BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Ahmad Mustofa, 2015 melakukan penelitian "Perancangan *E-Commerce* Penjualan Komputer Dan Alat Elektronik Berbasis Web Pada Toko Damar Komputer Pringsewu". Penelitian tersebut menghasilkan sebuah aplikasi berbasis *website* untuk melakukan pemesanan komputer dan alat elektronik. Yang menjadi perbedaan dengan penelitian yang akan dilaksanakan adalah pada penelitian yang akan dilaksanakan akan dibangun aplikasi berbasis *android* untuk pemesanan *hardware* komputer.

Retnani Latifah, et al. pada tahun 2017 melakukan penelitian yang berjudul "Modifikasi Algoritma *Caesar Chiper* Dan *Rail Fence* Untuk Peningkatan Keamanan Teks Alfanumerik Dan Karakter Khusus". Penelitian tersebut menghasilkan sebuah aplikasi untuk melakukan enkripsi dan dekripsi pesan teks menggunakan gabungan algoritma *Caesar* dan *Rail Fence* yang digunakan pada perangkat *desktop*. Terdapat perbedaan dengan penelitian yang akan dilaksanakan, yaitu pada penelitian yang akan dibuat algoritma *Rail Fence* akan digunakan untuk mengamankan data *login* pelanggan pada aplikasi pemesanan *hardware* komputer.

Desi Ratna (2018), "Implementasi Algoritma *Rail Fence Chiper* Dalam Keamanan Data Gambar 2 Dimensi" menghasilkan sebuah aplikasi yang dapat mengamankan sebuah *file* gambar. Berdasarkan penelitian tersebut pada penelitian

ini penulis akan memanfaatkan algoritma *Rail Fence* untuk mengamankan data *login* dari sebuah aplikasi pemesanan *hardware* komputer.

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 tehnik 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". (Andi Juansyah ; 2015 : 2)

II.2.2. Penjelasan Algoritma

Algoritma adalah sistim 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.2.3. 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.2.3.1. Jenis-Jenis Kriptografi

Kriptografi berdasarkan jenis kunci yang digunakan dapat digolongkan menjadi 3 yaitu kriptografi kunci simetris, kriptografi kunci asimetris dan kriptografi kunci hybrid(gabungan dari simetris dan asimetris). Pertama, kriptografi kunci simetris yaitu kriptografi yang dalam operasi enkripsi dan dekripsnya menggunakan kunci yang sama, dikenal sebagai private key. Contoh kriptografi kunci simetris yaitu *DES* (Data *Encyption* Standard), *3DES*, *IDEA*,

Blowfish, Twofish, Shift Cipher, Hill Cipher, Vernam cipher dan *AES (Advanced Encryption* Standard). Kedua, kriptografi kunci asimetris yaitu kriptografi yang dalam operasi enkripsi dan dekripsinya menggunakan 2 buah kunci berbeda yang disebut dengan kunci privat dan kunci publik. Contoh dari kriptografi kunci asimetris yaitu *RSA (Riverst Shamir Adleman), DSA (Digial Signature Algorithm), ECC (Elliptic Curve Cryptography), DH (Deffie Hellman)*dan *El Gamal*. Ketiga, kriptografi kunci gabungan simetris dan asimetris yaitu kriptografi yang menggunakan model persetujuan dari kedua belah pihak baik pengirim maupun penerima, dimana *session key* digunakan untuk mengenkripsi percakapan maupun mengenkripsi pertukaran data yang terjadi. Dalam hal ini setiap *session key* hanya dapat digunakan satu kali saja sehingga untuk sesi selanjutnya harus dibuat *session key* yang baru.

