BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Penelitian yang dilakukan oleh Naniek Widyastuti (2014) dengan Judul Pengembangan Metode Beaufort Cipher Menggunakan Pembangkit Kunci Chaos. Penelitian ini secara khusus membahas tentang bagaimana teori chaos diterapkan pada penyandian menggunakan metode Beaufort Cipher untuk meningkatkan keamanan pada kunci yang digunakan. Berdasarkan hasil pengujian yang dilakukan terhadadap citra yang diujikan menunjukkan bahwa algoritma Beaufort Cipher yang menggunakan kunci yang dibangkitkan menggunakan fungsi chaos terbukti efektif dan aman. Hal ini dibuktikan dengan rata-rata waktu proses yang dibutuhkan untuk melakukan proses enkripsi maupun dekripsi cukup cepat yaitu sekitar 0,7 detik. Dan berdasarkan pengujian secara visual dan uji statistik algoritma enkripsi yang digunakan tidak dapat memberikan petunjuk apa-apa untuk dilakukan statistical attack oleh kriptanalis.

Penelitian yang dilakukan oleh Mia Diana (2018) dengan judul Optimalisasi *Beaufort Cipher* Menggunakan Pembangkit Kunci RC4 Dalam Penyandian SMS. Kunci pada algoritma kriptografi sangat penting peranannya dalam proses enkripsi dan dekripsi. Semakin acak bilangan kunci yang digunakan, maka semakin acak pula cipher yang dihasilkan. Algoritma RC4 dan beaufort cipher merupakan algoritma dari

teknik kriptografi. Algoritma RC4 memiliki kelebihan dalam membangkitkan kunci yang acak, sedangkan beaufort cipher memiliki kelemahan dalam hal jumlah kunci yang digunakan terlalu banyak. Penelitian ini menguraikan bagaimana mengoptimalkan pembentukan kunci pada algoritma beaufort dengan memanfaatkan proses pembangkitan kunci pada algoritma RC4 yang diimplementasikan pada menyandikan SMS yang sampai saat ini belum bersifat point-to-point (tidak langsung dikirim kepada tujuan). Hasil penelitian ini memberikan kemudahan bagi pengguna dalam proses pembangkitan kunci enkripsi maupun dekripsi serta menghasilkan cipher SMS yang lebih acak dan sulit dipahami oleh pihak lain.

Penelitian yang dilakukan oleh De Rosal Ignatius Moses Setiadi (2018) dengan judul Kombinasi Cipher Subtitusi (Beaufort Dan Vigenere) Pada Citra Digital. Riset tentang kriptografi pada citra terus berkembang. Banyak metode yang telah diterapkan pada kriptografi citra. Algoritma Vigenere merupakan algoritma yang cukup populer dan masih dikembangkan sampai saat ini. Vigenere memiliki kelebihan dalam komputasi yang cepat, dan kuat terhadap serangan. Beaufort cipher murupakan salah satu turunan dari algoritma Vigenere yang menggunakan operator pengurangan pada kunci. Penelitian ini mengusulkan kombinasi algoritma Beaufort dan Vigenere cipher dengan menggunakan dua kunci untuk meningkatkan keamanan. Metode diusulkan dalam penelitian ini diimplementasikan untuk enkripsi pada

citra digital dan diukur dengan nilai MSE, PSNR dan analisis histogram. Hasil pengukuran dari kombinasi kedua metode ini didapatkan kualitas enkripsi yang lebih baik dibandingkan dengan metode Beaufort atau Vigenere saja.

Penelitian yang dilakukan oleh Arios, Jesfer Robin (2018) dengan judul Implementasi Algoritma Kriptografi Beaufort Cipher dan Algoritma Kompresi Reverse Unary Code Pada File Citra. Adapun jenis data yang digunakan adalah file citra dengan format Bitmap (*. bmp) dan hasil akhir kompresi jika didekompresi hasilnya adalah sebuah file asli dengan ekstensi (*.bmp). Dalam percobaan yang dirancang belum mampu memampatkan ukuran citra dengan Algoritma Reverse Unary Code sehingga pembesaran file ketika melakukan proses enkripsi dengan Algoritma Beaufort Cipher, dimana metode ini menghasilkan Ratio of compression (Rc)rata-rata sebesar 7,48%, Compression ratio (CR) 557,72% dan Space Saving (SS) -452,78%

Penelitian yang dilakukan oleh Angga Aditya Permana (2018) dengan judul Penerapan Kriptografi Pada Teks Pesan dengan Menggunakan Metode Vigenere Cipher Berbasis Android. Perkembangan teknologi khususnya dalam bidang komunikasi antar manusia sudah sangat mudah dilakukan dengan telepon genggam dan fiturnya sangat bervariasi. Pertukaran informasi jarak jauh ini menuntut keamanan terhadap kerahasiaan informasi yang dipertukarkan. Oleh karena itu, metode kriptografi dilakukan untuk mengamankan informasi

tersebut. Salah satu metode kriptografi untuk penyandian teks adalah metode Vigenere Cipher. Penelitian ini bertujuan untuk membangun aplikasi kriptografi teks pesan pada smartphone berbasis android dengan metode Vigenere Cipher. Metode ini mengenkripsi teks pesan menjadi pesan rahasia yang kemudian hasilnya diteruskan sebagai teks pesan ke aplikasi pengiriman pesan seperti aplikasi SMS (*Short Message Service*), Whatsapp, Line, dan sejenisnya untuk selanjutnya didekripsi. Penelitian ini menghasilkan aplikasi berbasis android yang dapat mengirimkan teks pesan terenkripsi menggunakan metode Vigenere Cipher untuk memberikan keamanan lebih pada proses pertukaran informasi

