskan tentang alur kegiatan dimana *user* akan melihat pesan-pesan yang telah terkirim. Dan untuk membaca pesan yang dienkripsi *user* akan memasukkan kunci yang sebelumnya telah digunakan untuk mengirimkan pesan kepada pemerima yang dituju. *Usecase scenario* dari *usecase* ”Membuka dan membaca pesan keluar” tertera pada tabel 4.4.

Tabel 4. 4 *Usecase scenario* membuka dan membaca pesan keluar

Nama	Membuka dan membaca pesan keluar
Aktor	Pengirim, Penerima
Entry Condition	Sistem menyimpan pesan keluar yang telah dikirimkan oleh pengirim
Exit Condition	Sistem menampilkan pesan keluar
Skenario Normal	
1. Pengguna menekan tombol "Pesan Keluar" untuk membuka pesan	
	2. Sistem menampilkan *list* pesan keluar
3. Pengguna memilih pesan keluar yang terdapat dalam *list*	
	4. Sistem menampilkan isi pesan yang masih terenkripsi
5. Pengguna memasukkan kunci *text field* kunci	
6. Pengguna menekan tombol "Dekrip"	
	7. Sistem menampilkan isi pesan yang telah terdekripsi
Skenario Alternatif (Memasukkan kunci yang salah)	
5. Pengguna memasukkan kunci yang salah pada *text field* kunci	
6. Pengguna menekan tombol "Dekrip"	
	7. Sistem menampilkan pesan "Pesan tidak terdekrip" pada kolom pesan terdekripsi
Skenario Alternatif (kolom dekrip kosong)	

5. Pengguna menekan tombol "Dekrip" tanpa memasukkan kunci pada *text field* kunci	
	6. Sistem menampilkan toast message "Mohon masukkan kunci"

c. Membuka dan membaca pesan masuk

Pada *usecase* scenario ketiga ini dijelaskan alur kegiatan yang dilakukan oleh *user* saat akan membuka pesan-pesan yang telah diterima. Untuk membuka pesan yang diterima dengan menggunakan enkripsi *user* memasukkan kunci yang telah diketahui dari pengirim sebelumnya. Scenario dari *usecase* "Membuka dan membaca pesan masuk" ini tertera pada tabel 4.5.

Tabel 4. 5 *Usecase scenario* membuka dan membaca pesan masuk

Nama	Membuka dan membaca pesan masuk
Aktor	Pengirim, Penerima
Entry Condition	Sistem menyimpan pesan masuk yang telah dikirimkan oleh pengirim
Exit Condition	Sistem menampilkan pesan keluar
Skenario Normal	
1. Pengguna menekan tombol "Pesan Masuk" untuk membuka pesan	
	2. Sistem menampilkan *list* pesan masuk
3. Pengguna memilih pesan masuk yang terdapat dalam *list*	
	4. Sistem menampilkan isi pesan yang masih terenkripsi
5. Pengguna memasukkan kunci *text field* kunci	

6. Pengguna menekan tombol "Dekrip"	
	7. Sistem menampilkan isi pesan yang telah terdekripsi
Skenario Alternatif (Memasukkan kunci yang salah)	
5. Pengguna memasukkan kunci yang salah pada *text field* kunci	
6. Pengguna menekan tombol "Dekrip"	
	7. Sistem menampilkan pesan "Pesan tidak terdekrip" pada kolom pesan terdekripsi
Skenario Alternatif (kolom dekrip kosong)	
6. Pengguna menekan tombol "Dekrip" tanpa memasukkan kunci pada *text field* kunci	
	7. Sistem menampilkan toast message "Mohon masukkan kunci"

d. Keluar

Pada *usecase scenario* "Keluar" ini dijelaskan alur kegiatan yang akan dilakukan oleh *user* ketika hendak keluar dari aplikasi *SMS Security*. *Usecase scenario* "Keluar" tertera pada Tabel 4.6.

Tabel 4. 6 *Usecase scenario* keluar

Nama	Keluar
Aktor	Pengirim, Penerima
Entry Condition	Sistem menampilkan halaman awal aplikasi

Exit Condition	Sistem mengakhiri proses yang berjalan
Skenario Normal	
1. Pengguna menekan tombol "Keluar"	
	2. Sistem mengakhiri proses yang berjalan dan keluar

4.4 Activity Diagram

Activity diagram adalah gambaran alur dari aksi yang dilakukan oleh *user* dan reaksi sistem pada saat sistem berjalan. *Activity diagram* dalam pebuatan aplikasi SMS *Security* yang menerapkan metode AES-128 terbagi dalam beberapa fitur yang digambarkan sebelumnya pada *use case diagram* dan *scenario*. Berikut merupakan *activity diagram* dari aplikasi SMS *Security*.

a. *Activity diagram* membuat pesan dan membuat kunci

Activity diagram berikut menjelaskan tentang alur grafis aktivitas yang hendak dilakukan oleh *user* untuk membuat sebuah pesan dan kuncinya utuk mengankan isi dari pesan tersebut. *Activity diagram* dari *usecase* membuat pesan dan membuat kunci dapat dilihat pada lampiran A (Lampiran *Activity Diagram*).

b. *Activity diagram* membuka dan membaca pesan masuk

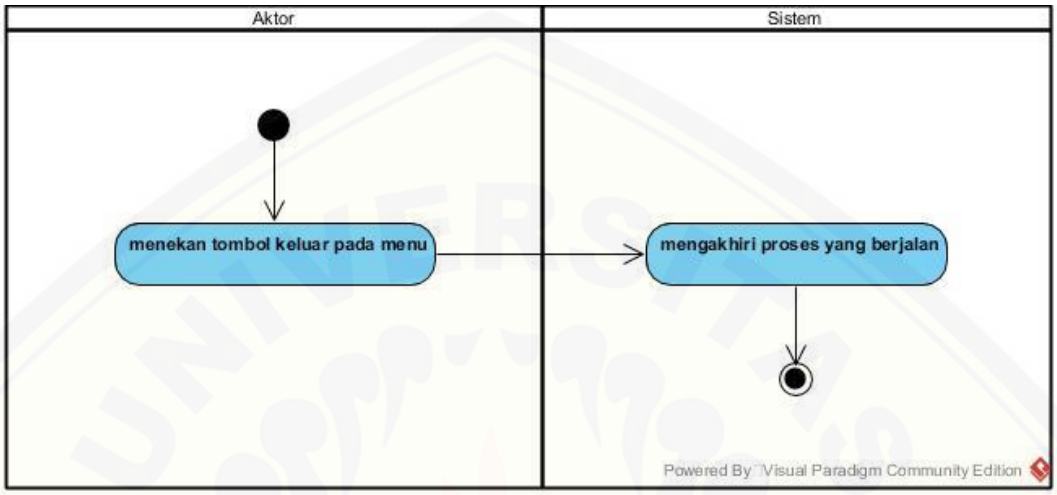
Activity diagram dari *usecase* "Membuka dan membaca pesan masuk" berikut menjelaskan tentang alur grafis aktivitas saat *user* hendak membuka pesan yang telah masuk pada perangkat telepon genggamnya melalui aplikasi *SMS Security*. *Activity diagram* membuka dan membaca pesan masuk tertera pada lampiran A (Lampiran *Activity Diagram*).

c. *Activity diagram* membuka dan membaca pesan keluar

Activity diagram dari *usecase* "Membuka dan membaca pesan keluar" berikut menjelaskan tentang alur grafis aktivitas saat *user* hendak membuka pesan yang telah terkirim pada perangkat telepon genggamnya melalui aplikasi *SMS Security*. *Activity diagram* membuka dan membaca pesan keluar tertera pada lampiran A (Lampiran *Activity Diagram*).

d. *Activity diagram* Keluar

Activity diagram dari *usecase* "Keluar" berikut menjelaskan tentang alur grafis aktivitas saat *user* hendak keluar dari aplikasi *SMS Security*. *Activity diagram* keluar tertera pada Gambar 4.3.