Berdasarkan kemunculannya, kriptografi digolongkan menjadi 2 yaitu kriptografi klasik dan kriptografi modern. Kriptografi klasik muncul pertama kali dan digunakan pada saat Perang Dunia II. Kriptografi klasik berbasis pada karakter dengan model operasi permutasi dan transposisi; dan biasanya merupakan kriptografi kunci simteris misalnya *Caesar Cipher*. Sedangkan kriptografi kunci modern yaitu kriptografi yang dibuat sedemikian rupa sehingga kompleks dan lebih sulit untuk dipecahkan disbanding dengan kriptografi kunci klasik umumnya. Jenis operasi pada kriptografi modern menggunakan mode *bit*, dimana semua data (plainteks, cipherteks, dan kunci) dinyatakan dalam string *bit biner* yaitu 0 dan 1. Rangkaian *bit* tersebut kemudian dipecah dalam blok-blok *bit* yang ditulis dalam berbagai cara bergantung pada panjang blok, misalnya

menggunakan padding *bits* atau dengan mengubah ke bentuk *heksadesimal.* (Christy Atika Sari, et. al. ; 2016 : 181-182)

II.2.4. Algoritma Rail Fence

Metode enkripsi *Rail Fence* adalah salah satu bentuk *cipher* transposisi yang sederhana yang diinspirasi dari model *Polybius square*. *Polybius square* adalah menyusun huruf sebagai matriks 5x5 dan mengkodekan huruf A sebagai 1-1, huruf B sebagai 1-2 dan seterusnya. Setiap karakter pada *Polybius square* diganti dengan indeks *cell* matriks tanpa menggunakan kunci khusus dan hanya merubah posisi sehingga teks tidak terbaca. Berbeda dengan *Polybius square*, metode *Rail Fence* menyusun teks secara *zig-zag* yang model matriksnya diketahui oleh pengirim dan penerima pesan. Berdasarkan jenis-jenis kriptografi algoritma *Rail Fence* termasuk ke dalam kriptografi klasik karena kriptografi klasik berbasis pada karakter dengan model operasi permutasi dan transposisi.

Teknik *Rail Fence* adalah menuliskan *plaintext* dalam urutan diagonal dan membacanya sebagai urutan baris sehingga terbentuk *ciphertext*. Contohnya jika ingin mengenkripsi HALO APA KABAR maka spasi dihilangkan kemudian disusun diagonal membentuk pola *zig-zag*. Metode *Rail Fence* mudah untuk dibobol oleh kriptanalis dengan mencoba beberapa nilai kedalaman untuk menentukan banyaknya baris yang digunakan. Terdapat pola tertentu berdasarkan jumlah baris yang digunakan. Misal jika ada dua baris maka huruf ke 1,3,5,... akan berada di baris pertama dan huruf ke 2,4,6,... akan ada di baris kedua. Meskipun metode *Rail Fence* adalah metode yang lemah, namun metode ini dapat

digabungkan dengan metode lain untuk meningkatkan keamanan *cipher text* sehingga tidak mudah dipecahkan. (Retnani Latifah, et al. ; 2017 : 2-3)

II.2.5. 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 dapt 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. Aplikasi *Mobile*

Aplikasi *mobile* dapat diartikan sebagai sebuah produk dari sistem komputasi *mobile*, yaitu sistem komputasi yang dapat dengan mudah dipindahkan secara fisik dan yang komputasi kemampuan dapat digunakan saat mereka sedang dipindahkan. Contohnya adalah *personal digital assistant* (PDA), *smartphone* dan ponsel. (Ramadhan dan Utomo ; 2014 : 47-55)

Berdasarkan jenisnya aplikasi mobile terbagi menjadi beberapa kelompok yaitu :

a. *Short Message Service* (SMS)

   Merupakan aplikasi mobile paling sederhana, dirancang untuk berkirim pesan dan berguna ketika terintegrasi dengan jenis aplikasi mobile lainnya.

b. *Mobile Websites* (Situs *Web Mobile*)

   Merupakan situs *web* yang dirancang khusus untuk perangkat *mobile*. Situs *web mobile* sering memiliki desain yang sederhana dan biasanya bersifat memberikan informasi.

c. *Mobile Web Application* (Aplikasi *Web Mobile*)

   Aplikasi *web mobile* merupakan aplikasi *mobile* yang tidak perlu diinstal atau dikompilasi pada perangkat target. Menggunakan XHTML, CSS, dan *JavaScript*, aplikasi ini mampu memberikan pengguna pengalaman layaknya aplikasi native/asli.

d. *Native Application* (Aplikasi Asli)

