BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Gilang pada tahun 2016 melakukan penelitian yang berjudul "Implementasi *Metode Advanced Encryption Standard* (AES) Dan *MessageDigest* 5 (MD5) Pada Enkripsi Dokumen (Studi Kasus LPSE UNIB)". Penelitian tersebut menghasilkan sebuah aplikasi untuk mengamankan *file* dokumen. Yang membedakan penelitian tersebut dengan penelitian yang akan dibuat adalah pada penelitian yang dibuat *file* yang diamankan adalah *file musik*.

Penelitian yang dilakukan oleh Muhammad Rofiq pada tahun 2012 dengan judul "Perancangan dan Implementasi Algoritma Elgamal untuk Keamanan Data pada Video *Streaming*" menghasilkan sebuah aplikasi enkripsi dan dekripsi *file* video dengan algoritma elgamal pada perangkat *desktop*. Sedangkan untuk aplikasi yang akan di rancang dalam penelitian ini akan menggunakan algortima AES dan juga menggunakan algoritma Railfence untuk mengacak kunci yang di gunakan untuk proses enkripsi dan dekripsi *file musik*.

Bonifacius (2016) melakukan penelitian "Implementasi Sistem Keamanan File dengan Metode Steganografi EOF dan Enkripsi Caesar Cipher". Penelitian tersebut menghasilkan sebuah aplikasi untuk mengamankan sebuah *file* gambar. Yang membedakan adalah pada penelitian yang akan di laksanakan akan menghasilkan sebuah aplikasi untuk mengamankan sebuah *file musik*.

Nurliana (2015) melakukan penelitian “Aplikasi Keamanan File Algoritma Blowfishpada Universitas Lancang Kuning”. Penelitian tersebut menghasilkan aplikasi untuk mengamankan *file* dokumen menggunakan algoritma blowfish. Terdapat perbedaan dengan penelitian yang akan dilaksanakan, yaitu pada penelitian yang akan dilaksanakan akan menghasilkan aplikasi untuk mengamankan *file musik* dan juga menggunakan algoritma AES.

Nurhardian dkk pada tahun 2016 melakukan penelitian “Implementasi Keamanan File dengan Kompresi Huffman dan Kriptografi menggunakan Algoritma RC4 serta Steganografi menggunakan End of File Berbasis Desktop pada SMK Negeri 3 KotaTangerang”. Dari penelitian tersebut telah di hasilkan sebuah aplikasi untuk mengamankan *file* dokumen dengan menyisipkan *file* tersebut kedalam media audio. Yang menjadi perbedaan dengan penelitian yang akan di laksanakan adalah pada penelitian yang akan di laksanakan *file musik* akan di enkripsi langsung tanpa melakukan penyisipan.

Pada kesempatan kali ini saya akan melakukan penelitian “Penerapan Algoritma AES pada Keamanan File Musik dengan generator kunci menggunakan Algoritma Railfence”. Dari penelitian tersebut akan di hasilkan sebuah aplikasi berbasis desktop untuk mengamankan *file musik* yang bersifat rahasia menggunakan algoritma AES. Algoritma railfence di gunakan untuk menghasilkan kunci yang lebih acak dalam untuk di gunakan pada proses enkripsi dan dekripsi.

II.2. Uraian Teoritis

II.2.1. Penjelasan Algoritma

Algoritma adalah sistim kerja komputer memiliki *brainware*, *hardware*, dan *software*. Tanpa salah satu dari ketiga sistim tersebut, komputer tidak akan berguna. Kita akan lebih fokus pada *software* komputer. *Software* terbangun atas susunan program) dan *syntax* (cara penulisan/pembuatan program). Untuk menyusun program atau *syntax*, di perlukannya langkah-langkah yang sistematis dan logis untuk dapat menyelesaikan masalah atau tujuan dalam proses pembuatan suatu *software*. Maka, algoritma berperan penting dalam penyusunan program atau *syntax* tersebut.

