

BAB II

TINJAUAN PUSTAKA

II.1 Penelitian Terkait

Adapun penelitian terkait yang akan digunakan sebagai sumber acuan yang relevan yaitu :

1. Penelitian yang dilakukan oleh Susanto, 2017, "Implementasi Keamanan Data Menggunakan Algoritma *Blowfish* Pada Sistem Informasi Koperasi RIAS". Menarik Kesimpulan Berdasarkan keseluruhan pengujian yang telah dilaksanakan, data yang di inputkan pada sistem informasi koperasi RIAS dan di simpan di dalam *database* berhasil di *enkripsi* sehingga tidak dapat dibaca dan dimengerti artinya. Sistem informasi koperasi RIAS juga telah berhasil mengembalikan data yang sebenarnya pada menu tampil data ataupun laporan, Sistem-sistem yang dibangun sebelumnya hanya menyimpan data pada *database* dengan bentuk sesuai dengan apa yang diinput, sehingga orang yang melihat isi *database* tersebut dapat membaca dan mengerti artinya.
2. Penelitian yang dilakukan oleh Muito, Anugrah Bagus Susilo, 2016, "Aplikasi Kriptografi *File* Menggunakan Metode *Blowfish* dan Metode *Base64* pada Dinas Kependudukan dan Pencatatan Sipil Kota Tangerang Selatan". Menarik kesimpulan Data dapat diamankan dengan Kriptografi Algoritma *Blowfish* dan *Base64*, Data tidak dapat dibuka oleh pihak yang tidak berhak yang tidak memiliki kunci untuk *enkripsi* dan *dekripsi file*, *File* yang diamankan menggunakan aplikasi ini tidak dapat dibuka oleh aplikasi lain, Program sistem keamanan dengan sistem Kriptografi Algoritma *Blowfish* dan *Base64* telah diuji coba, sehingga program dinyatakan sudah sesuai.
3. Penelitian yang dilakukan oleh Fahri H, dkk, 2022, "Pemanfaatan algoritma aes untuk keamanan data karyawan pt. telkom indonesia pematangsiantar". Menarik kesimpulan Dengan adanya system keamanan data karyawan pada PT.TELKOM Indonesia Pematang siantar dapat terbantu dalam mengamankan data-data yang bersifat rahasia.

Setelah dilakukan analisis maka dapat di lihat bahwa algoritma *AES* dapat mengamankan *file* dengan berbagai ekstensi, seperti : *doc*, *xls*, *ppt*, *pdf* dan juga *png*. Hasil dari data yang di enkripsi merupakan kumpulan kombinasi karakter yang tidak dapat dimengerti oleh manusia. Dengan menggunakan kunci yang sama maka hasil Enkripsi dan Dekripsi maka hasilnya akan selalu sama.

4. Penelitian yang dilakukan oleh Amrullah, Adi, 2015, “Penggunaan Algoritma *Aes-Rijndael* Pada Sistem Enkripsi Dan Dekripsi Untuk Komunikasi Data”. Menarik kesimpulan Proses enkripsi menggunakan algoritma AES Rijndael dengan kunci 128-bit berhasil dilakukan yang disimulasikan pada aplikasi AES Simulator. *Ciphertext* hasil dari proses enkripsi sebelumnya berhasil didekripsi dengan menggunakan kunci yang sama ketika proses enkripsi Output *Plaintext* hasil dekripsi sangat bergantung dengan kunci yang digunakan. Ketika *ciphertext* didekripsi menggunakan kunci yang berbeda dengan kunci yang digunakan ketika proses enkripsi, maka tidak akan didekripsi / Informasi tidak akan terbuka.

II.2. Aplikasi

Aplikasi berbasis komputer merupakan sebuah sistem pengolah data menjadi sebuah informasi yang berkualitas dan dipergunakan untuk suatu alat bantu pengambil keputusan. Aplikasi ini mengintegrasikan antara manusia dengan mesin yang memanfaatkan perangkat keras dan perangkat lunak komputer.(Renny Rahmadi Putra, dkk, 2016).

II.3. 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 kerahasiaan data, *integritas* data, serta *autentikasi* data. Tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan Kriptografi. (M. Miftakul amin, 2016 : 130).

