BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Penelitian Sebelumnya dimaksudkan bahwa masalah yang hendak di teliti belum pernah di pecahkan oleh peneliti terlebih dahulu. Jika permasalahannya mirip, maka harus ditegaskan perbedaan penelitiannya dengan penelitian terlebih dahulu. Berikut adalah beberapa jurnal penelitian terdahulu terkait judul penelitian yang sedang dilaksanakan.

Wina Ayu Lestari, Rohmat Tulloh, S.T., M.T., Atik Novianti, S.ST.,M.T. (2019) melakukan penelitian "Media Pembelajaran Interaktif Enkripsi *Caesar Cipher*, *Vigenere Cipher*, dan Algoritma RSA" Pada penelitian ini terdapat beberapa tampilan pada aplikasi ini seperti tampilan menu utama, tampilan detail materi, tampilan latihan soal , Tampilan referensi , Dengan begitu hasil yang didapat pada aplikasi ini memudahkan penggunanya dalam mengetahui dan memahami materi yang disampaikan, namun hanya bisa digunakan pada *platform* komputer.

Batara Silaban & Tonni Limbong (2017) melakukan penelitian "Aplikasi Pembelajaran Pengenalan Kriptografi Algoritma *Affine Cipher* dan *Vigenere Cipher* Menggunakan Metode *Computer Assisted Instruction*" Dalam penelitian ini, peneliti membuat sebuah aplikasi berbasis komputer untuk mempermudah

pengguna dalam memahami materi kriptografi. Aplikasi ini bisa digunakan oleh siapa saja namun hanya bisa digunakan pada *platform* komputer.

Yusfrizal (2019) melakukan penelitian "Rancang Bangun Aplikasi Kriptografi Pada Teks Menggunakan Metode Reverse Chiper dan RSA Berbasis *Android*" Dalam penelitian ini, peneliti membuat sebuah aplikasi berbasis *Android* untuk mempermudah pengguna dalam memahami simulasi penggunaan kriptografi. Aplikasi ini bisa digunakan oleh siapa saja namun hanya berfungsi sebagai aplikasi enkripsi dekripsi.

Jefril Rahmadoni (2018) melakukan penelitian "Perancangan Simulasi Pembelajaran Kriptografi Klasik Menggunakan Metode *Web Based Learning*" Dalam penelitian ini, peneliti membuat sebuah aplikasi berbasis *web* untuk mempermudah pengguna dalam memahami materi kriptografi. Aplikasi ini bisa digunakan oleh kalangan mahasiswa melalui *web browser* namun memerlukan jaringan internet untuk dapat mengaksesnya.

II.2. Landasan Teori

II.2.1. Pengertian Aplikasi

Menurut Jogiyanto HM (dalam suhartini (2017), aplikasi merupakan penerapan, menyimpan sesuatu hal, data, permasalahan, pekerjaan ke dalam suatu sarana atau media yang dapat digunakan untuk diterapkan menjadi sebuah bentuk yang baru.

Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya aplikasi merupakan suatu perangkat komputer yang siap pakai bagi *user* (Helmi Fauzi Siregar, Dkk, 2018 : 113).

II.2.2. Pengertian Kriptografi *Caesar Cipher*

Kriptografi *Caesar Cipher* (sandi Caesar) biasa disebut juga sebagai sandi *Shift*, kode Caesar atau *Caesar's Shift* adalah salah satu teknik enkripsi yang paling sederhana dan paling dikenal luas. Ini adalah salah satu tipe dari sandi subtitusi dimana masing-masing huruf dari *plaintext* digantikan dengan huruf huruf dari abjad yang sebelumnya telah digeser urutannya terhadap suatu angka (Janner Simarmata, Dkk, 2019 : 32).

II.2.3. Pengertian Kriptografi *Reverse Cipher*

Kriptografi *Reverse Cipher* adalah suatu metode yang cara kerjanya dengan mengganti satu huruf dengan huruf yang lain. Sandi ini adalah contoh yang paling sederhana dari transposisi yaitu dengan mengubah suatu kalimat dengan cara menuliskan setiap kata secara terbalik (*reverse*) (Janner Simarmata, Dkk, 2019 : 63).

II.2.4. Pengertian Kriptografi *Affine Cipher*

Kriptografi *Affine Cipher* merupakan pengembangan dari metode *Caesar Cipher*. Pada *Affine, plaintext* akan dikalikan dengan sebuah nilai dan ditambahkan dengan sebuah pergeseran (Muhammad Fadlan, Dkk, 2018 : 270).

II.2.5. Pengertian *Adobe Animate CC*

Adobe Animate CC adalah suatu perangkat lunak hasil perkembangan dari versi terdahulu yaitu *Adobe Flash CS* yang memiliki keunggulan fitur-fitur dalam pembuatan animasi, *game* dan multimedia (Afista Galih Pradana, 2019 : 50).