Pengertian algoritma adalah susunan yang logis dan sistematis untuk memecahkan suatu masalah atau untuk mencapai tujuan tertentu. Dalam dunia komputer, algoritma sangat berperan penting dalam pembangunan suatu *software*. Dalam dunia sehari-hari, mungkin tanpa kita sadari algoritma telah masuk dalam kehidupan kita.

Algoritma berbeda dengan logaritma.Logaritma merupakan operasi matematika yang merupakan kebalikan dari eksponen atau pemangkatan. Contoh logaritma seperti bc= aditulis sebagai blog a = c (b disebut basis). (Maulana, Gun Gun ; 2017 : 70)

II.2.2.Kriptografi

Kriptografi (*Cryptography*) berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu kripto dan graphia. Kripto artinya menyembunyikan, sedangkan

graphia artinya tulisan. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasian data, keabsahan data, integritas data, serta autentikasi data. Tetapi tidak semua aspek keamanan informasi dapat di selesaikan dengan kriptografi.

Kriptografi dapat pula di artikan sebagai ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan di kirim dari suatu tempat ke tempat lain, isi pesan tersebut mungkin dapat di sadap oleh pihak lain yang tidak berhak untuk mengetahui isi pesan tersebut. Untuk menjaga pesan, maka pesan tersebut dapat di ubah menjadi sebuah kode yang tidak dapat dimengerti pihak lain.

Enkripsi adalah sebuah proses penyandian yang melakukan perubahan sebuah kode (pesan) dari yang bisa dimengerti (*plaintext*) menjadi sebuah kode yang tidak bisa dimengerti (*chipertext*). Sedangkan proses kebalikannya untuk mengubah *cipertext* menjadi *plaintext* disebut dekripsi. Proses enkripsi dan deskripsi memerlukan suatu mekanisme dan kunci tertentu. Kriptografi adalah ilmu mengenai teknik enkripsi di mana data di acak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi.Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi di lakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak di rahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma di anggap sesuatu yang tidak baik.Rahasia terletak di beberapa parameter yang di gunakan, jadi kunci di tentukan oleh parameter. (Amin, M. Miftakul ; 2016 : 130-131)

II.2.3. Algoritma *AdvancedEncryptionStandard* (AES)

Algoritma kriptografi bernama Rijndael yang didesain oleh Vincent Rijmen dan 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.

Algoritma Rijndael inilah yang kemudian dikenal dengan *Advanced Encryption Standard* (AES). Setelah mengalami beberapa proses standardisasi 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.

Pada algoritma AES, jumlah blok *input*, blok *output*, dan *state* adalah 128 *bit*. Dengan besar data 128 *bit*, berarti Nb = 4 yang menunjukkan panjang data 22 tiap baris adalah 4 *byte*. Dengan blok *input* atau blok data sebesar 128 *bit*, *key* yang digunakan pada algoritma AES tidak harus mempunyai besar yang sama dengan blok *input*.

Cipherkey pada algoritma AES bisa menggunakan kunci dengan panjang 128 *bit*, 192 *bit* atau 256 *bit*. Perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan di implementasikan pada algoritma AES ini. Tahapan enkripsi dan dekripsi AES, yaitu :

1. Enkripsi AES

Pada awal proses enkripsi, input yang telah dicopykan ke dalam *state* akan mengalami transformasi *byteAddRoundKey*. Setelah itu, state akan mengalami transformasi *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara

berulang-ulang sebanyak Nr. Proses ini dalam algoritma AES disebut sebagai *roundfunction*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya, dimana pada *round* terakhir *state* tidak mengalami transformasi *MixColumns*.

2. Dekripsi AES

Transformasi *chipher* dapat dibalikkan dan di implementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma AES. Transformasi *byte* yang digunakan pada *inverscipher* adalah *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundKey*. (Gilang ; 2016 : 280-281)

Terdapat 4 transformasi putaran pada proses enkripsi dan dekripsi :

1. *SubBytes*

   Berfungsi untuk menukar isi dari *byte* dengan menggunakan tabel substitusi.