II.4. Algoritma *Blowfish*

Blowfish merupakan Algoritma Kriptografi dengan penggunaan kunci pada blok *cipher* simetris (*symmetric block cipher*) yakni kunci yang digunakan pada proses *enkripsi* sama dengan kunci yang digunakan pada proses *dekripsi* dengan data masukan dan keluaran berupa blok-blok data berukuran 64 *bit*. *Blowfish* dirancang oleh *Bruce Schneier* pada tahun 1993 yang ditujukan untuk mikroprosesor besar (32 *bit* ke atas dengan *cache* data yang besar).

Blowfish dioptimasi untuk aplikasi dimana kunci tidak sering berubah dikarenakan *Blowfish* menggunakan *subkunci* yang besar. *Subkunci* ini harus dihitung sebelum proses *enkripsi* dan *dekripsi* data. Algoritma ini terdiri dari dua bagian yaitu *Key expansion* dan *data encryption*. *Key expansion* berfungsi merubah kunci yang besarnya dapat mencapai 448 *bit* menjadi beberapa *array subkunci* dengan total 4168 *byte*. *Data encryption* merupakan proses *enkripsi* yang terdiri dari iterasi beberapa operasi sederhana sebanyak 16 kali. Setiap iterasi terdiri dari permutasi dan substitusi antara bagian kunci dengan data. Seluruh proses menggunakan operasi penambahan dan *XOR* (*exclusive or*) pada variabel 32 *bit*. Tambahan operasi lainnya adalah empat penelusuran *table* (*table lookup*) untuk setiap putaran. (Dedi Dony, 2016 : 139).

II.4.1 Bagian-Bagian Algoritma *Blowfish*

Algoritma *Blowfish* terdiri atas dua bagian, yaitu ekspansi kunci dan *enkripsi* data

berikut (Mujito, Anugrah : 2016:55):

1. Ekspansi kunci (*Key-expansion*)

Berfungsi merubah kunci (minimum 32-bit, maksimum 448-bit) menjadi beberapa array subkunci (*subkey*) dengan total 4168 byte (18x32-bit untuk Parray dan 4x256x32-bit untuk S-box sehingga totalnya 33344 bit atau 4168 byte). Kunci disimpan dalam K-array:

$K_1, K_2, \dots, K_{j-1}, K_j, K_{j+1}, \dots, K_{14}$

Kunci-kunci ini yang dibangkitkan (*generate*) dengan menggunakan subkunci yang harus dihitung terlebih dahulu sebelum enkripsi atau dekripsi data. Sub-sub kunci yang digunakan terdiri dari.

P-array yang terdiri dari 18 buah 32-bit subkunci :

P_1, P_2, \dots, P_{18} .

S-box yang terdiri dari 4 buah 32-bit, masing-masing memiliki 256 entri :

$S_{1,0}, S_{1,1}, \dots, S_{1,255}$.

$S_{2,0}, S_{2,1}, \dots, S_{2,255}$.

$S_{3,0}, S_{3,1}, \dots, S_{3,255}$.

$S_{4,0}, S_{4,1}, \dots, S_{4,255}$.

Langkah-langkah perhitungan atau pembangkitan subkunci tersebut adalah sebagai berikut:

- a. Inisialisasi P-array yang pertama dan juga empat S-box, berurutan, dengan string yang telah pasti. String tersebut terdiri dari digit-digit heksadesimal dari phi, tidak termasuk angka tiga di awal.

Contoh :

$P_1 = 0x243f6a88$.

$P_2 = 0x85a308d3$.

$P_3 = 0x13198a2e$.

P4= 0x03707344 dan seterusnya sampai S-box yang terakhir (daftar heksadesimal digit dari phi untuk P-array dan Sbox bisa lihat Lampiran).

