BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Rambe, dkk (2018) mengenai Aplikasi Pengamanan Data Dan Disisipkan Pada Gambar Dengan Algoritma RSA Dan *Modified* LSB Berbasis *Android*, Rambe, dkk menyimpulkan bahwa kombinasi metode keamanan menggunakan metode kriptografi RSA dan steganografi *modified* LSB pada perangkat android yang dibuat dengan pemrograman *Eclipse* berhasil mencapai tujuan untuk mengamankan data pada perangkat *android*.

Berdasarkan penelitian yang dilakukan oleh Handoyo, dkk (2018) mengenai Teknik Penyembunyian dan Enkripsi Pesan pada Citra Digital dengan Kombinasi Metode LSB dan RSA, Handoyo, dkk menyimpulkan bahwa Penggunaan RSA 16 bit pada penyandian citra pesan dapat meningkatkan keamanan karena nilai *p* dan *q* dapat lebih bervariasi. Dengan variasi yang semakin banyak maka enkripsi RSA dapat lebih aman.

Berdasarkan penelitian yang dilakukan oleh Sekarwati dan Budiman (2017) mengenai Implementasi Algoritma *Rivest-Shamir-Adleman* (RSA) Dan Metode *Least Significant Bit* (LSB) Untuk Keamanan *File* Teks Dan Dokumen Menggunakan Visual C#, Sekarwati dan Budiman menyimpulkan bahwa proses enkrispi dan penyisipan (*encoding*) dapat berjalan dengan baik pada *file* teks maupun dokumen. Pada proses ekstraksi (*decoding*) dan dekripsi, aplikasi juga

dapat berjalan dengan baik. Dari skenario uji coba enkripsi dan dekripsi dapat dilihat bahwa jumlah karakter hasil enkripsi lebih banyak dari karakter yang dienkripsi.

Berdasarkan penelitian yang dilakukan oleh Farid, dkk (2016) mengenai Implementasi Metode Steganografi *Least Significant Bit* Dengan *Algoritma Hill Cipher* Pada Citra Bitmap, Farid, dkk menyimpulkan bahwa Penggabungan metode kriptografi *Hill Cipher* dan steganografi *Least Significant Bit* (LSB) dapat meningkatkan keamanan pada pesan dalam bertukar informasi.

Berdasarkan penelitian yang dilakukan oleh Harjo, dkk (2016) mengenai Aplikasi Steganografi Menggunakan LSB *(Least Significant Bit)* dan Enkripsi *Triple Des* Menggunakan Bahasa Pemrograman C#, Harjo, dkk menyimpulkan bahwa Hasil dari penerapan untuk penyisipan pesan rahasia pada gambar berjalan dengan baik. Pesan atau dokumen yang disisipkan pada file gambar dapat diperoleh kembali secara utuh atau dengan kata lain pesan yang disisipkan sebelum proses enkripsi dan setelah proses dekripsi mempunyai hasil yang sama tanpa ada perubahan pesan atau gangguan.

II.2. Landasan Teori

Berikut ini adalah beberapa landasan teori yang dikutip dari beberapa referensi berupa jurnal-jurnal yang berkaitan :

II.2.1. Kriptografi

Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan. Kriptografi mengubah informasi asli (*plaintext*) melalui proses enkripsi menjadi

informasi acak (*ciphertext*) menggunakan algoritma dan kunci tertentu, lalu setelah diterima oleh penerima informasi, *ciphertext* akan diubah kembali menjadi plaintext melalui proses dekripsi menggunakan algoritma dan kunci yang sama dengan proses enkripsi. (Rambe, dkk, 2018 : 724).

Kriptografi adalah suatu ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dikirimkan, dari pengirim kepenerima tanpa mengalami gangguan dari pihak ketiga. Menurut Bruce Scheiner dalam bukunya "Applied Cryptography", kriptografi adalah ilmu pengetahuan dan seni menjaga pesan tetap aman. Dalam kriptografi diperlukan parameter yang digunakan untuk proses konversi data yaitu suatu set kunci. Enkripsi dan dekripsi data dikontrol oleh sebuah kunci atau beberapa kunci. Proses enkripsi dan dekripsi dalam kriptografi. (Harjo, dkk, 2016 : 13).

Gambar II.1. Cara Penyembunyian Pesan Steganografi
(Sumber : Harjo, dkk, 2016 : 14).

II.2.1.1. Prinsip-Prinsip Dasar Kriptografi

Prinsip-prinsip yang mendasari kriptografi yakni :

1. *Confidelity* (kerahasiaan) yaitu layanan agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima / pihak-pihak memiliki ijin). Umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematis yang mampu mengubah data hingga menjadi sulit untuk dibaca dan dipahami.

2. *Data integrity* (keutuhan data) yaitu layanan yang mampu mengenali/mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
3. *Authentication* (keotentikan) yaitu layanan yang berhubungan dengan identifikasi. Baik otentikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentikasi keaslian data/informasi.
4. *Availability* (Ketersediaan), yaitu dimana *user* yang mempunyai hak akses atau *authorized users* diberi akses tempat waktu dan tidak terkendala apapun.
5. *Non-repudiation* (anti-penyangkalan) yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal dari dirinya). (Azis, 2018 : 73).

