BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Ashty M.Aaref dan Ann Z. Ablhd (2017) mengenai *A New Cryptography Method Based On Hill And Rail Fence Algorithma*, Ashty M.Aaref dan Ann Z. Ablhd menyimpulkan bahwa algoritma Rail Fence dan Hill Cipher dapat menyandikan sebuah pesan dengan keamanan yang baik.

Berdasarkan penelitian yang dilakukan oleh Andysah Putera Utama Siahaan (2017) mengenai *Rail Fence Cryptography* in *Securing Information*, Siahaan menyimpulkan bahwa untuk mendapatkan penyandian yang kuat pada *algoritma Rail Fence* maka harus dikombinasikan menggunakan metode yang lain misalnya menggunakan kunci.

Berdasarkan penelitian yang dilakukan oleh Retnani Latifah, dkk (2017) mengenai Modifikasi *Algoritma Caesar Chiper Dan Rail Fence* Untuk Peningkatan Keamanan Teks *Alfanumerik* Dan Karakter Khusus, Latifah, dkk menyimpulkan bahwa Metode *Caesar Cipher* menggunakan modulus 95 dengan sedikit perubahan rumus di enkripsi dan dekripsi. Selain itu pada setiap baris *Rail Fence* juga dilakukan *reverse* order untuk melakukan pengacakan urutan sehingga *ciphertext* tidak dengan mudah dipecahkan oleh pihak yang tidak diinginkan.

Berdasarkan penelitian yang dilakukan oleh Frisai Anistasari Sinaga dan Mesran (2017) mengenai Implementasi Algoritma ROT13 Dan Algoritma *Caesar*

Cipher Dalam Penyandian Teks, Frisai Anistasari Sinaga dan Mesran menyimpulkan bahwa Proses penyandian teks pada file teks ini merupakan salah satu cara untuk menjaga keaslian data teks dari para kriptanalis, sehingga para kriptanalis akan sulit dan membutuhkan waktu yang lama untuk mengetahui teks asli/*plaintext*.

II.2. Landasan Teori

Berikut ini adalah landasan teori yang digunakan pada penelitian ini . beberapa landasan teori berikut dikutip dari beberapa referensi yang berkaitan dengan penelitian :

II.2.1. Kriptografi

Kriptografi adalah seni tentang bagaimana agar teks menjadi aman dengan cara mengubah teks tersebut menjadi bentuk yang tidak dapat dibaca. Teks yang asli dan masih dapat terbaca disebut sebagai *plaintext*, sedangkan teks yang tidak dapat dibaca dan tidak bermakna disebut sebagai *ciphertext*. Proses pengubahan *plaintext* menjadi *ciphertext* disebut sebagai enkripsi (*encryption*), sedangkan proses pengembaliannya disebut sebagai dekripsi (*decryption*). Proses enkripsi digunakan untuk menyembunyikan informasi yang ada di teks dari orang yang tidak diinginkan. Kriptografi dapat diklasifikasikan menjadi tiga tipe yaitu kriptografi dengan kunci simetris, kriptografi dengan kunci asimetris dan kriptografi dengan fungsi hash. Pada kriptografi dengan kunci simetris, pengirim dan penerima teks menggunakan kunci rahasia yang sama untuk melakukan enkripsi dan dekripsi. Teknik ini hanya menerima plaintext yang berisi huruf alfabet, angka dan karakter-karakter khusus.

Sedangkan hasil *cipher text* dapat berupa alfabet, angka, karakter khusus maupun kombinasi ketiganya. Teknik kriptografi dengan kunci simetris dapat dikategorikan lagi, yaitu kriptografi klasik dan kriptografi modern. (Retnani Latifah, dkk, 2017 : 2).

Kriptografi klasik adalah teknik kriptografi yang telah dikembangkan dari zaman terdahulu, bahkan sebelum komputer ditemukan. Meskidemikian, sampai saat ini teknik kriptografi klasik masih digunakan untuk mengamankan informasi. Kriptografi klasik masih dapat dibagi lagi menjadi *cipher substitusi* (*substitution cipher*) dan *cipher transposisi* (*transpotition cipher*). *Cipher substitusi* adalah algoritma kriptografi yang mengganti setiap unit *plaintext* dengan satu unit *ciphertext*. Sedangkan *cipher* transposisi adalah mengubah urutan huruf *plaintext* atau melakukan *transpose* terhadap rangkaian karakter. (Retnani Latifah, dkk, 2017 : 2).

II.2.2. Metode *Railfence Cipher*

Rail Fence Cipher menuliskan *plaintext* dalam urutan diagonal dan membaca sebagai urutan baris sehingga terbentuk *ciphertext*. Contohnya jika ingin mengenkripsi HALO APA KABAR maka spasi dihilangkan kemudian disusun diagonal membentuk pola zig-zag. Misal ingin dibuat dengan kedalam (jumlah baris) 3, maka hasil perubahan susunannya adalah sebagai berikut:

H				**A**				**A**			
	A		**O**		**P**		**K**		**B**		**R**
		L				**A**				**A**	

Hasil *ciphertext* : HAA AOPKBR LAA. (Retnani Latifah, dkk, 2017 : 3).