II.2. Landasan Teori

II.2.1. Keamanan Informasi data

Keamanan data menjadi hal yang sangat penting pada saat ini karena untuk setiap pengambilan keputusan, kebijakan harus berdasarkan data. Banyak data yang berisikan informasi penting dan terbatas untuk diketahui pihak yang terkait saja. Pada dunia perbankan banyak kegiatan yang melibatkan data nasabah yang harus diproteksi dan serta sifatnya rahasia. Banyak kegiatan yang akan menimbulkan resiko bilamana informasi yang sensitif dan berharga tersebut diakses oleh orang-orang yang tidak berhak (*unauthorized person*). Faktor keamanan data menjadi sangat penting dan harus diperhatikan. Salah satu cara untuk

meningkatkan keamanan data diperlukan kriptografi dengan metode enkripsi (Pratiwi ; 2016 : 132)

II.2.2. Algoritma Beafort Chiper

Beaufort Cipher merupakan salah satu algoritma dalam teknik keamanan kriptografi klasik. Kunci (K) pada *Beaufort Cipher* adalah urutan karakter-karakter *K* = *k*1 ... *k*d dimana *k*1 didapat dari banyaknya pergeseran dari alfabet ke-*i* sama seperti *Viginere Cipher*. Artinya bahwa jumlah kunci yang dibangkitkan harus sama dengan jumlah karakter *plaintext* yang diamankan. Algoritma ini melakukan proses *enkripsi* dan *dekripsi* secara *stream* (masing-masing karakter *plaintext* harus memiliki pasangan kunci). Hal ini yang menyebabkan algoritma ini sama hampir sama dengan algoritma *Vigenere Cipher* (Mia Diana, 2018:15)

Formula beaufort cipher : Misalkan m menentukan beberapa nilai integer positive.

$P= C = K = (Z26)^{m}$

untuk sebuah kunci K = (k1, k2, …, km),

kita definisikan : eK(x1, x2, …, xm) = (k1 -x1 , k2 - x2, …, km - xm) ……(2.1)

dK(y1, y2, …, ym) = (k1- y1, k2 - y2, …, km - ym) ……(2.2)

dimana semua operasi adalah berbasis pada Z26 Rumus enkripsi yang digunakan untuk menghitung nilai cipher image tiap pixel adalah sebagai berikut :

Eki(a) = (a – ki)mod 256

……...(2.3)

Sedangkan rumus yang digunakan untuk mendapatkan kembali plainteks yang berupa image tiap pixel yang telah terenkripsi (dekripsi) adalah:

Eki(a) = (a + ki)mod 256

……(2.4)

II.2.3. Netbeans

NetBeans mengacu pada dua hal, yakni platform untuk pengembangan aplikasi desktop java, dan sebuah *Integrated Development Environment* (IDE) yang dibangun menggunakan platform NetBeans. Platform NetBeans memungkinkan aplikasi dibangun dari sekumpulan komponen perangkat lunak moduler yang disebut ‘modul’. Sebuah modul adalah suatu arsip Java (Java *archive*) yang memuat kelas-kelas JAVA untuk berinteraksi dengan *NetBeans Open API* dan file manifestasi yang mengidentifikasinya sebagai modul. Aplikasi yang dibangun dengan modul-modul dapat dikembangkan dengan menambahkan modul-modul baru. Karena modul dapat dikembangkan secara independen, aplikasi berbasis platform NetBeans dapat dengan

mudah dikembangkan oleh pihak ketiga secara mudah dan *powerful.* (Maria ; 2015 : 25)

II.2.4. *Java*

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telephone genggam.

1. Kelebihan *Java* Berikut adalah beberapa keunggulan *Java* dibanding dengan bahasa pemrograman lain :
    a. *Java* merupakan bahasa yang sederhana.
    b. *Java* dirancang agar mudah dipelajari dan digunakan secara efektif.
    c. *Java* tidak menyediakan fitur-fitur rumit bahasa pemrograman tingkat tinggi.
    d. Banyak pekerjaan pemrograman yang mulanya harus dilakukan secara manual, sekarang sudah dilakukan secara otomatis di *Java* seperti dealokasi memori.
2. Kekurangan *Java*
    a. *Write once, debug everywhere*. Ada beberapa hal yang tidak kompatibel antara platform satu dengan platform lain. Untuk J2SE, misalnya *SWT-AWT bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
    b. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari *executable code* menjadi *source code*. Ini

dimungkinkan karena *executable Java* merupakan *byte code* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, method, dan tipe data.

c. *Eavy memory usage*. Penggunaan memori untuk program berbasis *Java* jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal (lebih spesifik lagi, *Delphi* dan *Object Pascal*). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena *trend* memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin desktop berumur lebih dari 4 tahun (Sulihati ; 2016 : 18)

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

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.