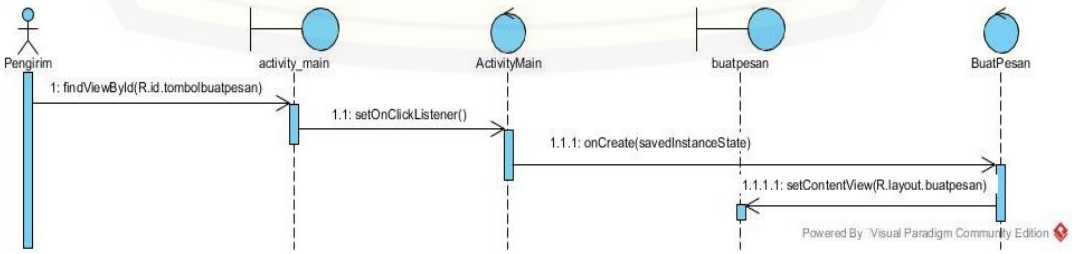
Gambar 4. 3 *Activity diagram* keluar

4.5 *Sequence Diagram*

Perancangan selanjutnya setelah *Activity Diagram* yaitu *Sequence Fiagram. Sequence Diagram* adalah diagram yang memperlihatkan interaksi pada objek-objek yang terdapat dalam sistem dalam rentang waktu tertentu. Berikut merupakan *Sequence Diagram* yang diterapkan dalam aplikasi SMS *Security:*

a. Membuat pesan dan membuat kunci

Pada *sequence diagram* ini dijelaskan alur sistem ketika *user* melakukan kegiatan membuat pesan serta membuat kunci. Kedua kegiatan tersebut dilakukan pada satu *interface*. *Sequence diagram* dari membuat pesan dan membuat kunci dapat dilihat pada gambar 4.4.



Gambar 4. 4 *Sequence diagram* membuat pesan dan membuat kunci

b. Membuka dan membaca pesan keluar

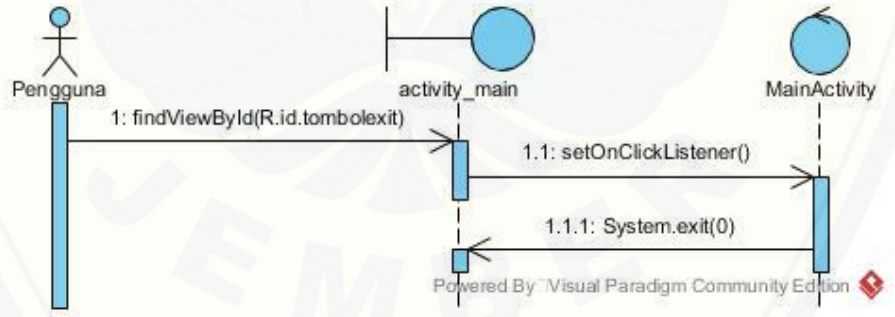
Pada *sequence* membuka dan membaca pesan keluar ini menjelaskan alur sistem ketika *user* melakukan kegiatan membuka dan membaca pesan yang telah terkirim yang terdapat pada telepon genggam pengguna melalui aplikasi *SMS Security*. *User* memasukkan kunci untuk melihat pesan yang terenkripsi pada halaman lihatpesan. *Sequence diagram* dari *usecase* membuka dan membaca pesan keluar tertera pada lampiran B (Lampiran *Sequence Diagram*).

c. Membuka dan membaca pesan masuk

Pada *sequence* membuka dan membaca pesan keluar ini menjelaskan alur sistem ketika *user* melakukan kegiatan membuka dan membaca pesan yang telah masuk yang terdapat pada telepon genggam pengguna melalui aplikasi *SMS Security*. *User* memasukkan kunci untuk melihat pesan yang terenkripsi pada halaman lihatpesan. *Sequence diagram* dari *usecase* membuka dan membaca pesan keluar tertera pada lampiran B (Lampiran *Sequence Diagram*).

d. Keluar

Pada *sequence* kali ini dijelaskan tentang alur system saat *user* hendak keluar dari aplikasi *SMS Security*. *Sequence diagram* dari *usecase* "Keluar" tertera pada gambar 4.5.



Gambar 4. 5 *Sequence diagram* keluar

4.6 Class Diagram

Proses desain selanjutnya yaitu *Class Diagram*. *Class Diagram* adalah diagram yang menggambarkan struktur *class* yang saling berhubungan yang terdapat dalam sistem. Struktur *Class Diagram* yang digunakan dalam aplikasi SMS *Security* tertera pada lampiran C (Lampiran *Class Diagram*).

4.7 Implementasi Perancangan

Setelah melewati proses perancangan desain, tahapan selanjutnya dalam penelitian ini yaitu pengimplementasian desain ke dalam bahasa pemrograman. Bahasa pemrograman yang dipakai dalam penelitian ini adalah bahasa pemrograman Java, yang mana bahasa pemrograman tersebut digunakan pada *IDE Android Studio*. Berikut merupakan alur dari pengimplementasian desain ke dalam bahasa pemrograman Java pada *IDE Android Studio.*

4.7.1. Alur Pembuatan Pesan dan Kunci

Proses pembuatan kunci dan isi pesan:

```
send.setOnClickListener{ew View.OnClickListener() {
      public void onClick(Viewv) {
            StringsecKey = textKunci.getText().toString();
            String pesan = text.getText().toString();
            String nomor = nomorKontak.getText().toString();

      if (nomor.length() >0 && secKey.length() > 0
            && pesan.length() > 0
            && secKey.length() == 16) {
                  byte[] enkripPesan = enkripSMS(secKey, pesan);
                  String pesanString = byteToHex(enkripPesan);
                  sendSMS(nomor, pesanString);
                  finish();
            }else {
                  Toast.makeTextBuatPesan.this, “Key must be 16
      characters”,
                        Toast.LENGTH_SHORT.show();
            }
}
```

Proses pengiriman pesan:

```
public static voidsendSMS(String nomor,String enkripPesan) {
            try{
                  SmsManager sms = SmsManager.getDefault();
                  ArrayList<String> parts =
      sms.divideMessage(enkripPesan);
                  Sms.sendMultipartTextMessage(nomor, null, parts,
      null, null);
            }catch (Exeption e) {
                  e.printStackTrace();
            }
}
```

4.7.2. Alur Proses Enkripsi

Proses konversi *array byte* ke *array hexadecimal:*

```
public static String byteToHex(byte[] b) {
	String hs = “”;
	String stmp = “”;
	for (int n = 0; n < b.length; n++) {
		Stmp= Integer.toHexString(i: b[n] & 0xFF);
		if (stmp.length() == 1)
			hs += (“0” + stmp);
		else
			hs += stmp;
	}
	Return hs.toUpperCase();
}
```

Proses enkripsi pesan:

```
public static byte[] enkripSMS(String secKey, String pesan) {
	try{
		byte[] returnArray;
		Key kunci = generateKey(secKey);
		Cipherc = Cipher.getInstance(“AES”);
		c.init(Cipher.ENCRYPT_MODE,kunci);
		returnArray = c.doFinal(pesan.getBytes());

		return returnArray;
	} catch (Exception e) {
		e.printStackTrace();
		byte[] returnArray = null;
		return returnArray;
	}
}

private static Key generateKey(String secretKeyString) throws
Exception {
	Key kunci = new
	SecretKeySpec(secretKeyString.getBytes(), “AES”);
	return kunci;
}
```

4.7.3. Alur Proses Dekripsi

Proses dekripsi pesan:

```
public static byte[] decryptSMS(String secretKeyString, byte[]
encryptedMsg) throws Exception {
	Key key = generateKey(secretKeyString);
	Cipher c = Cipher.getInstance(“AES”);
	c.init(Cipher.DECRYPT_MODE, key);
	byte[] decValue = c.doFinal(encryptedMsg);
	return decValue;
}
```

```
private static Key generateKey(String secretKeyString) throws
Exception {
      Key key = new SecretKeySpec (secretKeyString.getBytes(),
      "AES");
      Return key2;
}
```

