BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Berdasarkan hasil penelitian yang dilakukan oleh Arief, dkk (2015) Algoritma asimetris seperti RSA dapat diterapkan dalam perangkat lunak FTP *client* dengan mengorbankan waktu *upload* tetapi menghasilkan keamanan yang lebih baik. Selain itu terjadi peningkatan ukuran *file* karena mekanisme base64 dan enkripsi RSA yang diaplikasikan ke setiap *byte* pada *file* yang akan dienkripsi. Proses enkripsi dan dekripsi algoritma RSA dapat di implementasikan ke semua *file*, namun dikarenakan keterbatasan memori JVM yang dapat digunakan maka pada penelitian ini *file* yang dapat di enkripsi berukuran maksimal sekitar 9 *megabytes.*Keutuhan *file* tetap terjaga bila serangkaian proses enkripsi, dekripsi, upload dan download dilakukan dengan program yang telah dibuat. Selain itu, *chipper file* juga dapat di-*download* dan didekripsi menggunakan kunci RSA lain tetapi akan diperoleh *file* seperti aslinya. Berdasarkan hasil analisis, algoritma RSA yang diterapkan pada *file* dalam suatu jaringan komputer dapat meningkatkan keamanan pada *file* walaupun ditransfer dalam jaringan publik.

Berdasarkan hasil penelitian yang dilakukan oleh Gilang Gumira, dkk 2016, mengenai Algoritma *Advanced Encryption Standard* dan Algoritma *Message Digest 5* dalam enkripsi dokumen berbasis *Website*, File yang dihasilkan oleh proses enkripsi bisa menjadi 2 kali lipat bahkan lebih dari ukuran dan jumlah

karakter file aslinya, ini dikarenakan hasil enkripsi dibuat dalam hex. Satu karakter diwakili dua karakter hex, misalkan karakter “U” dalam hex menjadi “75” begitu juga karakter simbol dan spasi juga ada hex-nya. Jadi file hasil enkripsi bisa 2 kali bahkan lebih dari ukuran dan jumlah karakter file aslinya.

Berdasarkan hasil penelitian yang dilakukan oleh Pahrizal dan David Pratama (2016) Aplikasi pengamanan data menggunakan algoritma *RSA* mempunyai dua teknik pembacaan yaitu teknik enkripsi (mengubah *file* asli menjadi *file* yang tidak dapat dibaca) dan teknik dekripsi (mengubah *file* yang tidak dapat dibaca menjadi *file* asli). Aplikasi pengamanan mempunyai kalimat sandi / *passphare* yang harus diingat dan bersifat sensitif, maksudnya huruf besar dan kecil dibedakan,agar *passphare* sulit ditebak oleh siapapun. Setelah melakukan uji coba, *file* yang telah diterapkan aplikasi pengamanan akan memiliki empat aspek keamanan, yaitu kerahasiaan, integritas data, otentikasi, dan nir-penyangkalan.

II.2. LandasanTeori

II.2.1. Aplikasi

Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah Program yang dibuat oleh manusia yang berfungsi untuk menyelesaikan permasalahan-permasalahan masalah yang akan dihadapi. (Zulfauzi, 2015 : 57).

II.2.2. *Hypertext Markup Language* (HTML)

Harisondan Ahmad Syarif (2016) Mengemukakan bahwa *HyperText Markup Languag*e (HTML) adalah sebuah bahasa *markup* yang digunakan untuk membuat sebuah halaman *web*, menampilkan berbagai informasi di dalam sebuah Penjelajah *web* Internet dan formating *hypertext* sederhana yang ditulis kedalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan kedalam format ASCII normal sehingga menjadi home page dengan perintah-perintah HTML.

