BAB II

LANDASAN TEORI

2.1. Tinjauan Jurnal

Agar aplikasi yang dibuat penulis dapat dipertanggung jawabkan secara akademis maka penulis akan menampilkan penelitian sebelumnya yang berkaitan dengan aplikasi yang akan dibuat penulis.

Menurut Noercholis (2014:3) mengatakan bahwa "*vigenere chiper* merupakan salah satu algoritma klasik dengan teknik substitusi. Teknik substitusi adalah teknik pada kriptografi yang menggunakan angka dengan menukarkan huruf dan angka agar dapat menghasilkan pesan yang sudah terenkripsi. Teknik subsitusi itulah yang merupakan kelemahan pada algoritma *vigenere chipper* karena teknik substitusi hanya menukar pesan yang akan dienkripsi dengan angka sehingga tidak sulit untuk mengetahui pesan aslinya (*plaintext*).

Oleh karena itu, penulis akan menggunakan metode RC4 dalam enkripsi dan dekripsi sms. Alasan penulis menggunakan menggunakan metode RC4 karena menurut penulis metode RC4 lebih rumit dan sulit untuk dipecahkan oleh orang lain daripada metode *vigenere chipper*.

2.1. Konsep Dasar Program

1. Java

> Menurut Wardhani (2013:474) mengemukakan bahwa
> *Java* adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. *Java* pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak *system* komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa *Java* memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk digunakan.

Java sampai saat ini masih merupakan bahasa pemrograman yang masih sangat di minati dan banyak digunakan oleh para progremer dan software developer untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi *console*, aplikasi *desktop*, *game*, dan applet (aplikasi yang berjalan di lingkungan *web browser*), sampai ke aplikasi-aplikasi yang berskala *enterprise*. Untuk memenuhi kebutuhan tipe aplikasi yang beragam tersebut, *Java* dikategorikan menjadi tiga edisi, yaitu: J2SE (*Java 2 Platform Standart Edition*) untuk membuat aplikasi-aplikasi desktop dan apllet, J2EE (*Java 2 Platform Enterprise Edition*) untuk membuat aplikasi-aplikasi *multitier* berskala *enterprise*, dan J2ME (*Java 2 Platform Micro Edition*) untuk membuat aplikasi-aplikasi yang dapat dijalankan dilingkungan perangkat-perangkat mikro seperti handphone, PDA dan Smartphone.

2. XML

Extensible Markup Language (XML) adalah bahasa markup serba guna yang direkomendasikan oleh W3C (*World Wide Web Consortium*) untuk mendeskripsikan berbagai macam data. XML menggunakan *markup tags* seperti

halnya HTML (*Hypertext Markup Language*) namun penggunaannya tidak terbatas pada tampilan halaman web saja. XML adalah sebuah cara untuk mendeskripsikan tipe data dan struktur data. XML merupakan sebuah cara merepresentasikan data tanpa tergantung pada sistem operasi maupun pada program aplikasi. XML adalah bahasa berbasis *text* sehingga XML dapat dengan mudah dipindahkan dari satu sistem komputer ke sistem komputer yang lain. Dengan XML, data direpresentasikan dalam sebuah dokumen yang terstruktur. Dokumen XML adalah dokumen yang terdiri dari XML *tag* atau *element*. Sama halnya dengan HTML, XML tag didefinisikan dengan kurung siku < >. XML dapat menggunakan *tag* sesuai dengan yang diinginkan, selama semua aplikasi yang menggunakan dokumen tersebut menggunakan *tag* dengan nama yang sama.

> Menurut Setiyawati dkk (2016:152) mengatakan bahwa
> XML merupakan sebuah teknologi *cross platform*, dan merupakan *tool* untuk melakukan transmisi informasi. XML bukanlah program, atau pustaka. XML adalah sebuah teknologi, sebuah standar dengan berbagai aturan tertentu. Dalam pengertian yang sederhana, sebuah dokumen XML hanyalah sebuah file teks biasa yang berisikan berbagai tag yang didefinisikan sendiri oleh pembuat dokumen XML tersebut. Sesuai dengan namanya, *Extensible Markup Language*, sebuah dokumen XML adalah sebuah dokumen dengan markup, sama seperti halnya dengan HTML. XML bukanlah hal baru dan bukan merupakan pengganti HTML.Keduanya mempunyai fungsi yang berbeda dalam penerapannya. XML ditujukan untuk fokus pada data,sedangkan HTML ditujukan untuk cara menampilkan data. XML merupakan sintaks yang digunakan untuk menjelaskan bahasa markup lain, sehingga dinamakan *meta language*.