Proses konversi *array hexadecimal* ke *array byte:*

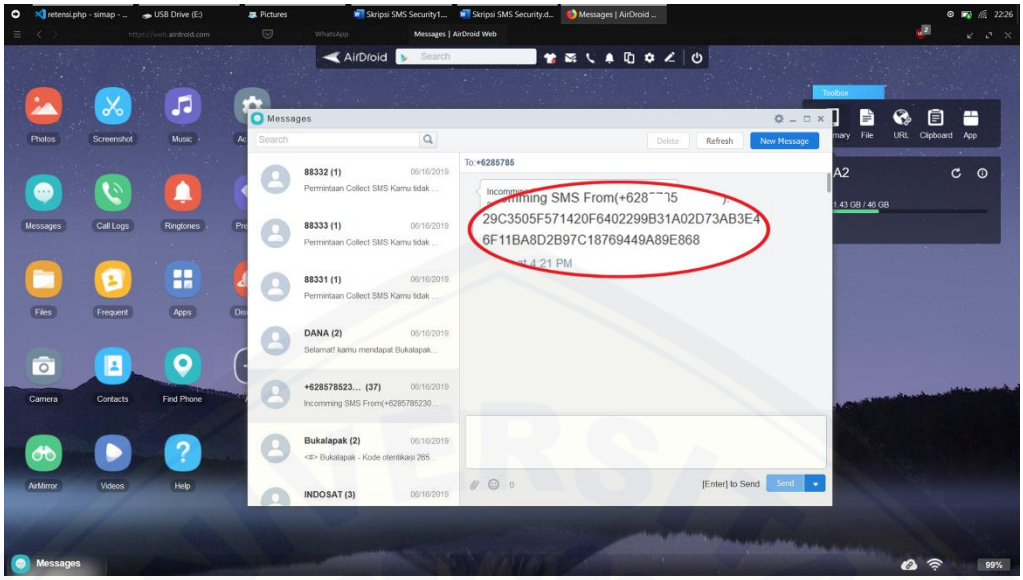
```
public static byte[] hex2byte(bte[] b) {
      if ((b.length %2) != 0)
            throw new IllegalArgumentException("hello");
      byte[] b2 = new byte [b.length / 2];
      for (int n = 0; n < b.length; n += 2) {
            String item = new String(b, n, 2) {
            b2[n / 2] = (byte) Integer.pareInt(item, 16);
      }
      Return b2;
}
```

4.8 Tahap Pengujian

Pada tahap ini aplikasi *SMS Security* diuji dengan menggunakan tiga macam cara. Cara pengujian yang digunakan yakni dengan menggunakan dua aplikasi *snooping* yakni aplikasi *AirDroid* dan *Call & SMS Forwarder* dan juga dengan metode black box testing.

4.8.1. Pengujian Menggunakan Aplikasi *AirDroid*

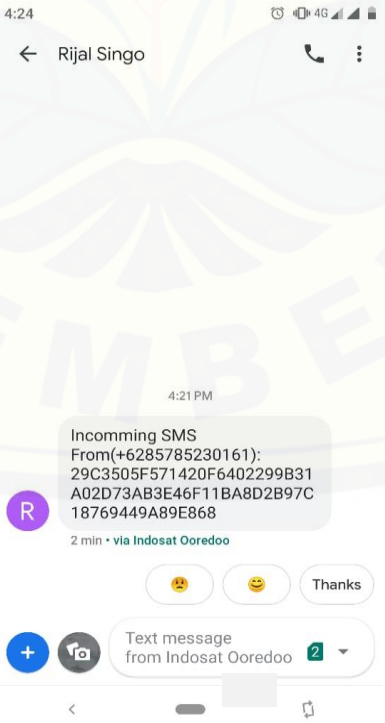
AirDroid merupakan aplikasi remote yang memungkinkan untuk digunakan sebagai software yang dapat memonitor isi file maupun informasi yang terdapat pada perangkat Android. Dalam penggunaannya sebagai software monitor, *AirDroid* harus dipasangkan dahulu pada perangkat Android dari orang yang ingin kita monitor perangkat selulernya, lalu bisa kita monitor melalui *website* dari *AirDroid*.



Gambar 4. 6 Tampilan *website AirDroid*

4.8.2. Pengujian Menggunakan Aplikasi *Call & SMS Forwarder*

Aplikasi *Call & SMS Forwarding* merupakan aplikasi snooping yang memungkinkan SMS ter-*forward* dengan otomatis ke nomor HP yang sudah diatur pada aplikasi ini untuk memonitor SMS dan juga aktivitas panggilan dari nomor sasaran.



Gambar 4. 7 Hasil *forward* SMS dari aplikasi *Call & SMS Forwarding App*

4.8.3. Pengujian Menggunakan *Black Box Testing*

Pada pengujian ini dilakukan beberapa pengujian apakah kebutuhan fungsional pada aplikasi *SMS Security* sudah terpenuhi atau belum. Hasil pengujin *black box* terdapat pada tabel 4.7.

Tabel 4. 7 Halaman pengujian *Black Box*

No.	Fitur	Aksi	Hasil	Kesimpulan
1.	Membuat pesan dan membuat kunci	Memilih tombol "Buat Pesan"	Menampilkan halaman form "Buat Pesan"	[✓] Berhasil [ ] Gagal
		a. Memilih kontak dengan menekan tomobol "Kontak" b. Memilih kontak yang tertera	Sistem mengambil data kontak yang ada di dalam penyimpanan	[✓] Berhasil [ ] Gagal
		a. Memasukkan isi pesan b. Memasukkan kunci c. Menekan tombol "Kirim"	Sistem mengirimkan pesan yang terenkripsi	[✓] Berhasil [ ] Gagal
2.	Membuka dan membaca pesan keluar	Menekan tombol "Pesan Keluar"	Sistem menampilkan *list* pesan keluar	[✓] Berhasil [ ] Gagal
		Memilih pesan yang tertera pada *list*	Sistem meampilkan isi pesan yang terenkripsi	[✓] Berhasil [ ] Gagal

		a. Memasukkan kunci pada *text field* "Kunci" b. Menekan tombol "Dekrip"	Sistem menampilkan isi pesan yang telah terankripsi	[✓] Berhasil [ ] Gagal
3.	Membuka dan membaca pesan masuk	Menekan tombol "Pesan Masuk"	Sistem menampilkan *list* pesan masuk	[✓] Berhasil [ ] Gagal
		Memilih pesan yang tertera pada *list*	Sistem meampilkan isi pesan yang terenkripsi	[✓] Berhasil [ ] Gagal
		c. Memasukkan kunci pada *text field* "Kunci" d. Menekan tombol "Dekrip"	Sistem menampilkan isi pesan yang telah terankripsi	[✓] Berhasil [ ] Gagal
4.	Keluar	Menekan tombol "Keluar"	Sistem mengakhiri proses dan keluar dari aplikasi	[✓] Berhasil [ ] Gagal

BAB 6. PENUTUP

Pada bab ini merupakan bagian terakhir dari penulisan skripsi yang berisi tentang kesimpulan dan saran yang didapat selama proses penelitian ini berlangsung. Kesimpulan yang telah didapat diharapkan dapat menjadi acuan dalam pengembangan lebih lanjut dari aplikasi *SMS Security*.

6.1 Kesimpulan

Berdasarkan pada hasil penelitian ini, diperoleh beberapa analisis yang didapat selama berlangsungnya penelitian ini. Kesimpulan dari penelitian yang telah dilakukan adalah sebagai berikut:

1. Implementasi kriptografi AES-128 dalam mengenkripsi isi pesan pada SMS yaitu dengan merubah kunci (*cipher key*) dan naskah (*plain text*) ke dalam bentuk *hexadecimal* yang diterima oleh penerima pesan sehingga pihak lain yang tidak mengetahui *cipher key* tidak bisa membaca isi dari pesan tersebut. Untuk membuka isi pesan dibutuhkan kunci yang telah diketahui terlebih dahulu oleh pengirim dan penerima.
2. Rancangan arsitektur dalam pembuatan aplikasi *SMS Security* menggunakan model *prototype* sebagai sebagai model pembangunan *project.* Untuk proses enkripsi diterapkan pada SMS dikirimkan oleh pengirim, dan proses dekripsi pesan dapat diakses oleh pengirim dan penerima melalui menu pesan ke;uar dan pesan masuk. *Database* yang digunakan oleh aplikasi *SMS Security* ini memakai *default database* yang dipakai oleh aplikasi SMS *default* dari sistem operasi Android.