- b. *XOR*-kan P1 dengan 32-bit awal kunci, *XOR*-kan P2 dengan 32-bit berikutnya dari kunci, dan seterusnya untuk semua *bit* kunci. Ulangi siklus seluruh *bit* kunci secara berurutan sampai seluruh P-array ter-*XOR* kan dengan *bit-bit* kunci. Atau jika disimbolkan : $P1 = P1 _ K1$, $P2 = P2 _ K2$, $P3 = P3 _ K3$, \dots $P14 = P14 _ K14$, $P15 = P15 _ K1$, \dots $P18 = P18 _ K4$. Keterangan : $_$ adalah simbol untuk *XOR*. 18
- c. Enkripsikan *string* yang seluruhnya nol (*all-zero string*) dengan Algoritma *Blowfish*, menggunakan *subkunci* yang telah dideskripsikan pada langkah 1 dan 2.
- d. Gantikan P1 dan P2 dengan keluaran dari langkah 3.
- e. Enkripsikan keluaran langkah 3 menggunakan Algoritma *Blowfish* dengan *subkunci* yang telah dimodifikasi.
- f. Gantikan P3 dan P4 dengan keluaran dari langkah 5.
- g. Lanjutkan langkah-langkah di atas, gantikan seluruh elemen P-array dan kemudian keempat S-box secara berurutan, dengan hasil keluaran Algoritma *Blowfish* yang terus-menerus berubah. Total keseluruhan, terdapat 521 iterasi untuk menghasilkan subkunci subkunci dan membutuhkan memori sebesar 4KB.

2. Enkripsi Data

Terdiri dari iterasi fungsi sederhana (*Feistel Network*) sebanyak 16 kali putaran (iterasi), masukannya adalah 64-bit elemen data X. Setiap putaran terdiri dari permutasi kunci *dependent*, substitusi kunci, dan data *dependent*. Semua operasi adalah penambahan (*addition*) dan *XOR* pada variabel 32-bit. Operasi tambahan lainnya hanyalah empat penelusuran tabel *array* berindeks untuk setiap putaran. Langkahnya adalah seperti berikut.

- a. Bagi X menjadi dua bagian yang masing-masing terdiri dari 32-bit: XL, XR.

b. Lakukan langkah berikut.

For $i = 1$ to 16:

$XL = XL _ P_i$

$XR = F(XL) _ XR$.

Tukar XL dan XR .

c. Setelah iterasi ke-16, tukar XL dan XR lagi untuk melakukan membatalkan pertukaran terakhir.

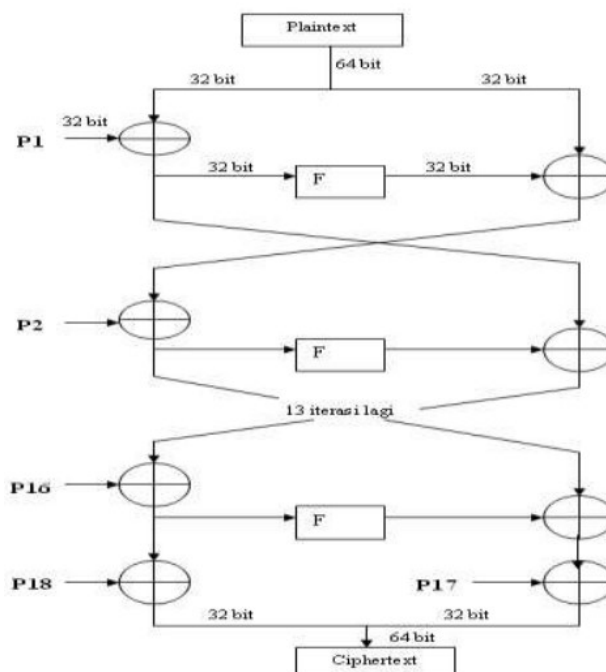
d. Lalu lakukan

$XR = XR _ P_{17}$.

$XL = XL _ P_{18}$.

e. Terakhir, gabungkan kembali XL dan XR untuk mendapatkan *ciphertexts*.

Untuk lebih jelasnya, gambaran tahapan pada jaringan *feistel* yang digunakan *Blowfish* adalah seperti pada Gambar II.1.