2. *ShiftRows*

   Proses pergeseran blok per baris pada *statearray*.

3. *MixColumn*

   Proses mengalikan blok data (pengacakan) di masing-masing *statearray* dengan rumus sebagai berikut:

   $(x)=\{03\}x2+\{01\}x2+\{01\}x+\{02\}$…….(1)

4. *AddRoundKey*

   Mengombinasikan *statearray* dan *roundkey* dengan hubungan XOR.

Pada proses dekripsi algoritma AES:

1. *InvShiftRows*

Melakukan pergeseran *bit* ke kanan pada setiap blok baris.

2. *InvSubBytes*

   Setiap elemen pada *state* dipetakan dengan tabel *Inverse* S-*Box*.

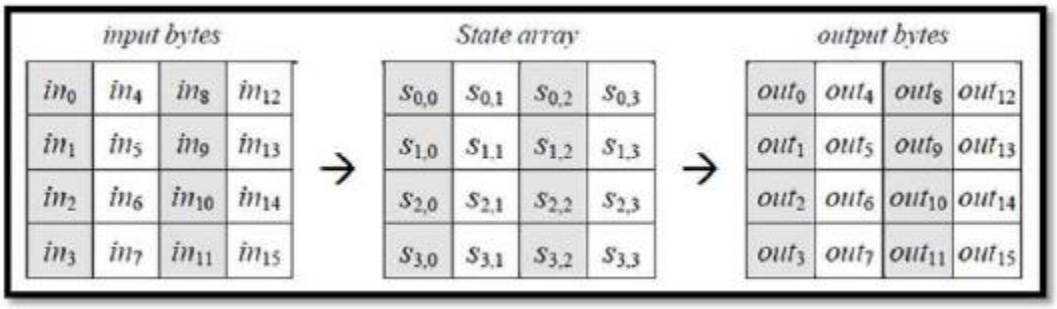
3. *InvMixColumn*

   Setiap kolom dalam *state* dikalikan dengan matriks AES.

4. *AddRoundKey*

   Mengombinasikan *statearray* dan *roundkey* dengan hubungan XOR.

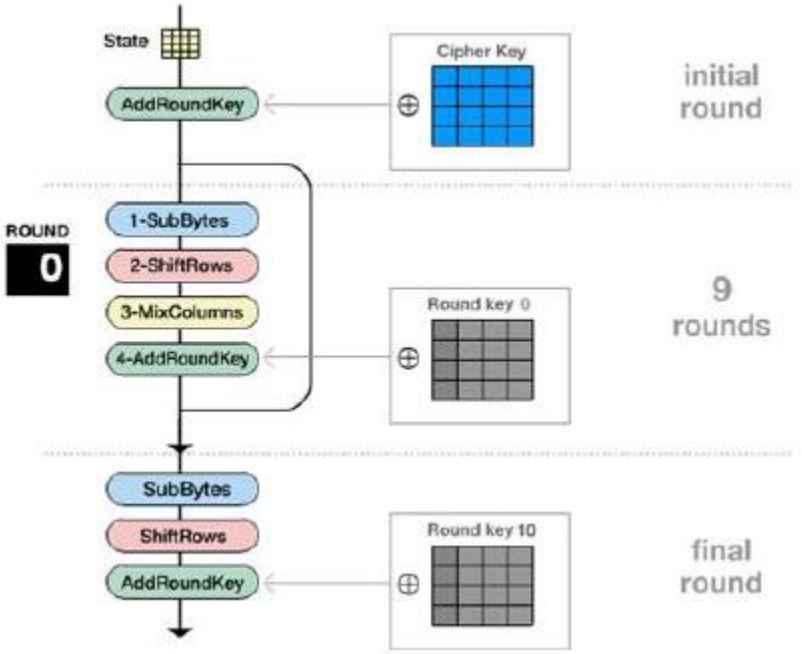
Penggambaran proses transformasi putaran dapat dilihat dari Gambar II.1 :