II.2.6. Pengertian *Action Script 3.0*

Actionscript merupakan suatu *programming language* (bahasa pemrograman) yang dapat ditambahkan pada dokumen *Flash* (baik itu pada *frame, movie clip,* atau *button*) untuk dapat membuat suatu animasi yang lebih interaktif. Seperti halnya pada pemrograman C, C++ dan *Java*, *Action Script* memiliki sifat yang sangat sensitif (*case-sensitive*) artinya penulisan huruf sangat berpengaruh (Winda Angela Hamka, Dkk, 2016 : 82).

ActionScript merupakan bahasa pemrograman yang berjalan di lingkungan *Adobe Flash Player* dan *Adobe AIR*. *ActionScript* memungkinkan interaksi, data *handling*, dan lain-lain pada *Flash*, *Flex*, serta *AIR* konten dan aplikasi. *ActionScript* dieksekusi oleh sebuah *ActionScript Virtual Machine* (AVM), yang merupakan bagian dari *Flash Player* dan *AIR*. Kode *ActionScript* secara khusus di *compile* kedalam *bytecode* format (suatu bahasa pemrograman yang ditulis dan dimengerti

oleh komputer) oleh sebuah *compiler*, saling melengkapi seperti yang dibangun ke dalam *Adobe Flash CS6* atau *Adobe Flex Builder*, atau yang tersedia pada *Adobe Flex SDK*. *Bytecode* tertanam di dalam file SWF, yang mana dieksekusi oleh *Flash Player* dan *AIR* (Muhamad Eka Prasetya, 2018 : 14).

II.2.7. Pengertian *Android*

Menurut Athoillah dan Irawan (dalam Santoso, S. dkk. 2019:132) *Android* merupakan sebuah *Operating System* berbasis *Linux* yang dipakai oleh perangkat seluler seperti *Smartphone* dan *tablet*. Menurut Hermawan (dalam Yunus, Y. dan Sardiwan, M. 2018:32) mengemukakan bahwa *Android* merupakan suatu *Operating System* atau OS yang sampai saat ini masih dalam tahap perkembangan, OS ini seperti OS lainnya seperti *Symbian, IOS* di *I-Phone*, dan lain sebagainya. Berdasarkan uraian pendapat diatas dapat diartikan bahwa pengertian dari *Android* adalah suatu sistem operasi pada *smartphone* atau *tablet* yang mempunyai banyak fitur didalamnya untuk mempermudah kehidupan manusia dan sampai sekarang terus berkembang semakin canggih (Afista Galih Pradana, 2019 : 50).

Dalam pengembangannya *Android* telah mengalami cukup banyak pembaruan sejak awal dirilis yang akan ditunjukan pada Tabel II.1.

Tabel II. 1. Versi-versi Android

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
7.1	*Nougat*	4 Oktober 2016
7.4	*Nougat*	5 Desember 2016
8.0	*Oreo*	21 Agustus 2017
9.0	*Pie*	6 Agustus 2018
10.0	*Android Q*	7 Agustus 2019

(Sumber: Pramadana, 2018: 14)

II.2.8. Pengertian UML

Menurut Windu Gata, Grace (2013:4), *Unified Modeling Language (UML)* adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membanngun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Ade Hendini, 2016 : 108).

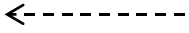
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

A. *Use Case Diagram*

Use Case diagram merupakan pemodelan untuk kelakuakn (*behavior*) sistem informasi yang akan dibuat. *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* akan ditunjukan pada Tabel II.2:

Tabel II. 2. Simbol *Use Case Diagram*

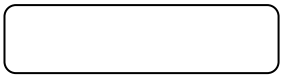
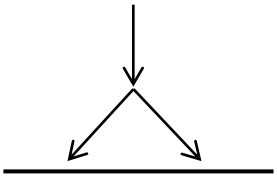
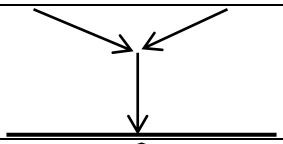
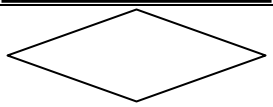
Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan 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 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 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:Ade Hendini, 2016 :108)

B. 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* akan ditunjukan pada Tabel II.3:

Tabel II. 3. Simbol Activity Diagram

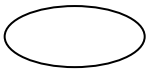
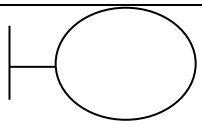
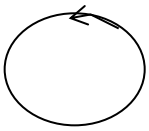
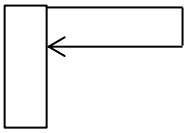
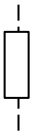
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara *parallel* atau untuk menggabungkan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

(Sumber:Ade Hendini, 2016 :109)

C. 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* akan ditunjukan pada Tabel II.4:

Tabel II. 4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, 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:Ade Hendini, 2016 :110)

D. Diagram Kelas (*Class Diagram*)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung 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 *Assosiations*, *Generalitation dan Aggregation*, attribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality* yang dapat dilihat pada tabel II.5 dibawah ini:

Tabel II. 5. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1atau 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 :111