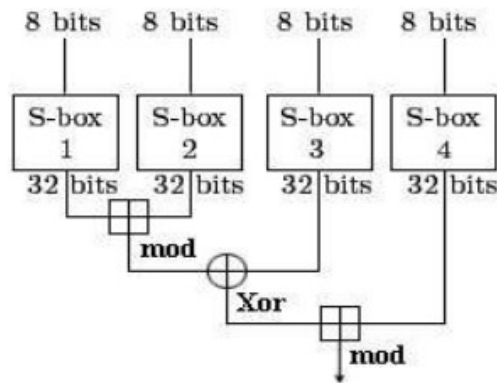


Gambar II.1. Diagram Enkripsi Algoritma Blowfish

Pada langkah kedua, telah dituliskan mengenai penggunaan fungsi F . Fungsi F adalah:

bagi XL menjadi empat bagian 8-bit: a,b,c dan d. $F(XL) = ((S1,a + S2,b \bmod 232) \text{ XOR } S3,c) + S4,d \bmod 232 \dots\dots\dots(2.1)$

Agar dapat lebih memahami fungsi F, tahapannya dapat dilihat pada Gambar II.2.



Gambar II.2. Fungsi F Dalam Blowfish

Dekripsi sama persis dengan *enkripsi*, kecuali bahwa $P1, P2, \dots, P18$ digunakan pada urutan yang berbalik (*reverse*). Algoritmanya dapat dinyatakan sebagai berikut :

For $i = 1$ to 16 do

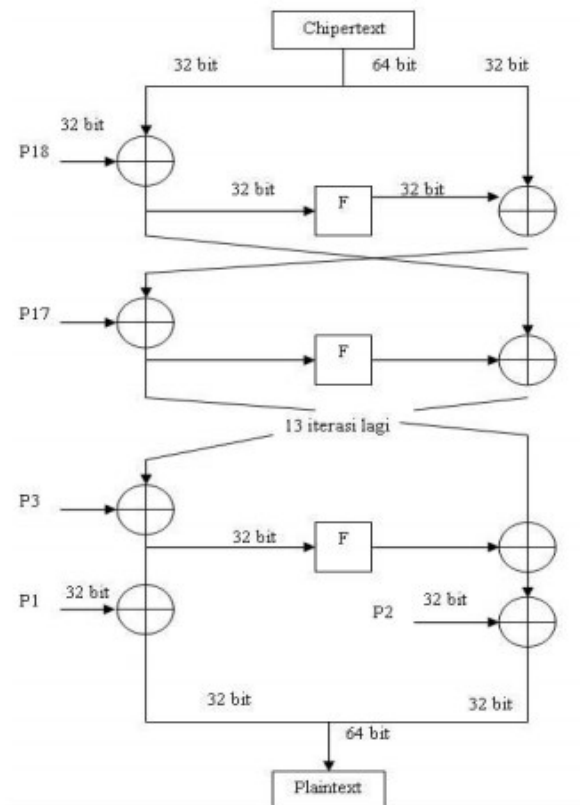
$XR_i = XL_{i-1} _ P_{19-i};$

$XL_i = F[XR_i] _ XR_{i-1};$

$21 \ XL_{17} = XR_{16} _ P_1;$

$XR_{17} = XL_{16} _ P_2;$

Blok diagram *dekripsi* seperti pada Gambar II.3 berikut :



Gambar II.3. Blok Diagram Dekripsi Blowfish

II.5. Algoritma Rijndael

Algoritma Rijndael menggunakan substitusi, permutasi dan sejumlah putaran yang dikenakan pada tiap blok yang akan dienkripsi dan dekripsi. Untuk setiap putarannya, Rijndael menggunakan kunci yang berbeda. Kunci setiap putaran disebut round key. Rijndael beroperasi dalam orientasi byte sehingga memungkinkan untuk implementasi algoritma yang efisien ke dalam software dan hardware. Ukuran blok untuk algoritma Rijndael adalah 128 bit (16 byte). Algoritma Rijndael ditetapkan sebagai AES (Advanced Encryption Standard) oleh NIST (National Institute of Standard and Technology) tahun 2001 untuk menggantikan DES (Data Encryption Standard) yang sudah berakhir masa penggunaannya sebagai standard enkripsi kriptografi simetri. DES dianggap sudah tidak aman lagi karena dengan perangkat keras khusus kuncinya bisa ditemukan dalam beberapa hari. NIST kemudian mengadakan sayembara terbuka untuk membuat standard algoritma kriptografi yang baru sebagai

pengganti DES. Standar tersebut nanti akan diberi nama Advanced Encryption Standard (AES). AES mendukung panjang kunci 128 bit sampai 256 bit dengan step 32 bit. Panjang kunci dan ukuran blok dapat dipilih secara independen. Setiap blok dienkripsi dalam sejumlah putaran tertentu. Karena AES menetapkan panjang kunci adalah 128, 192 dan 256, maka dikenal AES-128, AES-192 dan AES-256. (Amrullah, 2015 : 32).