Gambar II.1. Proses Input Bytes, State Array, Output Bytes
(Sumber : Asri Prameshwari dan Nyoman Putra Sastra ; 2018)

Dari Gambar II.1 dapat dijelaskan bahwa algoritma AES ini ada dasarnya, algoritma AES ini merupakan *array of bytes* dengan dua dimensi yang disebut dengan *state*. Rumus ukuran dari *state* adalah $NROWSxNCOLS$, dari *state* ini akan diproses enkripsi dan dekripsi yang hasilnya akan dimasukkan ke dalam *arrayofstate*. Pada awal proses enkripsi data dimasukkan ke dalam *input bytes* yang kemudian akan di salin kedalam *arraystate*, pada proses ini nantinya akan dilakukan enkripsi dan dekripsi, hasil keluarannya akan di tampung dalam *output bytes*.

Pada awal proses enkripsi, *input* yang telah disalin ke dalam *state* akan mengalami transformasi *AddRoundKey*. Setelah itu *state* akan mengalami transformasi *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey* secara berulang-ulang sebanyak *round*/putaran (Nr). Proses ini dalam algoritma AES disebut sebagai *roundfunction*. *Round* yang terakhir, *state* tidak diberikan transformasi *MixColumns* . Ilustrasi proses awal enkripsi dengan menggunakan algoritma AES -128 dijelaskan pada Gambar II.2 :



Gambar II.2. Proses Enkripsi Dengan Menggunakan Algoritma AES-128 (Sumber : Asri Prameshwari dan Nyoman Putra Sastra ; 2018)

II.2.4. Pengertian *Musik*

Sebutan musik ini awal mulanya berasal dari bahasa Yunani yakni *mousikos*, yang mana sebutan tesebut diambil dari salah satu dewa Yunani. *Mousikos* disimbolkan sebagai suatu Dewa keindahan yang menguasai dalam

bidang keilmuan dan seni. Musik adalah waktu yang memang untuk di dengar.Musik merupakan wujud waktu yang hidup, yang merupakan kumpulan ilusi dan aluna suara. Alunan musik yang berisi rangkaian nada yang berjiwakan, mampu menggerakan hati para pendengarnya (Sylado, 1983). Musik berhubungan erat dengan waktu, bisa berupa cerita yang terjadi ataupun hal yang berusaha pencipta ungkapkan saat terjadinya sebuah peristiwa, bisa berupa perasaan, politik, maupun keadaan sosial. Hal ini berkesinambungan dengan rangkaian nada yang memiliki jiwa sehingga dapat membuat para penikmatnya mendengar alunan yang berusaha pencipta sampaikan. Musik adalah suatu hasil karya seni berupa bunyi dalam bentuk lagu atau komposisi yang mengungkapkan tentang pikiran dan perasaan penciptanya melalui unsur-unsur pokok musik yakni irama, melodi, harmoni, dan bentuk atau struktur lagu serta ekspresi sebagai sumber kesatuan (Jamalus, 1988).

II.2.5. Algoritma Railfence

Metode enkripsi Rail Fence adalah salah satu bentuk *cipher* transposisi yang sederhana yang diinspirasi dari model *Polybiussquare*. *Polybiussquare* adalah menyusun huruf sebagai matriks 5x5 dan mengkodekan huruf A sebagai 1-1, huruf B sebagai 1-2 dan seterusnya. Setiap karakter pada *Polybiussquare* diganti dengan indeks *cell* matriks tanpa menggunakan kunci khusus dan hanya merubah posisi sehingga teks tidak terbaca. Berbeda dengan *Polybiussquare*, metode Rail Fence menyusun teks secara *zig-zag* yang model matriksnya diketahui oleh pengirim dan penerima pesan.