Bagian-Bagian dari Dokumen XML

1. *Root node* yaitu node yang melingkupi keseluruhan dokumen. Dalam satu dokumen xml hanya ada satu root node. Node-node yang lainnya berada di dalam root node.
2. *Element node* yaitu bagian dari dokumen XML yang ditandai dengan tag pembuka dan tag penutup, atau bisa juga sebuah tag tunggal elemen kosong seperti <anggota nama=”ilham”/> . Root node biasa juga disebut root element.
3. *Attribute node* termasuk nama dan nilai atribut ditulis pada tag awal sebuah elemen atau pada tag tunggal.
4. *Text node* adalah text yang merupakan isi dari sebuah elemen, ditulis diantara tag pembuka dan tag penutup.
5. *Comment node* adalah baris yang tidak dieksekusi oleh parser.
6. *Processing Instruction node* adalah perintah pengolahan dalam dokumen XML. Node ini ditandai awali dengan karakter <? Dan diakhiri dengan ?>. Tapi perlu diingat bahwa header standard XML <?xml version=”1.0” encoding=”iso-8859-1”?> bukanlah processing instruction node. Header standard bukanlah bagian dari hirarki pohon dokumen XML.
7. *NameSpace Node* merupakan node yang mewakili deklarasi namespace.

Sama dengan HTML, File XML berbentuk teks sehingga bila diperlukan kita bisa membacanya tanpa memerlukan bantuan software khusus. Hal ini memudahkan pengembang aplikasi yang menggunakan XML untuk mendebug programnya. XML lebih fleksible dibanding HTML dalam hal kemampuannya menyimpan informasi dan data. Pada XML kita bisa menyimpan data baik dalam atribut maupun sebagai isi elemen yang diletakkan diantara tag pembuka dan tag penutup.

3. Android

Menurut Zurnawita (2014:91) menjelaskan bahwa "*Android* adalah sistem operasi yang digunakan di smartphone dan juga tablet PC. Fungsinya sama seperti sistem operasi Symbian di Nokia, iOS di *Apple* dan *BlackBerry* OS".

Android tidak terikat ke satu merek *handphone* saja, beberapa vendor terkenal yang sudah memakai *android* antara lain Samsung, Sony Ericsson, HTC, Nexus, Motorolla, dan lain-lain. *Android* pertama kali dikembangkan oleh perusahaan bernama *Android* Inc., dan pada tahun 2005 di akuisisi oleh raksasa internet google.

Android dibuat dengan basis kernel Linux yang telah dimodifikasi, dan untuk setiap release-nya diberi kode nama berdasarkan nama hidangan makanan seperti *Enclair*, *Frozen Yogurt* (Froyo), *Ice Cream Sandwich*, *Jelly Bean*, Kitkat. Keunggulan utama *Android* adalah gratis dan *open source*, yang membuat smartphone *Android* dijual lebih murah dibandingkan dengan *Blackberry* atau iPhone meski fitur (*hardware*) yang ditawarkan *Android* lebih baik. Beberapa fitur utama dari *android* antara lain WiFi *hotspot*, *Multi-touch*, *Multitasking*, GPS, accelerometers, support java, mendukung banyak jaringan *(GSM/EDGE, IDEN,*

CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE & WiMAX) serta juga kemampuan dasar handphone pada umumnya.

4. OOP (*Object Oriented Programming*)

Menurut Fuspita dkk (2014:47) menjelaskan bahawa "*Object Oriented Programming* (OOP) atau pemrograman berorientasi objek adalah suatu cara baru dalam berpikir serta berlogika dalam menghadapi masalah–masalah yang akan dicoba diatasi dengan bantuan komputer". Apabila dilihat dari perbandingan antara pemrograman dengan teknik prosedural dan OOP sekilas sama namun berbeda, sebenarnya konsep dasar dari pembuatan program adalah bagaimana informasi dapat diberikan kepada *user*. Apapun teknik pemprograman yang diberikan pada hakekatnya sama proseduralnya maupun OOP masing-masing mempunyai kelebihan dan kekurangan.

2.3. Metode Algoritma

Algoritma adalah prosedur komputasi yang mengambil beberapa nilai atau kumpulan nilai sebagai input kemudian di proses sebagai output sehingga algoritma merupakan suatu urutan langkah komputasi yang mengubah input menjadi output. Dalam penulisan skripsi ini, penulis menggunakan algoritma RC4.

RC4 merupakan algoritma *stream chiper* yang dibuat oleh RSA *Data Security, Inc* (RSADSI). Menurut Hakim (2014:1) menjelaskan bahwa "RC4 (*Rivest Cipher* 4) adalah sebuah *synchrone streamcipher*, yaitu *cipher* yang memiliki kunci simetris dan mengenkripsi *plainteks* secara digit per digit atau *byte*

per *byte* dengan cara mengkombinasikan dengan operasi biner (biasanya X*OR*) dengan sebuah angka semi acak".

RC4 menggunakan dua buah kotak substitusi (*S-Box*) *array* 256 *byte* yang berisi permutasi dari bilangan 0 sampai 255 dan *S-box* kedua yang berisi permutasi fungsi dari kunci sepanjang variabel. Langkah-langkah pada algoritma RC4 adalah sebagai berikut :

1. Lakukan inisialisasi dua buah *s-box* (I dan j)

   ```
   int[] sbox = new int[SBOX_LENGTH];
       int j = 0;
       for (int i = 0; i < SBOX_LENGTH; i++)
           sbox[i] = i;
   ```

   dimana *s-box* i digunakan untuk bilangan permutasi dan *s-box* j digunakan untuk key sepanjang variable.