Tabel 1. Jumlah Proses Berdasarkan Bit Block Dan Kunci

	(NK)	(NB)	(NR)
AES - 128	4	4	10
AES - 192	6	4	12
AES - 256	8	4	14

Algoritma Rijndael mempunyai 3 parameter:

1. Plaintext adalah array yang berukuran 16 byte, yang berisi data masukan.
2. Ciphertext adalah array yang berukuran 16 byte, yang berisi hasil enkripsi.
3. Key adalah array yang berukuran 16 byte, yang berisi kunci cipher (disebut juga cipher key).

Garis besar algoritma Rijndael yang beroperasi pada blok 128 bit dengan kunci 128 bit adalah sebagai berikut:

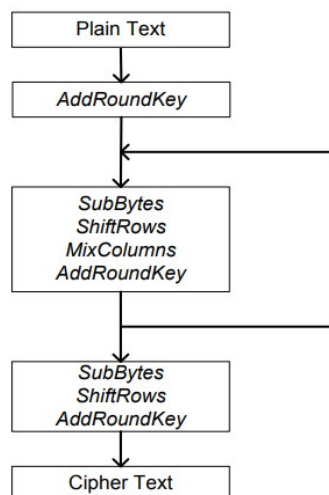
1. AddRoundKey, melakukan XOR antara state awal (plaintext) dengan cipher key. Tahap ini disebut juga initial round.
2. Putaran sebanyak Nr-1 kali. Proses yang dilakukan pada setiap putaran adalah:
 - a. SubBytes adalah substitusi byte dengan menggunakan tabel substitusi (S-Box).

- b. ShiftRows adalah pergeseran baris-baris array state secara wrapping.
- c. MixColumns adalah mengacak data di masing-masing kolom array state.
- d. AddRoundKey adalah melakukan XOR antara state sekarang dengan round key

3. Final round, proses untuk putaran terakhir:

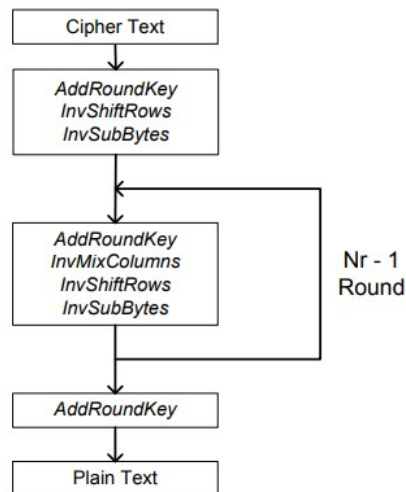
- a. SubBytes
- b. ShiftRows
- c. AddRoundKey

Garis besar Enkripsi Algoritma Rijndael seperti gambar II.4:



Gambar II.4. Algoritma Enkripsi Rijndael

Garis besar Dekripsi Algoritma Rijndael seperti gambar II.5:



Gambar II.5. Algoritma Dekripsi Rijndael

II.6. Konversi

Konversi merupakan mengubah sebuah nilai satuan ke satuan lainnya, dan tidak pernah mengubah suatu nilai besaran. Dari hal tersebut dapat dilakukan dengan mengkonversi satuan yang sama ataupun berbeda. (Karlo, 2019)

$$010101011111 (2) = 2537 (8)$$

1. Konversi Dari Bilangan Desimal

Yaitu dengan cara membagi bilangan desimal dengan 8 kemudian diambil sisa pembagiannya

Contoh:

$$385 (10) = \dots (8)$$

$$385 : 8 = 48 + \text{sisa } 1$$

$$48 : 8 = 6 + \text{sisa } 0$$

$$601 (8)$$

2. Konversi Bilangan Desimal Ke Oktal

Yaitu dengan cara membagi bilangan desimal dengan 16 kemudian diambil sisa pembagiannya