6.2 Saran

Pengembangan lebih lanjut dari aplikasi *SMS Security* ini diharapkan dapat memberikan sajian *user interface* yang lebih menarik serta dapat dibangun pula untuk *platform mobile* lain seperti iOS, maupun *Windows Phone.*

DAFTAR PUSTAKA

Anwar, H. F., & Hastuti, K. (2015). PENERAPAN ALGORITMA DES DAN RC6 PADA APLIKASI ENKRIPSI SMS PADA ANDROID. *Jurnal Teknik Informatika*, 1-2.

Barbosa, L. S. (2015, September 21). *POWER8 in-core cryptography*. Retrieved Mei 25, 2019, from www.ibm.com: https://www.ibm.com/developerworks/library/se-power8-in-core-cryptography/index.html

I, E. R., Istiyadi, D., & Nurdiansyah, Y. (2014). IMPLEMENTASI ALGORITMA AES-128 PADA MOBILE LEARNING UNIVERSITAS JEMBER. *UNEJ Jurnal Program Studi Sistem Informasi*, 3-4.

Ilyas, I. A., & Widodo, S. (2014). KRIPTOGRAFI FILE MENGGUNAKAN METODE AES DUAL PASSWORD. *Jurnal Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2014)*, 264-267.

Jindan, H., Slamin, & Adiwijaya, N. O. (2015). Pesan Rahasia dengan Metode Kriptografi Elgamal pada Perangkat Android Mobile. *UNEJ Jurnal Program Studi Sistem Informasi*, 3-8.

Kromodimoeljo, S. (2010). *TEORI DAN APLIKASI KRIPTOGRAFI*. SPK IT Consulting.

Kumar, N., Zadgaonkar, A. S., & Shukla, A. (2013). EVOLVING A NEW SOFTWARE DEVELOPMENT LIFE CYCLE MODEL SDLC-2013 WITH CLIENT SATISFACTION. *International Journal of Soft Computing and Engineering (IJSCE)*, 216-217.

Nishika, & Yadav, R. K. (2013). CRYPTOGRAPHY ON ANDROID MESSAGE APPLICATIONS – A REVIEW. *International Journal of Soft Computing and Engineering (IJSCE)*, 362-363.

Nugroho, A. (2012). IMPLEMENTASI ALGORITMA CAESAR CIPHER ROT13 DAN BASE64 UNTUK ENKRIPSI DAN DEKRIPSI PESAN SMS PADA HANDPHONE BERBASIS ANDROID. *Jurnal STNIK ANIKOM Yogyakarta*, 4-5.

Peni, K. (2018, Oktober 29). *Pengujian Sistem*. Retrieved Mei 25, 2019, from medium.com: https://medium.com/skyshidigital/pengujian-sistem-52940ee98c77

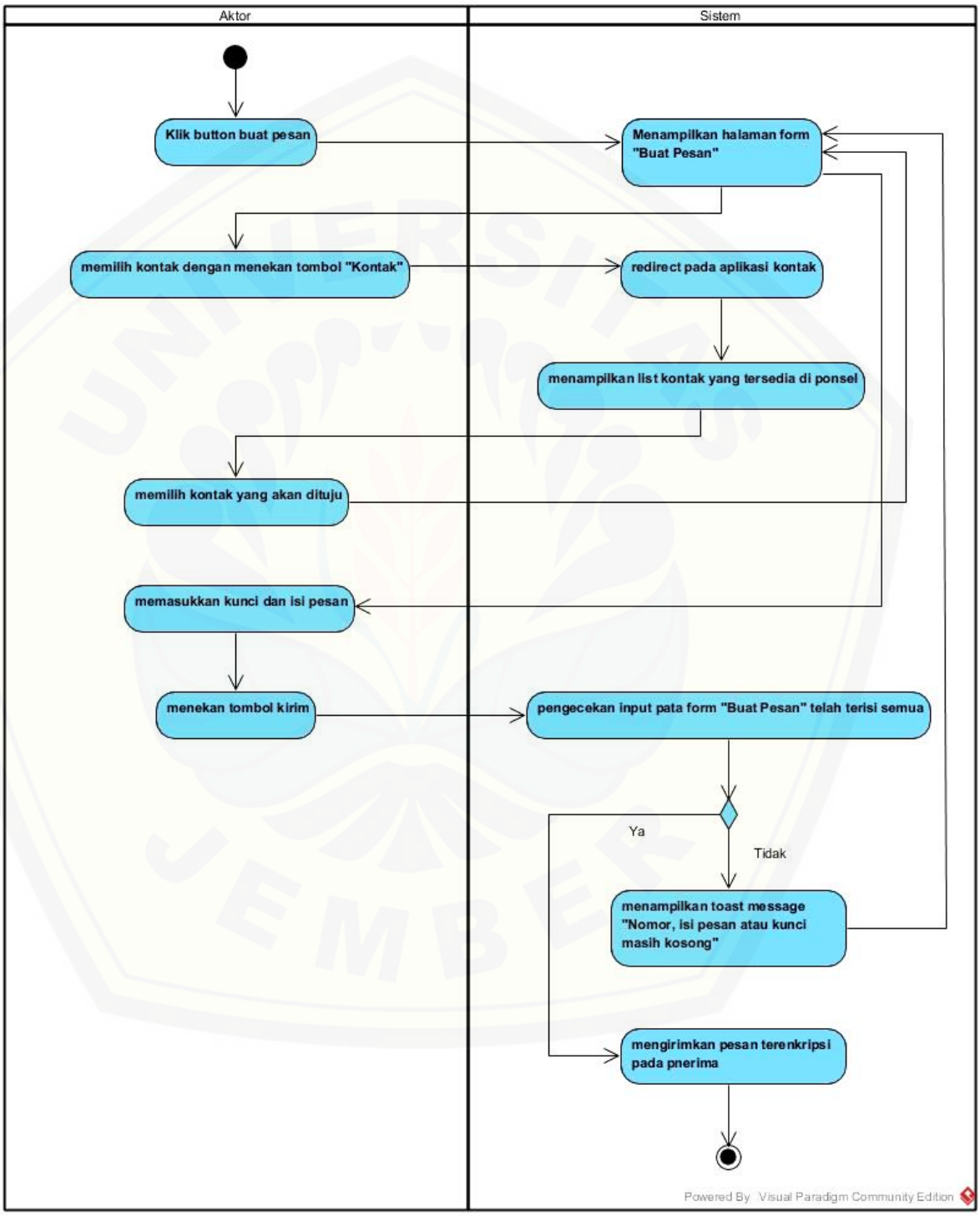
Triyuswoyo, Y., Ferdianti, F., Baskoro, D. A., Ambarwati, A., & Septiawan. (2014). IMPLEMENTASI ALGORITMA CAESAR, CIPHER DISK, DAN SCYTALE PADA APLIKASI ENKRIPSI DAN DEKRIPSI PESAN

SINGKAT, LumaSMS. *Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2014)*, 469-470.