II.2.3. ROT13

ROT13 (*Rotate* 13) adalah enkripsi *substitution cipher* yang umum digunakan di sistem operasi UNIX. Pada sistem enkripsi ROT13 sebuah huruf digantikan dengan huruf yang letaknya di atas 13 posisi darinya. (Frisai Anistasari Sinaga dan Mesran, 2017 : 38).

Enkrip metode ROT13 menggunakan penjumlahan *ASCII plaintext* dengan *ASCII* kunci 13 sebagai berikut :

$C_i = P_i + 13$

Keterangan :

C_i = *Ciphertext* hasil enkrip

P_i = *Plaintext* yang akan dienkrip

Dekrip metode ROT13 menggunakan pengurangan *ASCII ciphertext* dengan *ASCII* kunci 13 sebagai berikut :

$P_i = C_i - 13$

Keterangan :

C_i = *Ciphertext* yang akan didekrip

P_i = *Plaintext* hasil dekrip. (Frisai Anistasari Sinaga dan Mesran, 2017 : 39).

II.2.4. *Visual Basic* 2010

Visual Basic 2010 pada dasarnya adalah sebuah bahasa pemrograman komputer. Pengertian bahasa pemograman itu adalah perintah-perintah atau intruksi yang dimengerti oleh komputer untuk melakukan tugas-tugas tertentu. *Visual Basic*

juga sering disebut sebagai sarana (*tool*) untuk menghasilkan program-program aplikasi berbasiskan *windows*. (Ninuk Wiliani dan Syadid Zambi, 2017 : 78).

II.2.5. Basis Data (*Database*)

Basis Data adalah suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media tanpa adanya suatu kerangkapan data, sehingga mudah untuk digunakan kembali, dapat digunakan oleh satu orang atau lebih program aplikasi secara *optimal*, data disimpan tanpa mengalami ketergantungan pada program yang akan digunakan, data disimpan sedemikian rupa sehingga apabila ada penambahan, pengambilan dan *modifikasi* dapat dilakukan dengan mudah dan terkontrol. (Indra Kanedi, dkk. 2013 : 49)

II.2.6. MySQL

MySQL dapat digunakan untuk membuat dan mengola *database* beserta isinya. Kita dapat memanfaatkan MySQL untuk menambah, mengubah dan menghapus data yang berada dalam *database*. MySQL merupakan sistem manajemen *database* yang bersifat *atrelational* artinya data-data yang dikelola dalam dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat. (Ninuk Wiliani dan Syadid Zambi, 2017 : 79).

II.2.7. XAMPP

XAMPP adalah *sofware web server apache* yang didalamnya tertanam *server MySQL* yang didukung dengan bahasa pemograman PHP untuk membuat *website* yang dinamis. XAMPP sendiri mendukung dua sistem operasi yaitu *Windows* dan

Linux. Untuk *Linux* dalam proses penginstalannya menggunakan *interface grafis* sehingga lebih mudah dalam penggunaan XAMPP di *Windows* dibanding dengan *Linux*. (Ninuk Wiliani dan Syadid Zambi, 2017 : 79).

II.2.8. *Unified Modeling Language* (UML)

Menurut Windu Gata (2013 : 4) 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.

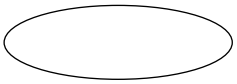
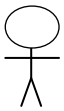
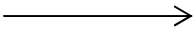
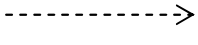
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

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 sebagai berikut:

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.

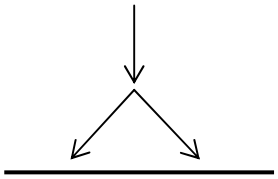
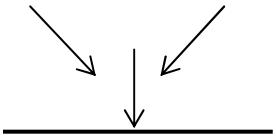
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

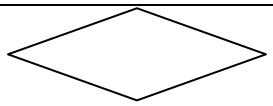
(Sumber : Windu Gata ; 2013 : 4)

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 sebagai berikut :

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.

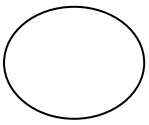
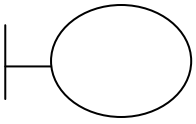
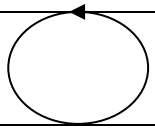
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
New Swimline	*Swimlane*, untuk menunjukkan siapa melakukan apa.

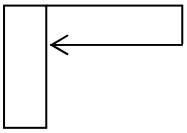
(Sumber : Windu Gata ; 2013 : 4)

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 sebagai berikut :

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 : Windu Gata ; 2013 : 7)

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 sebagai berikut :

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 : Windu Gata ; 2013 : 8)