Teknik Rail Fence adalah menuliskan *plaintext* dalam urutan diagonal dan membacanya 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 kedalaman (jumlah baris) 3, maka hasil perubahan susunannya dapat dilihat dalam gambar berikut :

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

Gambar II.2.5. Algoritma Railfence

Metode Rail Fence mudah untuk dibobol oleh kriptanalis dengan mencoba beberapa nilai kedalaman untuk menentukan banyaknya baris yang digunakan. Terdapat pola tertentu berdasarkan jumlah baris yang digunakan. Misal jika ada dua baris maka huruf ke 1,3,5,... akan berada di baris pertama dan huruf ke 2,4,6,... akan ada di baris kedua. Meskipun metode Rail Fence adalah metode yang lemah, namun metode ini dapat digabungkan dengan metode lain untuk meningkatkan keamanan *ciphertext* sehingga tidak mudah dipecahkan. (Retnani ; 2017 : 2-3)

II.2.6. Bahasa Pemrograman *VisualBasicNet*. 2010

MicrosoftVisualStudio adalah sebuah *Integrated Development Environment* buatan *Microsoft Coroporation*. *Microsoft Visual Studio* dapat di gunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang

berjalan di atas *Windows*) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas *.NET Framework*). Selain itu, *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. *Tollbuiltin* mencakup *form* desainer untuk membangun sebuah aplikasi GUI, *web* desainer, *class* desainer dan *database schema* desainer. *Microsoft Visual Studio* didukung bahasa pemrograman yang berbeda. Adapun bahasa pemrograman yang didukung oleh *Visual Basic Studio* adalah bahasa pemrograman C++, *Visual Basic*, *Visual* C#. *VisualStudio* juga dapat mendukung bahasa pemrograman lain seperti M, phyton dan ruby yang semuanya itu terdapat pada *packextra* yang terpisah dari *visual studio*. (Nency dkk ; 2016 : 205-206)

II.2.7. Pengertian UML

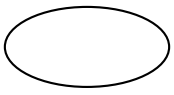
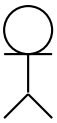
UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem

(Windu dkk ; 2013 : 81). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.2.7.1.*Use Case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuan (*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 didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. (Windu dkk ; 2013). Fungsi-fungsi tersebut dapat dilihat pada tabel Use Case Diagram dibawah ini :

Tabel II.1. Use Case Diagram

Gambar	Keterangan
	*UseCase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, 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 aktir, 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 *UseCase*, tetapi tidak memiliki kontrol terhadap *usecase*
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem

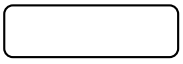
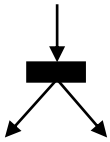
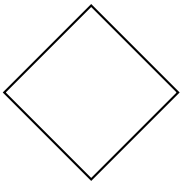
<<include>>	*Include*, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program
<<extends>>	*Extend*, merupakan perluasan dari *usecase* lain jika kondisi atau syarat

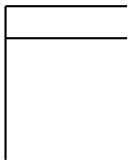
(Sumber : Ade Hendini ; 2016)

II.2.7.2.*Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dkk ; 2013 : BIT. 10. 80-87). Proses sebuah sistem dapat diperlihatkan pada tabel Activity Diagram dibawah ini :

Tabel II.2. Activity Diagram

Gambar	Keterangan
	*StartPoint*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*EndPoint*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*DecisionPoints*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*

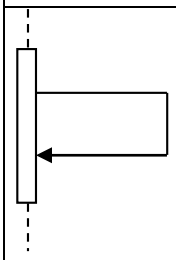
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

(Sumber : Ade Hendini ; 2016)

II.2.7.3.*Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dkk ; 2013). Untuk menggambarkan kelakuan obyek pada *use case* dapat dilihat pada diagram dibawah ini :

Tabel II.3. *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
	*BoundaryClass*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *formentry* dan *form* cetak
	*Controlclass*, 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*,mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini ; 2016)