LAMPIRAN

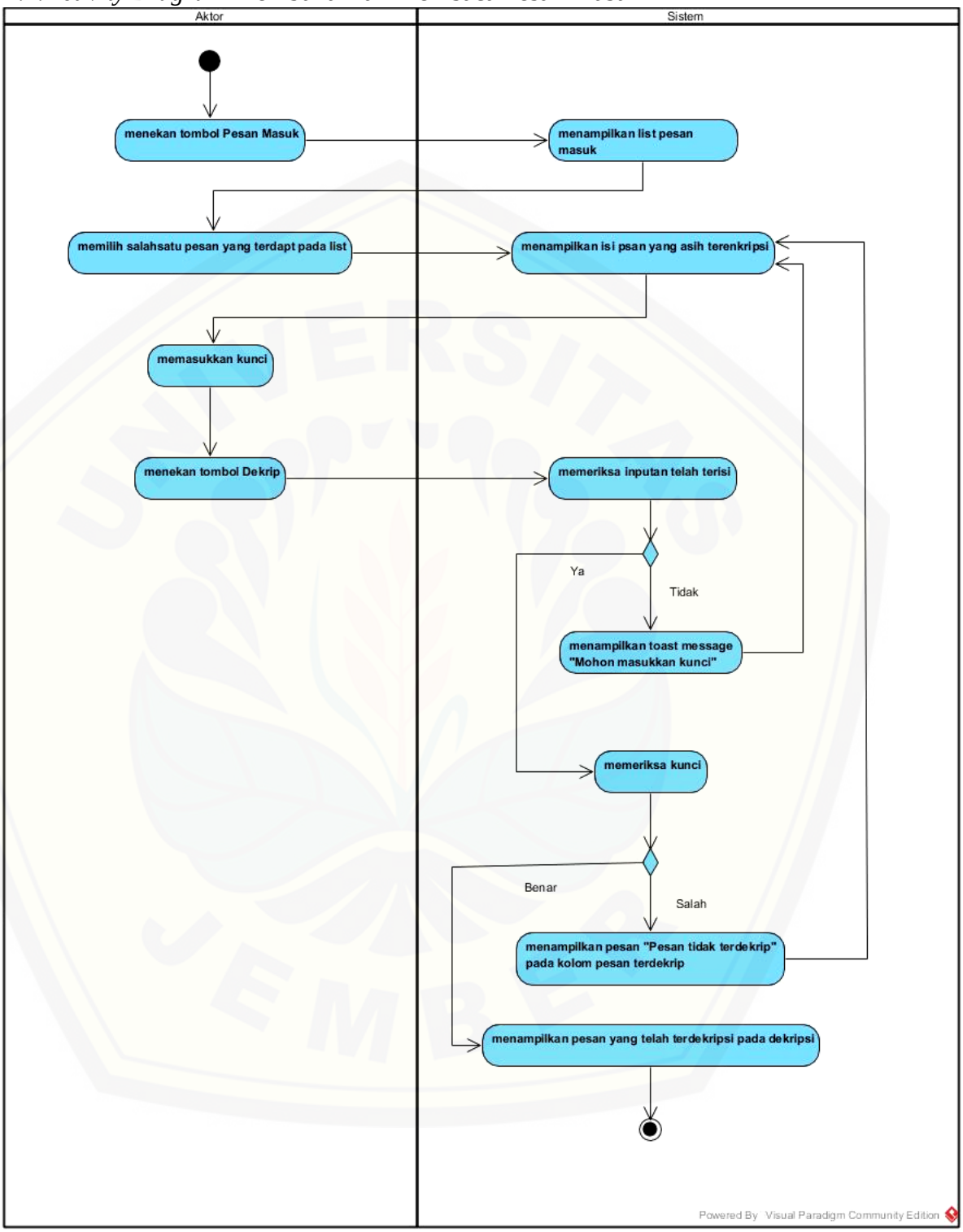
A. Lampiran *Activity Diagram*

A.1.*Activity Diagram* Membuat Pesan Dan Membuat Kunci



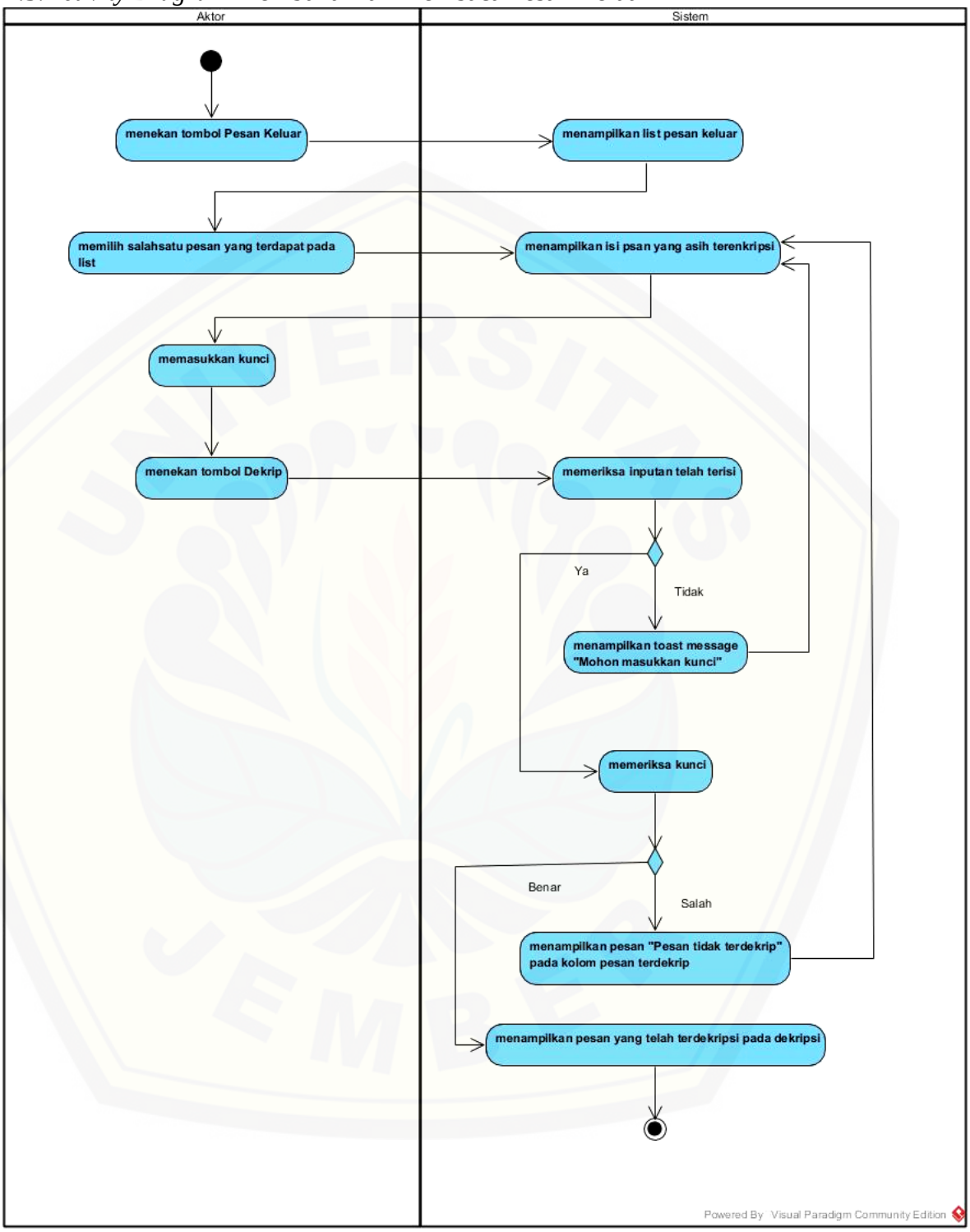
Gambar 4. 8 *Activity diagram* membuat pesan dan membuat kunci