II.2.3. *Website*

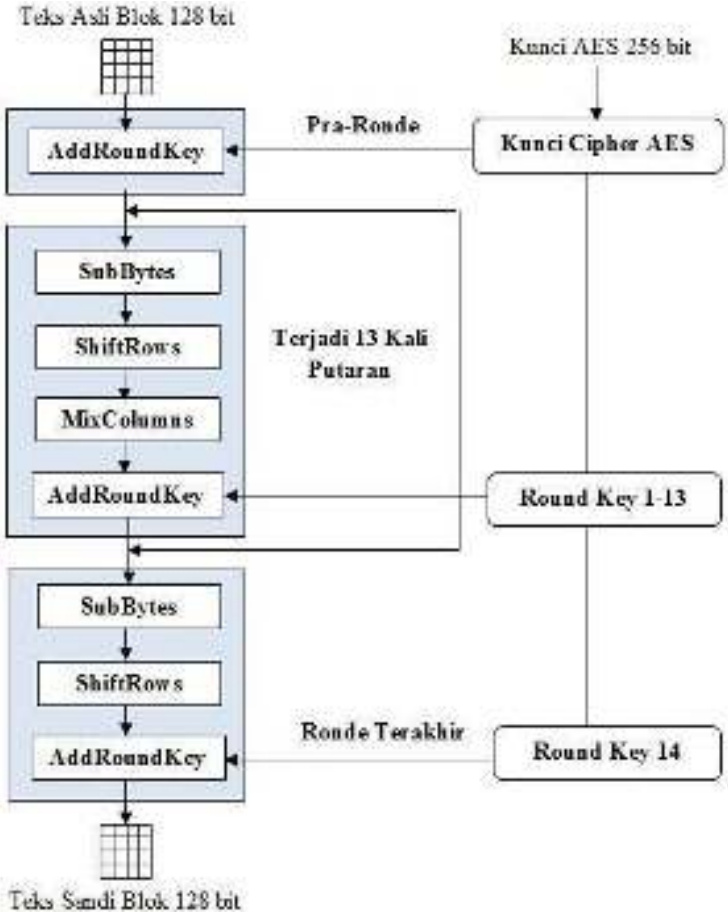
Sebuahsitus *web* (sering pula disingkat menjadi situs saja, *website* atau *site*) adalah sebutan bagi sekelompok halaman *web* (*web page*), yang umumnya merupakan bagian dari suatu nama *domain* (*domain name*) atau *subdomain* di *World Wide Web* (WWW) di Internet. Sebuah *web page* adalah dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*), yang hampir selalu bisa diakses melalui HTTP, yaitu protokol yang menyampaikan informasi dari *server website* untuk ditampilkan kepada para pemakai melalui *web browser* baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*). (Ali Zaki, 2009 dalamRudika, 2014).

II.2.4. Algoritma *Advanced Encryption Standart*

Algoritma kriptografi bernama Rijndael yang didesain oleh Vincent Rijmendan John Daemen asal Belgia keluar sebagai pemenang kontes algoritma kriptografi pengganti DES yang diadakan oleh NIST (*National Institutes of Standards and Technology*) milik pemerintah Amerika Serikat pada 26 November 2001 (Gilang Gumira,et al.2016a).Algoritma Rijndael inilah yang kemudian dikenal dengan *Advanced Encryption Standard* (AES). Setelah mengalami beberapa proses standarisasi oleh NIST, Rijndael kemudian diadopsi menjadi standard algoritma kriptografi secara resmi pada 22 Mei 2002. Pada 2006, AES merupakan salah satu algoritma terpopuler yang digunakan dalam kriptografi kunci simetrik (Rijmen, 1998 dalam Gilang Gumira,et al.2016b).

