BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Dedi Darwis pada tahun 2016 melakukan penelitian yang berjudul "Implementasi Teknik Steganografi *Least Significant Bit* (LSB1) Dan Kompresi Untuk Pengamanan Data Pengiriman Surat Elektronik". Penelitian tersebut menghasilkan sebuah aplikasi untuk menyisipkan pesan rahasia ke dalam sebuah gambar menggunakan algoritma LSB yang selanjutnya gambar tersebut dikirim melalui surat elektronik. Terdapat perbedaan dengan penelitian yang akan dilaksanakan, yaitu pada penelitian yang akan dibuat akan diterapkan algoritma RC4 untuk menyandikan pesan sebelum disisipkan ke dalam gambar sehingga pesan rahasia tersebut lebih terjaga keamanannya.

Penelitian lain juga dilakukan oleh Inu Nugraha dan Nana Suarna pada tahun 2013 dengan judul "Konsep *Hidden Message* Pada Citra BMP 24-*Bit* Menggunakan Teknik Steganografi *Least Significant Bit* (LSB) Bahasa Pemrograman *Delphi*" menghasilkan sebuah aplikasi steganografi yang digunakan pada komputer. Perbedaan dengan penelitian yang dibuat adalah pada penelitian yang akan dibuat aplikasi yang akan dibangun akan digunakan pada *smartphone android*.

II.2. Uraian Teoritis

II.2.1. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus computer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, "Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu". (Juansyah, Andi ; 2015 : 2)

II.2.2. Pengertian Steganografi

Steganografi merupakan seni untuk menyembunyikan pesan di dalam media digital sedemikian rupa sehingga orang lain tidak menyadari ada sesuatu pesan di dalam media tersebut. Kata steganografi (*steganography*) berasal dari bahasa Yunani *steganos* yang berarti "tersembunyi/terselubung" dan *grephien* "menulis" sehingga kurang lebih artinya "menulis (tulisan) terselubung". (Dedi Darwis ; 2016 : 3)

II.2.3. Pengertian Gambar

Citra merupakan istilah lain untuk gambar sebagai salah satu komponen multimedia yang memegang peranan yang sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi.

Secara harfiah, citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagai dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, misalnya mata pada manusia, kamera, pemindai (*scanner*), dan sebagainya. Sehingga bayangan objek yang disebut citra tersebut terakam. (Permadi dan Murinto ; 2015 : 1029)

II.2.4. Kriptografi

Kriptografi (*Cryptography*) berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu *kripto* dan *graphia. Kripto* artinya menyembunyikan, sedangkan *graphia* artinya tulisan. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasian data, keabsahan data, integritas data, serta autentikasi data. Tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi.

Kriptografi dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan dikirim dari suatu tempat ke tempat lain, isi pesan tersebut mungkin dapat disadap oleh pihak lain yang tidak berhak untuk

mengetahui isi pesan tersebut. Untuk menjaga pesan, maka pesan tersebut dapat diubah menjadi sebuah kode yang tidak dapat dimengerti pihak lain.

Enkripsi adalah sebuah proses penyandian yang melakukan perubahan sebuah kode (pesan) dari yang bisa dimengerti (*plaintext*) menjadi sebuah kode yang tidak bisa dimengerti (*chipertext*). Sedangkan proses kebalikannya untuk mengubah *cipertext* menjadi *plaintext* disebut dekripsi. Proses enkripsi dan deskripsi memerlukan suatu mekanisme dan kunci tertentu. 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. 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. (Amin, M. Miftakul ; 2016 : 130-131)

II.2.5. Penjelasan Algoritma

Algoritma adalah sistem kerja komputer memiliki *brainware*, *hardware*, dan *software*. Tanpa salah satu dari ketiga sistim tersebut, komputer tidak akan berguna. Kita akan lebih fokus pada *software* komputer. *Software* terbangun atas susunan program) dan *syntax* (cara penulisan/pembuatan program). Untuk menyusun program atau *syntax*, diperlukannya langkah-langkah yang sistematis dan logis untuk dapat menyelesaikan masalah atau tujuan dalam proses

pembuatan suatu *software*. Maka, algoritma berperan penting dalam penyusunan program atau *syntax* tersebut.