II.2.1.2. Istilah-Istilah Pada Kriptografi

Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi :

1. *Plaintext* (M) adalah pesan yang hendak dikirimkan (berisi data asli) yang dapat dimengerti.
2. *Ciphertext* (C) adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
3. Enkripsi (E) adalah proses pengubahan *plaintext* menjadi *ciphertext*.
4. Dekripsi (D) adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.
5. Kunci (K) adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi. (Azis, 2018 : 73).

II.2.1.3. Tujuan Kriptografi

Berikut ini adalah empat tujuan kriptografi yang termasuk ke dalam aspek keamanan informasi, yaitu :

1. Kerahasian Data (*Confidentiality*) :

Menjaga data agar tetap terahasia dari pihak-pihak yang tidak berwenang yang mungkin mencoba membaca data tersebut.

2. Integritas Data (*Integrity*) :

Memastikan data yang dikirim masih tetap sama dengan data yang diterima tanpa ada perubahan atau modifikasi terhadap data tersebut.

3. Autentikasi (*Authentatio*n) :

Memastikan bahwa pengirim dan penerima benar-benar terjamin keasliannya. Dua pihak yang berkomunikasi harus saling mengetahui satu dengan lainnya.

4. Non-Repudiasi (*Non-Repudiation*) :

Pengirim tidak bisa menyangkal kalau dia telah mengirim data, karena pengirim akan mendapatkan bukti kalau dia telah mengirim data kepada si penerima. (Harahap, 2016 : 62).

II.2.1.4. Pembagian Kriptografi

Berikut ini adalah pembagian kriptografi menurut Sadikin (2017 : 9) :

1. Fungsi *Hash*

   Fungsi *hash* adalah fungsi yang melakukan pemetaan npesan dengan panjang sembarang ke sebuah teks khusus yang disebut *message digest* dengan panjang tetap. Fungsi *hash* umumnya dipakai sebagai nilai uji (*check value*) pada mekanisme keutuhan data.

2. Penyandian dengan kunci simetrik (*Symmetric Key Encipherment*)

   Penyandian dengan kunci simetrik adalah penyandian yang kunci enkripsi dan kunci simetrik adalah penyandian yang kunci enkripsi dan kunci dekripsi bernilai sama. Kunci pada penyandian simetrik diasumsikan bersifat rahasia hanya pihak yang melakukan enkripsi dan dekripsi yang mengetahui nilainya. Oleh karena itu penyandian dengan kunci simetrik disebut juga penyandian dengan kunci rahasia *secret key encipherment*.

3. Penyandian dengan kunci asimetrik (*Asymmetric Key Encipherment*)

   Penyandian dengan kunci asimetrik atau sering juga disebut dengan penyandian kunci publik (*public key*) adalah penyandian dengan kunci enkripsi dan dekripsi berbeda nilai. Kunci enkripsi yang juga disebut dengan kunci publik (*public key*) bersifat terbuka. Sedangkan, kunci dekripsi yang juga disebut kunci privat (*private key*) bersifat tertutup/rahasia. (Sadikin, 2017 : 10).

II.2.1.4.1. Algoritma RSA

Algoritma RSA melakukan pemfaktoran bilangan yang sangat besar, oleh karena alasan tersebut RSA dianggap aman. Untuk membangkitkan dua kunci, dipilih dua bilangan prima acak yang besar. Algoritma RSA dibuat oleh 3 orang peneliti dari *Massachussets Institute of Technology* (MIT) pada tahun 1976, yaitu: Ron (R)ivest, Adi (S)hamir, dan Leonard (A)dleman. Berikut adalah algoritma RSA :

1. Pilih 2 buah bilangan prima.
2. Hitung nilai n dengan mengalikan p dan q.
3. Hitung nilai phi dengan cara phi = (p – 1) * (q – 1).
4. Pilih sebuah bilangan sebagai kunci enkripsi (e) dimana bilangan tersebut lebih besar dari 1 dan lebih kecil dari phi.
5. Hitung nilai untuk kunci dekripsi dengan rumus (d * e) % phi = 1.
6. Kemudian pilih sebuah pesan (M).

Enkripsi dari M adalah C = Me % n.

Dekripsi dari C adalah M = Cd % n.

Dengan, p dan q bilangan prima (rahasia), n = p.q (tidak rahasia), phi(n) = (p - 1)(q - 1) (rahasia), e (kunci enkripsi) (tidak rahasia), d (kunci dekripsi) (rahasia), M (plainteks) (rahasia), dan C (cipherteks) (tidak rahasia). (Sekarwati dan Budiman, 2017 : 56).