1. Enkripsi AES

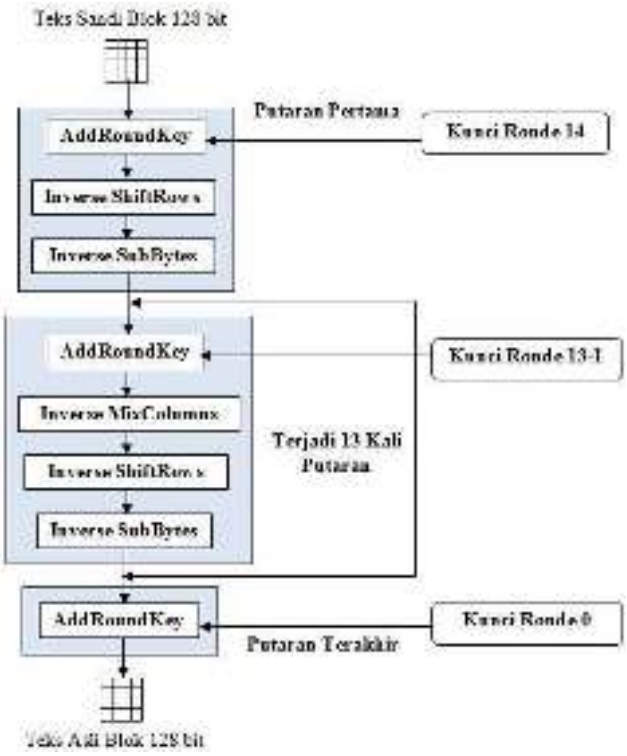
Pada awal proses enkripsi, input yang telah dicopykan ke dalam *state* akan mengalami transformasi *byte AddRoundKey*. Setelah itu, state akan mengalami transformasi *SubBytes, ShiftRows, MixColumns*, dan *AddRoundKey* secara berulang - ulang sebanyak Nr. Proses ini dalam algoritma AES disebut sebagai *round function. Round* yang terakhir agak berbeda dengan *round – round* sebelumnya, dimana pada *round* terakhir *state* tidak mengalami transformasi *MixColumns*.



GambarII.1. Diagram AlirAlgoritma AES Enkripsi

2. Dekripsi AES

Transformasi chipher dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi *byte* yang digunakan pada invers cipher adalah *InvShiftRows, InvSubBytes, InvMixColumns,* dan *AddRoundKey*.



GambarII.2. Diagram AlirAlgoritma AES Dekripsi

II.2.5. Algoritma RSA

Dari banyak algoritma kriptografi asimetris yang ada, algoritma yang paling populer adalah RSA. Algoritma RSA dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976. Nama RSA merupakan singkatan dari nama tiga orang penemunya, yaitu Rivest, Shamir,dan Adleman (Arief Muhamad, et al. 2015c).

Algoritma RSA melakukan pembankitan kunci dimana kegiatan untuk menghasilkan dua buah kunci yang berbeda yaitu kunci privat (*private key*) yang

digunakan untuk mendekripsi pesan dan kunci publik (*public key*) yang digunakan untuk mengenkripsi pesan yang mana biasanya di sebarkan kepada pihak yang akan mengirimkan pesan kepada pemilik
kunci.

Proses pembangkitan kunci dapat dilakukan dengan menggunakan tahap – tahap berikut :

1. Memilih dua bilangan prima p dan q.
2. Menghitung nilai n = p.q (Persamaan II.1)
3. Menghitung nilai $\phi(n) = (p - 1).(q - 1)$ (Persamaan II.2)
4. Memilih sebuah bilangan bulat (e) dimana $1 < e < \phi$ (n) yang relatif prima terhadap ($\phi(n)$).
5. Menghitung nilai (d), dimana $1 < d < \phi$ (n) dimana $e . d = 1 \pmod{\phi (n)}$.

Berdasarkan tahapan pembangkitan kunci diatas maka dapat diperoleh kunci *public* dari variabel (n) dan (e) sedangkan kunci *private* diperoleh dari variabel (d).

Proses enkripsi pada algoritma RSA merupakan operasi enkripsi yang cukup sederhana yang mana dapat dijabarkan pada tahapan – tahapan berikut :

1. Membentuk pesan kedalam blok – blok pesan (p1, p2, …. , pn) dan mengkonversi blok pesan tersebut kedalam bilangan integer.
2. Melakukan enkripsi pada blok – blok pesan tersebut menggunakan persamaan berikut :

$$c = m^e \bmod n \qquad \text{(Persamaan II.3)}$$

Di mana :

c = integer pesan ter-enkripsi

m = integer pesan asli

e dan n = kunci publik

Dekripsi pada RSA merupakan proses kebalikan dari proses enkripsi yang mana menggunakan kunci privat dalam operasinya. Tahapan dalam proses dekripsi pada algoritma RSA dapat dijabarkan sebagai berikut :