Pengertian algoritma adalah susunan yang logis dan sistematis untuk memecahkan suatu masalah atau untuk mencapai tujuan tertentu. Dalam dunia komputer, algoritma sangat berperan penting dalam pembangunan suatu *software*. Dalam dunia sehari-hari, mungkin tanpa kita sadari algoritma telah masuk dalam kehidupan kita.

Algoritma berbeda dengan logaritma. Logaritma merupakan operasi matematika yang merupakan kebalikan dari eksponen atau pemangkatan. Contoh logaritma seperti bc = a ditulis sebagai blog a = c (b disebut basis). (Maulana, Gun Gun ; 2017 : 70)

II.2.5.1. Algoritma RC4

Pada tahun 1987 di Laboratorim Rsa, Ron Rivest menemukan suatu algoritma yang diberi nama RC4. RC itu sendiri merupakan singkatan dari Ron's Code. Karena algoritma RC4 dapat diimplementasikan secara efisien pada perangkat lunak maka menjadikan algoritma RC4 populer untuk aplikasi *internet* antara lain digunakan sebagai standar WEP (*Wired Equivalent Privacy*), WPA (*Wifi Protected Acces*) dan TLP (*Transport Layer Protocol*). RC4 juga diimplementasikan pada protokol SSL (*Source Socket Layer*) yaitu sebuah protokol untuk memproteksi trafik *internet*.

Terdapat dua tahapan untuk membangkitkan aliran kunci algoritma RC4 yaitu *Key Scheduling Algorithm* (KSA) dan *Pseudo-Random Generator Algorithm*

(PRGA). *Key Scheduling Algorithm* (KSA) merupakan tahapan pemberian nilai awal berdasarkan kunci enkripsi. *State* dari nilai awal tersebut berupa *array* dengan representasi permutasi 256 *byte* (dengan indeks 0 sampai dengan 255) dinamakan *array* S. Menggunakan rentang tersebut karena RC4 mengenkripsi pada mode *byte* (255=2^8 dan 8 *bit* = 1 *byte*). Artinya maksimal panjang kunci yang dapat tersimpan pada *array* U adalah 256 karakter. Permutasi terhadap nilai *array* S dilakukan dengan *pseudo-code* berikut :

```
j = 0
for i = 0 to 255
S[i] = i
for i = 0 to 255
j = ( j + S[i] + U[i] ) mod
256 swap ( S[i], S[j] ) (*pertukaran nilai S[i] dan S[j] *)
```

Tahap selanjutnya hasil dari *array* S yang telah melalui KSA akan diproses kembali pada PRGA (*Pseudo-Random Generator Algorithm*). Pada tahap PRGA terjadi modifikasi *state* dan *output* sebuah *byte* dari aliran kunci, dimana *array* S beroperasi dengan *array* U yang selanjutnya akan menghasilkan *keystream*. Nilai S[i] dan S[j] diambil dan dijumlahkan dengan modulo 256 untuk memembangkitkan aliran kunci. Hasil dari perhitungan tersebut akan menjadi indeks S [indeks] yang menjadi aliran kunci K yang kemudian digunakan untuk mengenkripsi plainteks ke-aliran kunci K yang kemudian digunakan untuk mengenkripsi plainteks ke-*idx*. Setiap putaran bagian *keystream* sebesar 1 *byte*

(dengan nilai antara 0 sampai dengan 255) dioutputkan oleh PRGA berdasarkan *state* S. Berikut adalah PRGA dalam bentuk *pseudo-code*:

```
i = o
j = 0
for idx = 0 to Panjang Plainteks –
1 do
i = ( i + 1 ) mod 256
j = ( j + S[i] ) mod 256
swap ( S[i], S[j] ) (*penukaran nilai S[i] dan S[j] *)
K = ( S[i] + S[j]) mod
256
Endfor
```

Setelah *keystream* terbentuk, kemudian *keystream* tersebut dimasukkan dalam operasi *XOR* dengan *plaintext*. (Galuh Adjeng Sekarsari, et al. ; 2015 : 251-252)

II.2.5.2. *Least Significant Bit* (LSB)