Contoh:

$$1583 (10) = \dots(16)$$

$$1583 : 16 = 98 + \text{sisanya}$$

$$96 : 16 = 6 + \text{sisanya}$$

$$62F (16)$$

3. Konversi Bilangan Desimal Ke Heksadesimal

Yaitu dengan cara membagi bilangan desimal dengan 16 kemudian diambil sisanya

Contoh :

$$1583 (10) = \dots(16)$$

$$1583 : 16 = 98 + \text{sisanya}$$

$$96 : 16 = 6 + \text{sisanya}$$

$$62F (16)$$

II.7 Sistem Bilangan Binner

Sistem bilangan biner berbasis angka 2, maksudnya bahwa penulisan bilangan hanya menggunakan digit biner (bit) yaitu 0 dan 1. Istilah bit dipakai dalam sistem bilangan biner yang merupakan singkatan dari binary digit. Byte adalah string yang terdiri dari 8 bit. Analoginya seperti sistem desimal yang didasarkan pada bilangan 10 dimana penulisan suatu bilangan hanya digit desimal 10 yaitu 0 sampai 9. Sistem biner dan desimal keduanya merupakan sistem notasi posisional, dimana nilai dalam sistem lainnya tergantung pada penempatannya dengan sebuah bilangan. Bilangan desimal 845, digit 4 menandakan 40; bilangan 9426, digit 4 menandakan 400 ($845 = 800 + 40 + 5$; $9426 = 9000 + 400 + 20 + 6$). Nilai digit ditentukan oleh bilangan digit apa dan dimana letaknya. Sistem desimal, digit dalam posisinya berada di kiri nilai desimal yang dikalikan dengan 1 (100). Posisi dua digit ke kiri nilai desimal dikalikan dengan 10 (101). Digit di kiri posisi berikutnya dikalikan dengan 100 (102). Pengali posisional sebagaimana kiri nilai desimal dikalikan 10 pangkat

angka berurut naik. Ide yang sama digunakan dalam sistem biner, kecuali pengali posisional merupakan 2 pangkat n ($2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$,). Misalkan, bilangan biner 101 mempunyai persamaan desimal: $2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = 4 + 0 + 1 = 5$. (Wahyu, 2017)

syarat : harus dibagi dengan nilai 2, setiap pembagian merupakan digit binary dari bilangan binary hasil konversi.

Contoh :

$$45_{10} = 101101_2$$

2	45	Sisa 1	→ LBS
2	22	Sisa 0	
2	11	Sisa 1	
2	5	Sisa 1	
2	2	Sisa 0	
	1	→ MBS	10010

II.8. Vb.net 2015

Merupakan IDE (*integrated development environment*) yang paling banyak digunakan khususnya untuk mengembangkan aplikasi *mobile*, *desktop*, maupun *website*. IDE adalah kerangka desain di mana anda dapat membangun aplikasi dengan Visual yang mudah dipahami. Versi terbaru dari *Microsoft Visual BASIC 2015 RTM* mencakup banyak fitur untuk membuat aplikasi besar, pengembangan *mobile cross-platform* untuk *iOS*, *Android*, dan *Windows*, termasuk *Xamarin*, *Apache Cordova*, *Unity*, dan banyak lagi. Adapun fitur terbaru dari *Visual Studio 2015 RTM*, *Visual Studio C++ for Cross-Platform Development*, *Visual Studio Tools for Apache Cordova*, *Visual Studio Emulator for Android*, *Visual Studio*

Tools for Universal Windows App Development, Visual Studio Tools for Universal Windows App Development, Visual C++, C# and Visual Basic, F#, NET Framework 4.6, Entity Framework, Visual Studio IDE, Debugging and Diagnostics, ASP.NET, Entity Framework, Azure, NuGet, JavaScript , TypeScript, IntelliTest, Application Insights, Release Management, Architecture, Design, and Modeling, XAML Language Service dan Miscellaneous.(Muhammad Wali : 2017 : 8).

II.9. Unified Modelling Language (UML)

Menurut Ade Hendini (2016) *Unified Modeling Language (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.