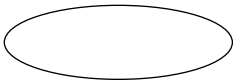
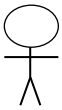
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

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

- *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, yaitu :

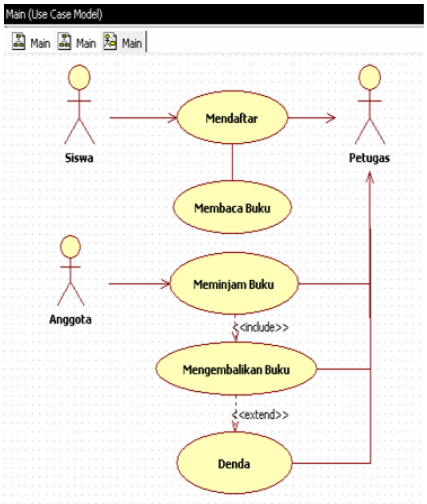
Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya 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)

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.1 berikut :



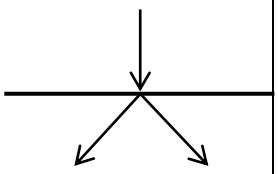
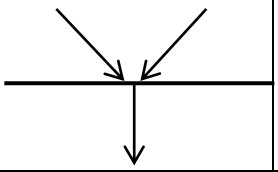
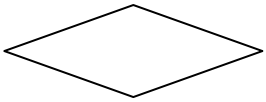
Gambar. II.1. *Use Case Diagram*
(Sumber : Windu Gata ; 2013 : 4)

- 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*, yaitu :

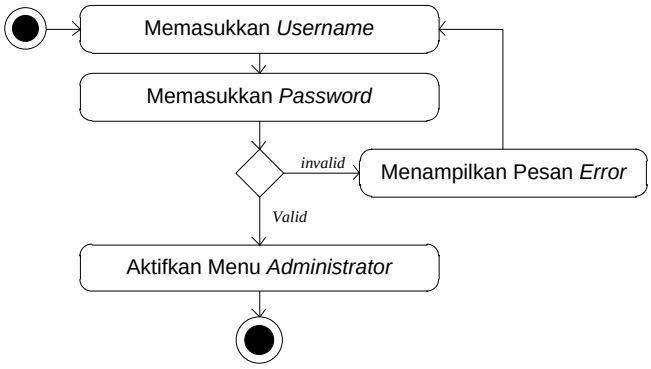
Tabel II.2. Simbol *Activity Diagram*

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.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

Contoh dari pembuatan *activity diagram* dapat dilihat pada gambar II.2 berikut :

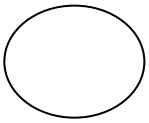
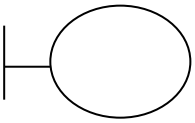
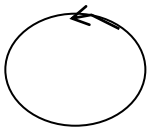


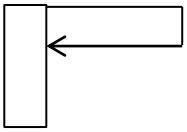
Gambar. II.2. *Activity Diagram*
(Sumber : Windu Gata ; 2013 : 6)

- 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*, yaitu :

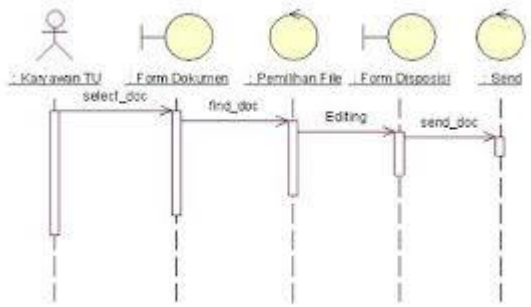
Tabel II.3. Simbol *Sequence Diagram*

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)

Contoh dari pembuatan *sequence diagram* dapat dilihat pada gambar II.3 berikut :



Gambar. II.3. *Sequence Diagram*
(Sumber : Windu Gata ; 2013 : 7)

- *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.

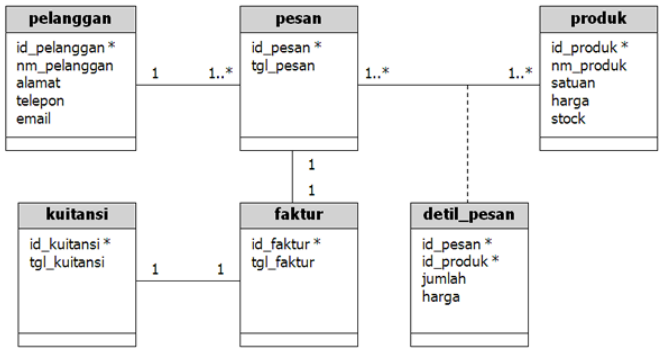
Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

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)

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.4 berikut :



Gambar. II.4. *Class Diagram*
(Sumber : Windu Gata ; 2013 : 8)