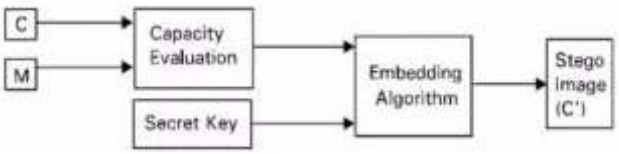
II.2.2. Steganografi

Steganografi adalah teknik penyembunyian data dalam sebuah medium yang dapat berupa jenis data apapun seperti *file* citra gambar, *audio*, *video*, maupun jenis data yang lainnya. (Farid, dkk, 2016 : 110).

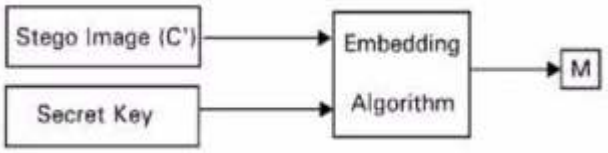
Steganografi berasal dari bahasa yunani "*steganos*" yang artinya "tersembunyi" atau "terselubung" dan "*graphein*" yang artinya "menulis". *Steganografi* dapat diartikan "tulisan tersembunyi" (*cover writing*). *Steganografi* adalah ilmu dan seni menyembunyikan pesan rahasia di dalam pesan lain sehingga keberadaan pesan rahasia tersebut tidak dapat diketahui. *Steganografi* membutuhkan dua properti: wadah penampung dan data rahasia yang akan

disembunyikan. *Steganografi* digital menggunakan media digital sebagai wadah penampung, misalnya: *voice, video, image* dan teks. Data rahasia yang disembunyikan jugadapatberupa *voice, video, image* dan teks. (Harjo, dkk, 2016 : 13).

Proses steganografi bisa dilihat pada gambar berikut :



Gambar II.2. Cara Penyembunyian Pesan Steganografi (Sumber : Harjo, dkk, 2016 : 13).



Gambar II.3. Cara Pengembalian Pesan Steganografi (Sumber : Harjo, dkk, 2016 : 14).

II.2.3. Metode *Least Signofocant Bit* (LSB)

Least Significant Bit (LSB) merupakan metode stegnografi yang paling sederhana dan mudah untuk diimplementasikan ke sebuah aplikasi. Metode ini menggunakan citra digital sebagai *convertext*. 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) (Harjo, dkk, 2016 : 14).

Langkah-langkah menyisipkan data dengan metode LSB :

1. Mengubah setiap *pixel* gambar asli menjadi *raster data*.
2. Mengubah pesan rahasia (bertipe karakter) menjadi bit-bit
3. Mengganti setiap bit rendah dan bit pesan.

   Jika bit rendah = bit pesan, maka *raster data* tidak berubah. Jika bit rendah lebih kecil dari (<) bit pesan, maka *raster data* ditambah 1. Dan jika bit rendah lebih besar dari (>) bit pesan, maka *raster data* dikurang 1.
4. Menulis *pixel* yang baru sesuai dengan *raster data*. (Sekarwati dan Budiman, 2017 : 56).

II.2.4. *Visual Basic* 2010

Visual Basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu *Visual Basic* 2008. Beberapa pengembangan yang terdapat didalamnya antara lain dukungan terhadap *library* terbaru dari *Microsoft*, yaitu *.Net Frame Work 4.0*, dukungan terhadap aplikasi berbasis *Cloud Computing*, serta perluasan dukungan terhadap *database-database*, baik *standalone* maupun *database server.* (Sari, dkk, 2015 : 2).

II.2.5. *Unified Modeling Language* (UML)

Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (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

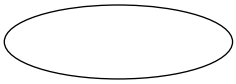
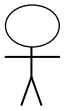
sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Urva dan Siregar, 2015 : 93).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*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 di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1 dibawah ini :

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, dan dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, 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 *control* 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 mengidikasikan aliran data.

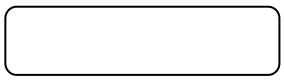
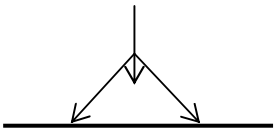
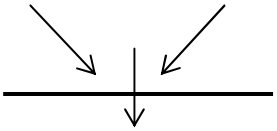
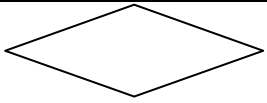
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Urva dan Siregar, 2015 : 94)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2 dibawah ini :

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*

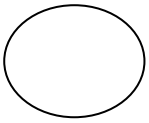
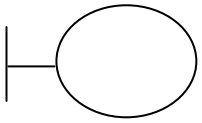
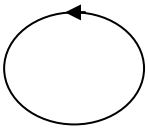
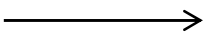
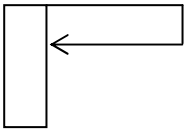
New Swimline	*Swimlane*, untuk menunjukkan siapa melakukan apa.

(Sumber : Urva dan Siregar, 2015 : 94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3 dibawah ini :

Tabel II.3. Simbol *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 *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry 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, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.

	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Urva dan Siregar, 2015 : 95)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility,* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4 dibawah ini :

Tabel II.4. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Urva dan Siregar, 2015 : 95)