Metode *Least Significant Bit* (LSB) merupakan metode metode yang tidak terlalu kompleks, penyimpanan pesan pada *cover object* juga cukup besar. Dasar metode ini adalah bilangan berbasis biner yaitu angka 0 dan 1, karena pada data *digital* merupakan susunan angka 0 dan 1 maka proses penerapannya menjadi mudah. Lebih lanjut metode ini berhubungan erat dengan ukuran 1 *bit* dan ukuran 1 *byte* dimana 1 *byte* data terdiri dari 8 *bit* data dan *bit* pada posisi paling kanan

disebut dengan LSB. Steganografi dengan metode LSB diganti dengan *bit* yang disembunyikan. Karena *bit* yang diganti hanya *bit* yang paling akhir, maka *stego image* yang dihasilkan hampir sama persis dengan *cover image* nya. (Dedi Darwis ; 2016 : 3)

Ukuran data yang akan disembunyikan bergantung pada ukuran data penampung. Misalkan saja pada file citra 8-*bit* yang berukuran 256x256 *pixel* terdapat 65536 *pixel*, setiap *pixel* berukuran 1 *byte*. Setelah diubah menjadi citra 24-*bit*, ukuran data *bitmap* menjadi 65536 x 3 = 196608 *byte*.

Karena setiap *byte* hanya bisa menyembunyikan satu *bit* di LSB-nya, maka ukuran data yang akan disembunyikan di dalam citra maksimum 196608/8 = 24576 *byte*. Ukuran data ini harus dikurangi dengan panjang nama berkas, karena penyembunyian data rahasia tidak hanya menyembunyikan isi data tersebut, tetapi juga nama berkasnya.

Pada susunan *bit* di dalam sebuah *byte* (1 *byte* = 8 *bit*), ada *bit* yang paling berarti (*most significant bit* atau MSB) dan *bit* yang paling kurang berarti (*least significant bit* atau LSB). Sebagai contoh *byte* 1101001**0**, angka *bit* 1 (pertama, digaris-bawahi) adalah *bit* MSB, dan angka *bit* 0 (terakhir, cetak tebal) adalah *bit* LSB. *Bit* yang cocok untuk diganti adalah *bit* LSB, sebab perubahan tersebut hanya mengubah nilai *byte* satu lebih tinggi atau satu lebih rendah dari nilai sebelumnya. Misalkan *byte* tersebut menyatakan warna kuning, maka perubahan satu *bit* LSB tidak mengubah warna kuning tersebut secara berarti. Mata manusia tidak dapat membedakan perubahan kecil tersebut. Misalkan segmen *pixel-pixel* citra/gambar sebelum penambahan *bit-bit* adalah :

00110011 10100010 11100010

10101011 00100110 10010110

11001001 11111001

Pesan rahasia (yang telah dikonversi ke sistem biner) misalkan '11100101', maka setiap *bit* dari pesan tersebut menggantikan posisi LSB dari segmen *pixel-pixel* citra menjadi (di cetak tebal):

00110011 10100011 11100011

10101010 00100110 10010111

11001000 11111001

Pada saat ekstrasi data ambil tiap *bit* dari LSB dari setiap *byte pixel* pada gambar, selanjutnya *bit* tersebut di satukan dan *konversi* ke dalam karakter. (Sitorus, Michael ; 2015 : 55-56)

II.2.6. Bahasa Pemrograman *Java*

Java dikembangkan oleh *Sun Microsystems* pada Agustus 1991. *Java* disebut juga merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C dan C++. Pemrograman *Java* bersifat tidak bergantung pada *platform*, yang artinya, *java* dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Sebagaimana halnya C++, salah satu bahasa yang mengilhami *Java*, *Java* juga merupakan bahasa pemrograman berorientasi objek. Sebagai bahasa pemrograman berorientasi objek, *Java* menggunakan kelas untuk membentuk suatu objek. Karakteristik *Java* antara lain adalah berorientasi objek (*object-oriented*), terdistribusi (*distributed*), sederhana (*simple*), aman (*secure*),

interpreted, robust, multithreaded, dan dinamis. (Annisa Rahmawati, et al. ; 2015 : 336)

II.2.6.1. *Eclipse IDE*

Eclipse IDE adalah sebuah *IDE* (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform_independent*). Berikut ini adalah sifat dari *Eclipse* :

a. *Multi platform* :

   Target sistem operasi *Eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.*

b. *Multi language* :

   Eclipse dikembangkan dengan bahasa pemrograman *Java*. Akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++, Cobol, Python, Perl, PHP,* dan lain sebagainya.

c. *Multi role* :