A.2.*Activity Diagram* Membuka Dan Membaca Pesan Masuk



Gambar 4. 9 *Activity diagram* membuka dan membaca pesan masuk

A.3.*Activity Diagram* Membuka Dan Membaca Pesan Keluar



Gambar 4. 10 *Activity diagram* membuka dan membaca pesan keluar

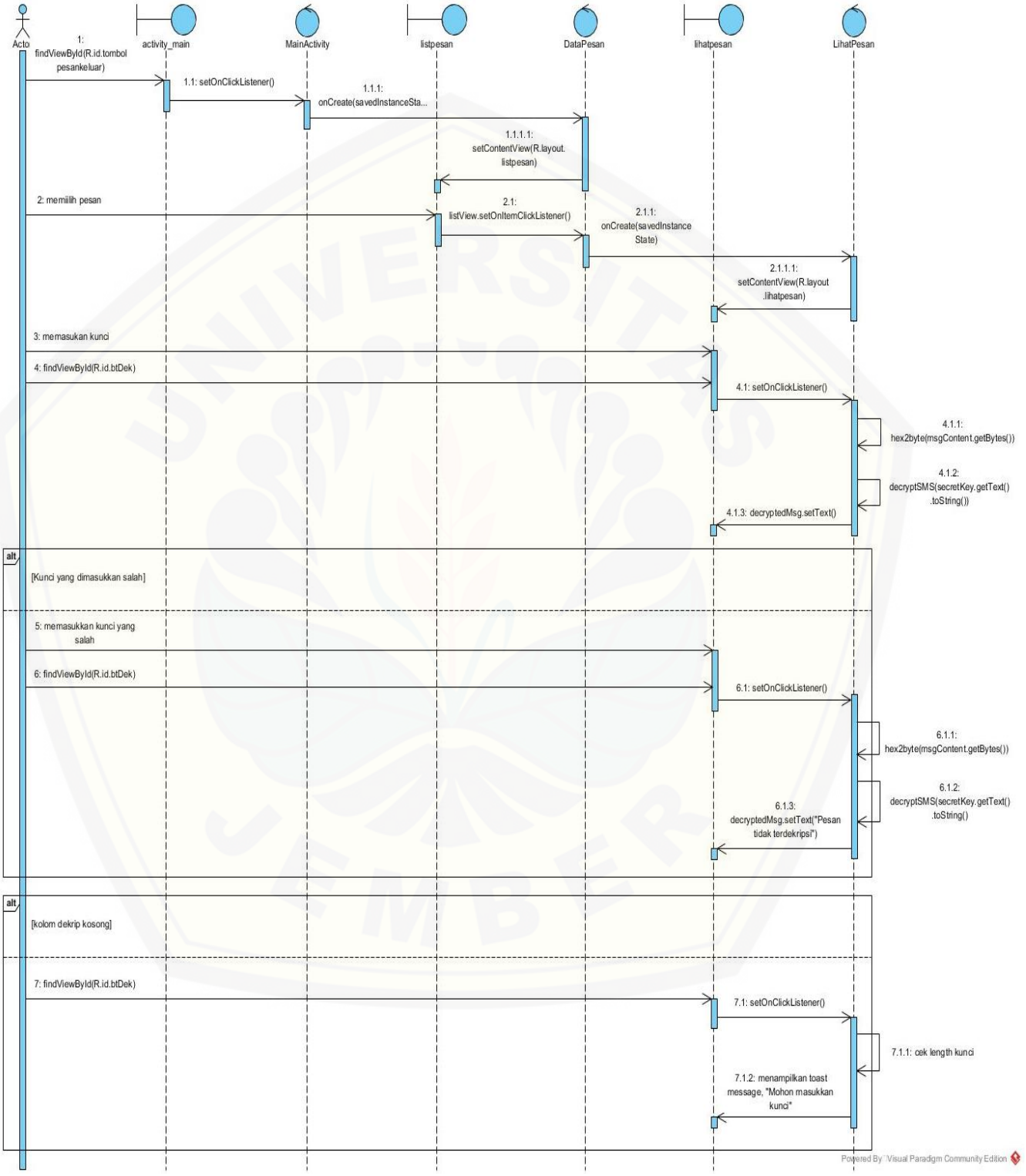
B. Tabel *S-Box*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Gambar 2. 17 Tabel *S-Box* (Barbosa,2015)