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

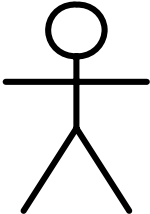

1. Use Case Diagram

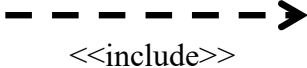
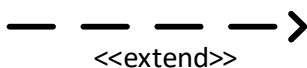
Use case diagram merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *Use Case Diagram* yaitu:

Tabel II.2

Simbol *Use Case Diagram*

Gambar	Keterangan
--------	------------

	<p><i>Use Case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktif yang dinyatakan dengan menggunakan kata kerja.</p>
	<p><i>Actor</i> atau Aktor adalah <i>Abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi <i>actor</i>, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bias muncul dalam beberapa peran. Perlu dicatat bahwa <i>actor</i> berinteraksi dengan <i>Use Case</i>, tetapi tidak memiliki control terhadap <i>usecase</i>.</p>
	<p>sosiasi antara aktor dan <i>usecase</i>, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang memintainteraksi secara langsung dan bukannya mengindikasikan data.</p>
	<p>sosiasi antara aktor dan <i>usecase</i> yang menggunakan panah terbuka untuk</p>

	mengindikasikan bila <i>actor</i> berinteraksi secara pasif dengan sistem.
	<i>include</i> , merupakan didalam <i>use case</i> lain (<i>required</i>) atau pemanggilan <i>usecase</i> oleh <i>usecase</i> lain, contohnya adalah pemanggilan sebuah fungsi program.
	<i>extend</i> , merupakan perluasan dari <i>use case</i> lain jika kondisi atau syarat terpenuhi.




(Sumber : Ade Hendini, 2016)

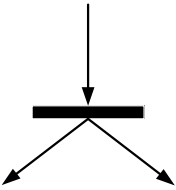
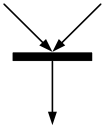
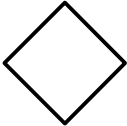
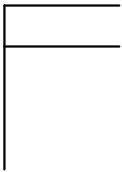
2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis simbol-simbol yang digunakan dalam *activity Diagram* yaitu:

Tabel II.3

Simbol *Activity Diagram*

Gambar	Keterangan
	<i>Start Point</i> , diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
	<i>End Point</i> , akhir aktivitas.
	<i>Activities</i> , menggambarkan suatu proses/kegiatan bisnis.

	<p>ork/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.</p>
	<p>oin (penggabungan) atau <i>rake</i>, digunakan untuk menunjukkan adanya dekomposisi.</p>
	<p>Decision Points, menggambarkan pilihan untuk pengambilan keputusan, <i>true</i> atau <i>false</i>.</p>
	<p>swimlane, pembagian <i>activity diagram</i> untuk menunjukkan siapa melakukan apa.</p>

(Sumber : Ade Hendini, 2016)

3. Diagram Urutan (*Sequence Diagram*)

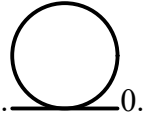
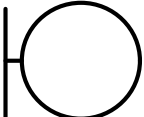
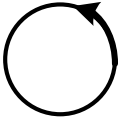


Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek.


Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu:

Tabel II.4

Simbol *Sequence Diagram*

Gambar	Keterangan
--------	------------

	<p><i>Entity Class</i>, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.</p>
	<p><i>Boundary Class</i>, berisi kumpulan kelas yang menjadi <i>interfaces</i> atau interaksi Antara satu atau lebih aktor dengan sistem, seperti tampilan <i>formentry</i> dan <i>form</i> cetak.</p>
	<p><i>Control class</i>, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.</p>
	<p><i>Message</i>, simbol mengirim pesan antar <i>class</i>.</p>
	<p><i>Recursive</i>, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.</p>
	<p><i>Decision Points</i>, menggambarkan pilihan untuk pengambilan keputusan, <i>true</i> atau <i>false</i>.</p>

	<i>lifeline</i> , garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i> .
---	---

(Sumber : Ade Hendini, 2016)

4. Diagram Kelas (*Class Diagram*)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung 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*, attribute (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek *eksternal* kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality*.

Tabel II.5

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 maksimal 4
------	---

(Sumber : Ade Hendini, 2016)