   Selain sebagai *IDE* untuk pengembangan aplikasi, *Eclipse* bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan *web* dan lain sebagainya. (Murtiwiyati dan Lauren ; 2013 : 3)

Eclipse pada saat ini merupakan salah satu *IDE* favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari *Eclipse* yang membuatnya populer

adalah kemampuannya untuk dapat dikembangkan pengguna dengan komponen yang dinamakan *plug-in*. (Hendra Nugraha Lengkong ; 2015 : 21)

II.2.6.2. *ADT Plugin for Eclipse*

Android *Development Tools* (*ADT*) adalah *plug-in* untuk *Eclipse IDE* yang dirancang khusus untuk memberikan *integrated environment* yang kuat untuk membuat aplikasi *android*. *ADT* memberikan kemampuan kepada *Eclipse* untuk membuat projek baru *Android* secara tepat, membuat aplikasi *User Interface*, menambahkan komponen berdasarkan *Android Framework API*, melakukan *debugging* aplikasi yang dibuat dengan menggunakan *Android SDK Tools* dan bahkan melakukan distribusi aplikasi yang dibuat. Pembuatan aplikasi *android* dengan *Eclipse* beserta *ADT* sangat dianjurkan karena merupakan cara tercepat untuk memulai membuat projek *Android*. Dengan disediakannya *project setup*, serta *tools* yang sudah terintegrasi. (Hendra Nugraha Lengkong ; 2015 : 22)

II.2.6.3. Versi *Android*

Android adalah sistem operasi seluler yang didasarkan pada versi modifikasi *linux*. Hampir semua *smartphone* memiliki sistem operasi *android*. Dalam pengembangannya *andorid* telah mengalami cukup banyak pembaruan sejak awal dirilis yang akan ditunjukan pada tabel di bawah ini.

Tabel II.1. Versi-versi *Android*

(Sumber : Tareq Ilham Pramadana, et al. ; 2018 : 14)

Versi	Nama	Tanggal Rilis
1.5	*Cupcake*	30 April 2009
1.6	*Donut*	15 September 2009
2.0-2.1	*Éclair*	26 Oktober 2009
2.2	*Froyo*	20 Mei 2010
2.3-2.3.2	*Gingerbread*	6 Desember 2010
2.3.3-2.3.7	*Gingerbread*	9 Februari 2011
3.1	*Honeycomb*	10 Mei 2011
3.2	*Honeycomb*	15 Juli 2011
4.0.3-4.0.4	*Ice Cream Sandwich*	16 Desember 2011
4.1.x	*Jelly Bean*	9 Juli 2012
4.2.x	*Jelly Bean*	13 November 2012
4.3.x	*Jelly Bean*	24 Juli 2013
4.4.x	*Kitkat*	31 Oktober 2013
5.0	*Lollipop*	15 Oktober 2014
6.0	*Marshmallow*	5 Oktober 2015

II.2.7. Pengertian *UML*

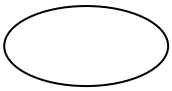
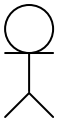
UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. *UML* merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. *UML* saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem (Windu dan Grace ; 2013 : 81). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut :

II.2.7.1. *Use Case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat

dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. (Windu dan Grace ; 2013)

Tabel II.2. *Use Case Diagram*

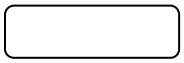
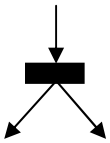
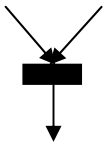
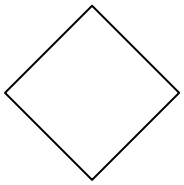
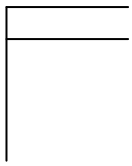
Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktir, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *Use Case*, tetapi tidak memiliki kontrol terhadap *use case*
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat

(Sumber : Ade Hendini ; 2016)

II.2.7.2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dan Grace ; 2013 : BIT. 10. 80-87).

Tabel II.3. ***Activity Diagram***

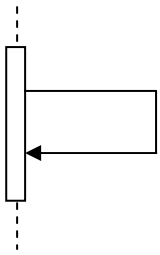
Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

(Sumber : Ade Hendini ; 2016)

II.2.7.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dan Grace ; 2013).

Tabel II.4. *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* dan *form* cetak
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
	Message, simbol mengirim pesan antar *class*
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini ; 2016)