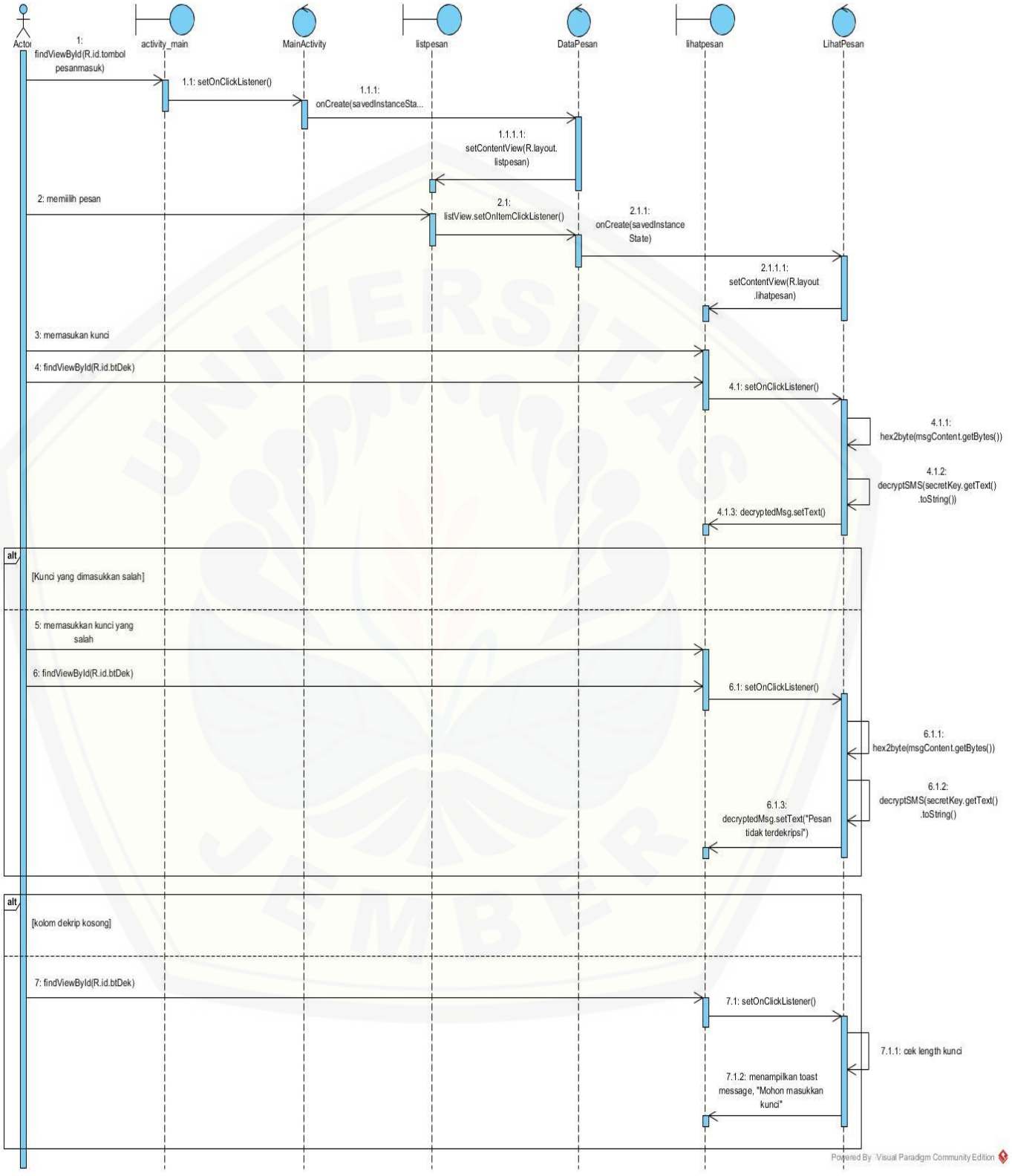
C. Lampiran *Sequence Diagram*

B.1. Sequence Diagram Membuka Dan Membaca Pesan Keluar



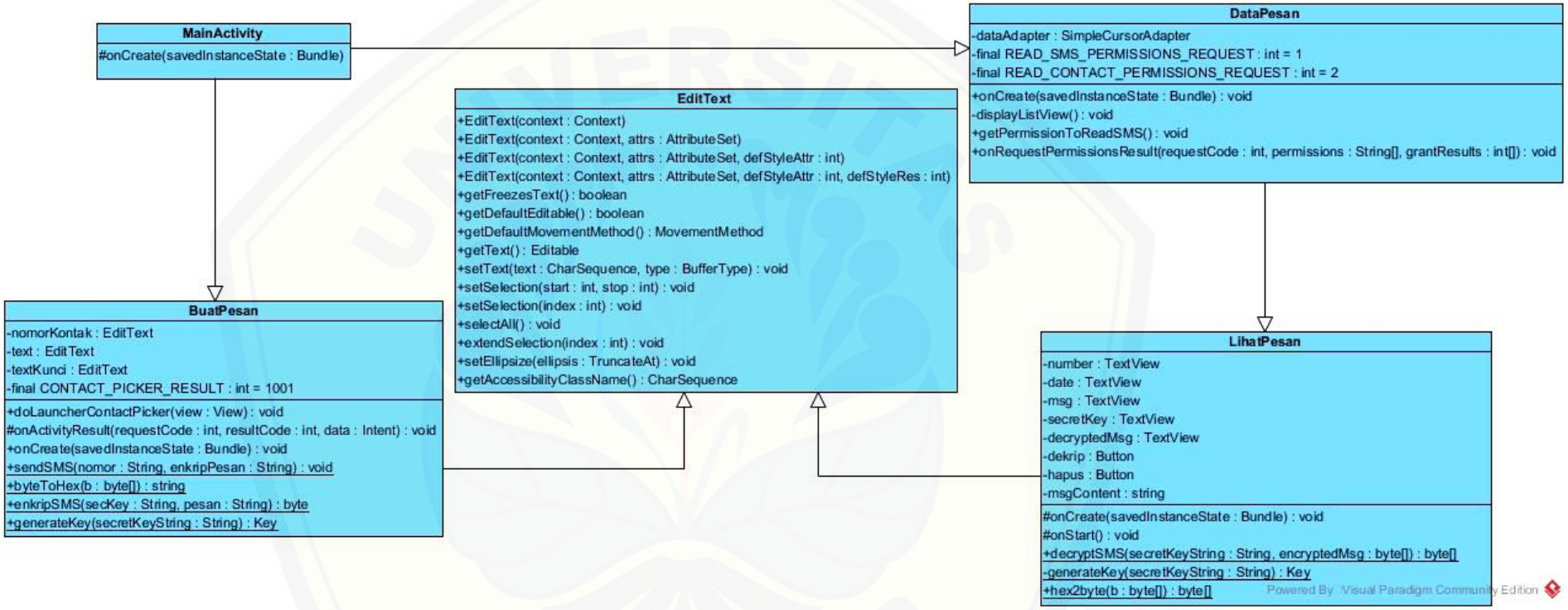
Gambar 4. 11 *Sequence diagram* membuka dan membaca pesan keluar

B.2. Sequence Diagram Membuka Dan Membaca Pesan Masuk



Gambar 4. 12 *Sequence diagram* membuka dan membaca pesan masuk

D. Lampiran *Class Diagram*



Gambar 4. 13 Class diagram SMS Security

E. Lampiran Kode Program

D.1. Kode BuatPesan.java

Kode BuatPesan.java

```
package com.ryu.smssederhana;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import java.security.Key;
import java.security.MessageDigest;
import java.util.ArrayList;
import java.util.Arrays;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;


public class BuatPesan extends Activity {
    EditText nomorKontak, text, textKunci;

    private static final int CONTACT_PICKER_RESULT = 1001;

    public void doLaunchContactPicker(View view) {
        Uri uri = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
        Intent contactPickerIntent = new Intent(Intent.ACTION_PICK,
uri);
        startActivityForResult(contactPickerIntent,
CONTACT_PICKER_RESULT);

    }

    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        String phone = "";
        Cursor contacts = null;
        try {
            if (resultCode == RESULT_OK) {
                switch (requestCode) {
                    case CONTACT_PICKER_RESULT:
                        Uri result = data.getData();
                        String id = result.getLastPathSegment();
```

```
                        contacts = getContentResolver().query(

ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                                null,

ContactsContract.CommonDataKinds.Phone._ID + "=?",
                                new String[] { id }, null);
                        int phoneIdx =
contacts.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DATA);
                        if (contacts.moveToFirst()) {
                            phone = contacts.getString(phoneIdx);
                            EditText phoneTxt = (EditText)
findViewById(R.id.nomorHp);
                            phoneTxt.setText(phone);
                        } else {
                            Toast.makeText(this, "error",
Toast.LENGTH_LONG).show();
                        }

                        break;

                }

            } else {
                Toast.makeText(BuatPesan.this, R.string.belumdipilih,
                        Toast.LENGTH_SHORT).show();
                }
        } catch (Exception e) {
            Toast.makeText(this, e.getMessage(),
Toast.LENGTH_LONG).show();
        } finally {
            if (contacts != null) {
                contacts.close();
            }
        }
    }

    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.buatpesan);

        Button send = (Button) findViewById(R.id.send);

        textKunci = (EditText) findViewById(R.id.kunci);
        text = (EditText) findViewById(R.id.smsBox);
        nomorKontak = (EditText) findViewById(R.id.nomorHp);

        send.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                String secKey = textKunci.getText().toString();
                String pesan = text.getText().toString();
                String nomor = nomorKontak.getText().toString();
```

```
                if (nomor.length() > 0 &&  secKey.length() > 0
                        && pesan.length() > 0
                        && secKey.length() == 16) {
                    byte[] enkripPesan = enkripSMS(secKey, pesan);
                    String pesanString = byteToHex(enkripPesan);

                    sendSMS(nomor, pesanString);

                    finish();

                } else {
                    Toast.makeText(BuatPesan.this,
                            "Key must be 16 characters",
                            Toast.LENGTH_SHORT).show();
                }

            }
        });

    }

    public static void sendSMS(String nomor, String enkripPesan) {
        try {
            //proses kirim
            SmsManager sms = SmsManager.getDefault();

            ArrayList<String> parts = sms.divideMessage(enkripPesan);
            sms.sendMultipartTextMessage(nomor, null, parts, null,
null);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static String byteToHex(byte[] b) {
        String hs = "";
        String stmp = "";
        for (int n = 0; n < b.length; n++) {
            stmp = Integer.toHexString(b[n] & 0xFF);
            if (stmp.length() == 1)
                hs += ("0" + stmp);
            else
                hs += stmp;
        }
        return hs.toUpperCase();
    }


    public static byte[] enkripSMS(String secKey, String pesan){
        try{
            byte[] returnArray;

            Key kunci = generateKey(secKey);
```

```
            Cipher c = Cipher.getInstance("AES");

            c.init(Cipher.ENCRYPT_MODE, kunci);

            returnArray = c.doFinal(pesan.getBytes());

            return returnArray;
        } catch (Exception e) {
            e.printStackTrace();
            byte[] returnArray = null;
            return returnArray;
        }
    }

    private static Key generateKey(String secretKeyString) throws
Exception {
        Key kunci = new
SecretKeySpec(secretKeyString.getBytes(),"AES");
        return kunci;
    }
}
```