2. Lakukan pengacakan *S-BOX* untuk membuat kunci untuk enkripsi

   ```
   for (int i = 0; i < SBOX_LENGTH; i++) {
           j = (j + sbox[i] + key[i % key.length]) % SBOX_LENGTH;
           swap(i, j, sbox);
   ```

2.4. Pengujian Aplikasi

Sebelum perangkat lunak dibuat, diimplementasikan dan digunakan oleh user terlebih dahulu perangkat lunak harus dilakukan pengujian. Pengujian pada perangkat lunak berfungsi untuk melakukan pengecekan terhadap perangkat lunak

yang sudah dibuat apakah perangkat lunak yang dibuat sudah benar serta tidak ada kesalahan pada perangkat lunak yang sudah dibuat.

Penulis menggunakan 2 metode pengujian aplikasi yaitu *metode white box testing dan black box testing.* Kedua pengujian ini dapat diterapkan di semua lingkungan arsitektur dan aplikasi.

1. Metode Pengujian *White Box*

> Menurut Mustaqbal dkk dalam Nidhra *and* Dondetti (2015:33) menjelaskan bahwa *White Box Testing* adalah salah satu cara untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Kalau modul yang telah dan sudah di hasilkan berupa output yang tidak sesuai dengan yang di harapkan maka akan dikompilasi ulang dan di cek kembali kode-kode tersebut hingga sesuai dengan yang diharapkan.

Pengujian perangkat lunak perlu dilakukan untuk mengevaluasi baik secara manual maupun otomatis untuk menguji apakah perangkat lunak sudah memenuhi persyaratan atau belum, dan untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.

2. Metode Pengujian *Black box*

Menurut Parmawati (2014:352) mengjelaskan bahwa “Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak”. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan. Pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangakaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu

program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut :

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau *database.*
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

2.5. Peralatan Pendukung

Penulis akan menjelaskan beberapa peralatan pendukung yang digunakan dalam pembuatan aplikasi ini :

1. UML (*United Modelling Language*)

Menurut Sanjani dkk dalam Nugroho (2014:88) menjelaskan bahwa "UML adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek". Pemodelan sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

a. Use case Diagram

Menurut Paryanta (2015:72) menjelaskan bahwa "*Use case*.adalah abstraksi dan interaksi antara sistem dan *actor*". *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem

akan terlihat di mata user. Sedangkan *use case diagram* memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client.

b. Activity Diagram

Menurut Paryanta (2015:72) menjelaskan bahwa "*Activity diagram* menggambarkan rangkaian aliran dari aktivitas, digunakan untuk aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti use case atau interaksi".

c. Sequence Diagram

Menurut Paryanta (2015:72) menjelaskan bahwa *"Sequence Diagram* digunakan untuk menggambarkan perilaku pada sebuah scenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem".

d. Class Diagram

Menurut Paryanta (2015:72) menjelaskan bahwa "*Class* adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama". Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class-class yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem biasanya mempunyai beberapa class diagram. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

e. Deployment Diagram

Menurut Shalahuddin (2013:154) menjelaskan bahwa "Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen

dalam proses eksekusi". *Deployment diagram* berhubungan erat dengan diagram kompoen dimana *deployment diagram* memuat satu atau lebih komponen – komponen. *Diagram* ini sangat berguna saat aplikasi berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

2. *Java Development Kit* (JDK)

Menurut Chandra dkk (2016:28) mengatakan bahwa "*Java Dvelopment Kit* (JDK) adalah sebuah produk yang dikembangkan oleh Oracle yang ditujukan untuk *developer* Java". JDK terdiri dari *run time environment* yang ada diatas layer sistem operasi serta *tool*s dan program yang memerlukan *compile, debug*, dan *run applets* dan aplikasi yang ditulis menggunakan bahasa pemrograman *Java*. *Java Develovment Kit* digunakan untuk *plugin* bahasa pemrograman *java*.

3. *Android Software Development Kit* (SDK)

Menurut Kadir (2013:2) menjelaskan bahwa "Android SDK adalah kumpulan *software* yang berisi mengenai pustaka, *debugger* (alat pencari kesalahan program), *emulator* (peniru perangkat bergerak), dokumentasi, kode contoh dan panduan".

4. *Android Development Tools* (ADT)

Menurut Marlena dalam Safaat H (2014:163) menngatakan bahwa "*Android Development Tools* (ADT) adalah plugin yang di desain untuk IDE *Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi *android* dengan menggunakan IDE *Eclipse*". Dengan

menggunakan ADT untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi project *android,* membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga kita dapat melakukan *running* aplikasi menggunakan *Android* SDK melalui *Eclipse*. Dengan ADT juga kita dapat melakukan pembuatan *package android* (apk) yang digunakan untuk distribusi aplikasi *android* yang kita rancang.

Semakin tinggi *platform android* yang kita gunakan, di anjurkan menggunakan ADT yang lebih terbaru, karena biasanya munculnya *platform* baru di ikuti oleh munculnya versi ADT yang terbaru. Untuk melakukan instalasi ADT di *Eclipse* dapat di lakukan secara *online* maupun *offline.*