1. Membentuk pesan *cipherteks* kedalam blok – blok pesan (p1, p2, …. , pn) dan mengkonversi blok pesan tersebut kedalam bilangan integer.
2. Melakukan enkripsi pada blok – blok pesan tersebut menggunakan persamaan berikut :

$$m = C^d \bmod n \qquad \text{(Persamaan II.4)}$$

Di mana :

c = integer pesan ter-dekripsi

m = integer pesan asli

n = kunci publik

d = kunci privat

***Visual Basic Net.* 2010**

Microsoft Visual Studio adalahsebuah Integrated Development Environment buatan Microsoft Coroporation. Microsoft Visual Studio dapatdigunakanuntukmengembangkanaplikasidalam native code (dalambentukbahasamesin yang berjalan di atas*Windows*) ataupun managed code (dalambentuk Microsoft Intermediate Language di atas .NET *Framework*). Selainitu, Visual Studio juga dapat digunakan untuk mengembangkan aplikasi Silverlight, aplikasi Windows Mobile (yang berjalan di atas .NET *Compact Framework*). Visual Basic mencakup sebuah kode editor yang didukung oleh fitur intellisense atau yang disebut dengan code refactoring. Debugger telah terintegrasi bekerja pada level source level debugger dan level debugger mesin. Toll built in mencakup form desainer untuk membangun sebuah aplikasi GUI, *web desainer*, *class desainer* dan *database schema desainer* (Edy Winarno,et el,2010:8 dalam Nancy dan Supriandi 2017).

II.2.7. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan visual yang digunakan untuk menspesifikasikan, memvisualisasikan, membangun, dan mendokumentasikan rancangan dari suatu sistem perangkat lunak (Rumbaugh, Jacobson, &Booch, 2005 dalam Ibnu Akil,2016:1)

UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan

desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek. (Zulfauzi, 2015 : 59).

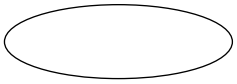
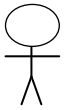
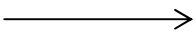
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem(UrvadanSiregar, 2015 : 93).

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 dibawah ini:

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukarpesan 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.

Lanjutan tabel simbol *Use Case*

Gambar	Keterangan
	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:UrvadanSiregar, 2015 : 94)

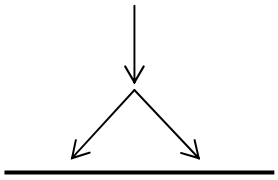
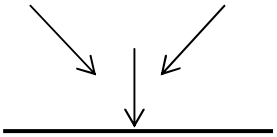
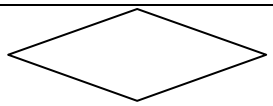
2. Diagram Aktivitas (*Activity Diagram*)

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 dibawah ini:

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.

Lanjutan tabel simbol *Activity* Diagram

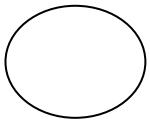
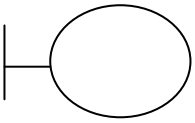
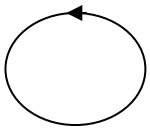
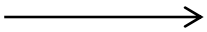
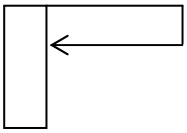
Gambar	Keterangan
	Activites, menggambarkan suatu proses/kegiatan bisnis.
	Fork (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	Join (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	Decision Points, menggambarkan pilihan untuk pengambilan keputusan, true, false.
	Swimlane, untuk menunjukkan siapa melakukan apa.

(Sumber : UrvadanSiregar, 2015 : 94)

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 dibawah ini :

Tabel II.3. 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.

Lanjutan tabel simbol *sequence* diagram

Gambar	Keterangan
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation.

(Sumber : UrvadanSiregar, 2015 : 95)

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 dibawah ini:

Tabel II.4. *Multiplicity Class Diagram*

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

Lanjutan tabel simbol *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
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 : UrvadanSiregar, 2015 : 95)