D.2. Kode DataPesan.java

Kode DataPesan.java

```
package com.ryu.smssederhana;

import android.Manifest;
import android.app.Activity;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.support.annotation.NonNull;
import android.support.annotation.RequiresApi;
import android.support.v4.content.ContextCompat;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.TextView;
import android.widget.Toast;

import java.text.DateFormat;
import java.util.Date;

public class DataPesan extends Activity {
    private SimpleCursorAdapter dataAdapter;

    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.listpesan);

        getPermissionToReadSMS();

    }

    private void displayListView() {
        Intent i = getIntent();
        Uri uriSMS = Uri
                .parse("content://sms/" +
i.getStringExtra("tipepesan"));
        Cursor cursor = getContentResolver().query(uriSMS, null, null,
null,
                null);

        String[] columns = new String[] { "address", "body", "date" };

        int[] to = new int[] { R.id.pengirim, R.id.isipesan,
R.id.waktu };
```

```
        dataAdapter = new SimpleCursorAdapter(this,
R.layout.pesan_row, cursor,
                columns, to, 0);

        ListView listView = (ListView) findViewById(R.id.listView1);

        dataAdapter.setViewBinder(new SimpleCursorAdapter.ViewBinder()
{
            @Override
            public boolean setViewValue(View view, Cursor cursor,
                                        int columnIndex) {
                if (columnIndex == 2) {
                    TextView tv = (TextView) view;
                    String pengirimDB = cursor.getString(cursor
                            .getColumnIndex("address"));
                    Uri contactUri = Uri.withAppendedPath(

ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
                            Uri.encode(pengirimDB));
                    Cursor cur =
getContentResolver().query(contactUri, null,
                            null, null, null);
                    ContentResolver contect_resolver =
getContentResolver();

                    int size = cur.getCount();
                    if (size > 0 && cur != null) {
                        for (int i = 0; i < size; i++) {
                            cur.moveToPosition(i);

                            String id1 = cur.getString(cur

.getColumnIndexOrThrow(ContactsContract.Contacts._ID));

                            Cursor phoneCur = contect_resolver
                                    .query(contactUri,
                                            null,

ContactsContract.CommonDataKinds.Phone.CONTACT_ID
                                                    + " = ?",
                                            new String[] { id1 },
null);

                            if (phoneCur.moveToFirst()) {
                                String namaKontak =
phoneCur.getString(phoneCur

.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
                                phoneCur.close();
                                tv.setText(namaKontak);
                            } else {
                                tv.setText(pengirimDB);
                            }
```

```
                    }

                    cur.close();
                } else {
                    tv.setText(pengirimDB);
                }

                return true;
            }

            // konversi tanggal
            if (columnIndex == 4) {
                TextView tv = (TextView) view;
                String waktu = cursor.getString(cursor
                        .getColumnIndex("date"));
                long l = Long.parseLong(waktu);
                Date d = new Date(l);
                String date =
DateFormat.getDateInstance(DateFormat.LONG)
                        .format(d);
                String time =
DateFormat.getTimeInstance().format(d);
                String view_waktu = date + " " + time;

                tv.setText(view_waktu);

                return true;
            }

            return false;
        }
    });
    listView.setAdapter(dataAdapter);
    listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> listView, View
view, int position, long id) {
            Cursor cursor = (Cursor)
listView.getItemAtPosition(position);
            String view_pengirim = cursor.getString(cursor
                    .getColumnIndexOrThrow("address"));
            String view_isipesan = cursor.getString(cursor
                    .getColumnIndexOrThrow("body"));

            String waktu = cursor.getString(cursor
                    .getColumnIndexOrThrow("date"));
            long l = Long.parseLong(waktu);
            Date d = new Date(l);
            String date =
DateFormat.getDateInstance(DateFormat.LONG)
                    .format(d);
            String time = DateFormat.getTimeInstance().format(d);
```

```
                String view_waktu = date + " " + time;

                String view_idpesan = cursor.getString(cursor
                        .getColumnIndexOrThrow("_id"));
                String view_thread = cursor.getString(cursor
                        .getColumnIndexOrThrow("thread_id"));
                Intent click = new Intent(DataPesan.this,
LihatPesan.class);
                Uri contactUri = Uri.withAppendedPath(

ContactsContract.PhoneLookup.CONTENT_FILTER_URI,
                        Uri.encode(view_pengirim));
                Cursor cur = getContentResolver().query(contactUri,
null, null,
                        null, null);
                ContentResolver contect_resolver =
getContentResolver();

                int size = cur.getCount();
                if (size > 0 && cur != null) {
                    for (int i = 0; i < size; i++) {
                        cur.moveToPosition(i);

                        String id1 = cur.getString(cur

.getColumnIndexOrThrow(ContactsContract.Contacts._ID));

                        Cursor phoneCur =
contect_resolver.query(contactUri, null,
ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?", new
String[] { id1 }, null);

                        if (phoneCur.moveToFirst()) {
                            String namaKontak =
phoneCur.getString(phoneCur

.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME));
                            phoneCur.close();
                            click.putExtra("no", namaKontak);
                        } else {
                            click.putExtra("no", view_pengirim);
                        }

                    }

                    cur.close();
                } else {
                    click.putExtra("no", view_pengirim);
                }
                click.putExtra("msg", view_isipesan);
                click.putExtra("idpesan", view_idpesan);
                click.putExtra("idthread", view_thread);
                click.putExtra("date", view_waktu);
                Intent i = getIntent();
```

```
                click.putExtra("asal", i.getStringExtra("tipepesan"));
                startActivity(click);

            }
        });

    }

    private static final int READ_SMS_PERMISSIONS_REQUEST = 1;
    private static final int READ_CONTACT_PERMISSIONS_REQUEST = 2;

    @RequiresApi(api = Build.VERSION_CODES.M)
    public void getPermissionToReadSMS() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_SMS)
                != PackageManager.PERMISSION_GRANTED ||
ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_CONTACTS)
                != PackageManager.PERMISSION_GRANTED) {
            if (shouldShowRequestPermissionRationale(
                    Manifest.permission.READ_SMS)) {
                Toast.makeText(this, "Please allow permission!",
Toast.LENGTH_SHORT).show();
            }
            requestPermissions(new
String[]{Manifest.permission.READ_SMS,
Manifest.permission.READ_CONTACTS},
                    READ_SMS_PERMISSIONS_REQUEST);
        }else{
            displayListView();
        }


    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String permissions[], @NonNull int[] grantResults) {
        if (requestCode == READ_SMS_PERMISSIONS_REQUEST ) {
            boolean allgranted = false;
            for(int i=0;i<grantResults.length;i++){

if(grantResults[i]==PackageManager.PERMISSION_GRANTED){
                    allgranted = true;
                } else {
                    allgranted = false;
                    break;
                }
            }

            if(allgranted){
                displayListView();
            }
```

```
        }else {

        }
    }

}
```

D.3. Kode LihatPesan.java

Kode LihatPesan.java

```
package com.ryu.smssederhana;

import android.app.Activity;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.security.Key;
import java.security.MessageDigest;
import java.util.Arrays;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class LihatPesan extends Activity {
    TextView number, date, msg, secretKey, decryptedMsg;
    Button dekrip, hapus;
    String msgContent;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.lihatpesan);
        number = (TextView) findViewById(R.id.tvNumber);
        date = (TextView) findViewById(R.id.tvDate);
        msg = (TextView) findViewById(R.id.tvMsg);
        dekrip = (Button) findViewById(R.id.btDek);
        secretKey=(EditText) findViewById(R.id.KunciBox);
        decryptedMsg=(TextView) findViewById(R.id.tvDekrip);
    }

    @Override
    protected void onStart() {
        super.onStart();
        Intent i = getIntent();
        number.setText(i.getStringExtra("no"));
        date.setText(i.getStringExtra("date"));
        msg.setText(i.getStringExtra("msg"));
        msgContent=i.getStringExtra("msg");

        dekrip.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
```

```
                String secretKeyString =
secretKey.getText().toString();
                if (secretKeyString.length() > 0
                        && secretKeyString.length() > 0) {
                    try { byte[] msg =
hex2byte(msgContent.getBytes());
                        byte[] result =
decryptSMS(secretKey.getText().toString(), msg);
                        decryptedMsg.setText(new String(result));

                    } catch (Exception e) {
                        decryptedMsg.setText("Pesan tidak
terdekripsi");
                    }

                } else
                    Toast.makeText(getBaseContext(), "Mohon masukkan
kunci!", Toast.LENGTH_SHORT).show();

            }
        });

    }

    public static byte[] decryptSMS(String secretKeyString, byte[]
encryptedMsg)
            throws Exception {
        Key key = generateKey(secretKeyString);
        Cipher c = Cipher.getInstance("AES");
        c.init(Cipher.DECRYPT_MODE, key);
        byte[] decValue = c.doFinal(encryptedMsg);

        return decValue;
    }

    private static Key generateKey(String secretKeyString) throws
Exception { Key key2 = new
SecretKeySpec(secretKeyString.getBytes(),"AES");
        return key2;
    }

    public static byte[] hex2byte(byte[] b) {
        if ((b.length % 2) != 0)
            throw new IllegalArgumentException("hello");

        byte[] b2 = new byte[b.length / 2];

        for (int n = 0; n < b.length; n += 2) {
            String item = new String(b, n, 2);
            b2[n / 2] = (byte) Integer.parseInt(item, 16);
        }
        return b2;
    }
}
```

D.4. Kode *build.gradle*

Kode *build.gradle*

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 24
    defaultConfig {
        applicationId "com.ryu.smssederhana"
        minSdkVersion 15
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        debug {
            minifyEnabled true
            useProguard false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'),
                    'proguard-rules.pro'
        }
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'),
                    'proguard-rules.pro'
        }
        flavorDimensions "version"
        productFlavors {

            flavor1 {

            }
            flavor2 {
                proguardFile 'flavor2-rules.pro'
            }

        }
    }

}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

androidTestImplementation('com.android.support.test.espresso:espresso
-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-
annotations'
    })
    implementation 'com.android.support:appcompat-v7:24.2.1'
```

```
    //compile 'com.android.support:appcompat-v7:24.2.1'
    implementation 'com.android.support.constraint:constraint-
layout:1.0.2'
    //compile 'com.android.support.constraint:constraint-
layout:1.0.2'
    testImplementation 'junit:junit:4.12'
}
```