   Merupakan aplikasi *mobile* yang harus diinstal pada perangkat target. Aplikasi ini dapat disebut aplikasi *platform*, karena aplikasi ini harus dikembangkan dan disusun untuk setiap *platform mobile* secara khusus. (Ramadhan dan Utomo ; 2014 : 47-55)

II.2.7. Android

Android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat *mobile,* semua perangkat

lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, *Android* merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh *Google*. Sedangkan *Android SDK (Software Development Kit)* menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform Android* dengan menggunakan bahasa pemrograman Java. (Ahmad : 2015 : 190-200)

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi *Android*, sebuah file yang ditandai dengan akhiran *.apk.* file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. File ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu file *.apk,* dan kemudian kita sebut sebagai sebuah aplikasi. (Ahmad : 2015 : 190-200)

II.2.8. Android *Studio*

Android *studio* adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi *Android* dan bersifat *open source* atau gratis. Peluncuran *Android Studio* ini diumumkan oleh Google pada 16 mei 2013 pada *event* Google I/O *Conference* untuk tahun 2013. Sejak saat itu, *Android Studio* mengantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi *Android*.

Android studio sendiri dikembangkan berdasarkan *IntelliJ IDEA* yang mirip dengan *Eclipse* disertai dengan *ADT plugin (Android Development Tools). Android studio* memiliki fitur :

a. Projek berbasis pada Gradle *Build*
b. *Refactory* dan pembenahan *bug* yang cepat
c. *Tools* baru yang bernama "*Lint*" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
d. Mendukung *Proguard And App-signing* untuk keamanan.
e. Memiliki GUI aplikasi android lebih mudah
f. Didukung oleh Google *Cloud Platfrom* untuk setiap aplikasi yang dikembangkan. (Andi Juansyah ; 2015)

II.2.9. Android SDK (*Software Development Kit*)

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. Android SDK terdiri dari *debugger, libraries, handset emulator,* dokumentasi, contoh kode program dan tutorial. Saat ini *Android* sudah mendukung arsitektur x86 pada *Linux* (distribusi *Linux* apapun untuk *desktop* modern), *Mac OS* X 10.4.8 atau lebih, *Windows XP* atau *Vista*. Persyaratan mencakup JDK, *Apache Ant* dan *Python* 2.2 atau lebih. IDE yang didukung secara resmi adalah *Eclipse* 3.2 atau lebih dengan menggunakan plugin *Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML serta menggunakan peralatan *command line* untuk menciptakan, membangun, melakukan *debug* aplikasi Android dan pengendalian perangkat

Android (misalnya *reboot*, menginstal paket perangkat lunak). (Sulihati dan Andriyani ; 2016 : 20)

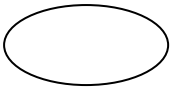
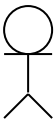
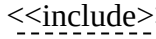
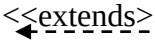
II.2.10. Pengertian UML

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 (Ade Hendini ; 2016 : 108). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.2.10.1. *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. (Ade Hendini ; 2016 : 108)

Tabel II.1. Use Case Diagram

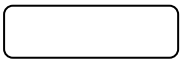
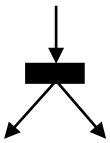
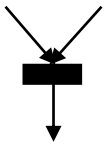
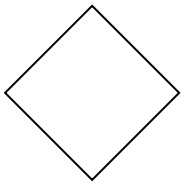
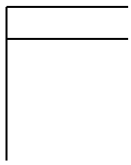
Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target 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.2.10.2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.(Ade Hendini ; 2016 : 109) Simbol-simbol yang digunakan dalam *activity Diagram* yaitu :

Tabel II.2. Activity Diagram

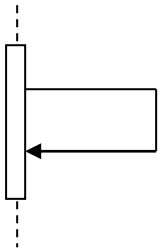
Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua 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.2.10.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Ade Hendini ; 2016 : 110). Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu :

Tabel II.3. *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)

II.2.10.4. *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan obyek yang dikoneksikan. *Class diagram* secara khas meliputi : Kelas (*Class*), *Relasi*, *Associations*, *Generalization* dan *Aggregation*, Atribut *(Attributes)*, Operasi *(Operations/Method)*, dan*Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar Kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti (Ade Hendini ; 2016 : 110).

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 maksimal 4

(Sumber : Ade Hendini ; 2016)