BAB II
TINJAUAN PUSTAKA

Pada bab ini akan membahas tentang teori dan komponen penunjang yang akan digunakan pada perancangan dan implementasi dari sistem keamanan smarthome menggunakan *fingerprint* dan MD5. Pembahasan mencakup pengertian *MD5, Fingerprint, Doorlock, NodeMCU,* dan *PHP Mysql*.

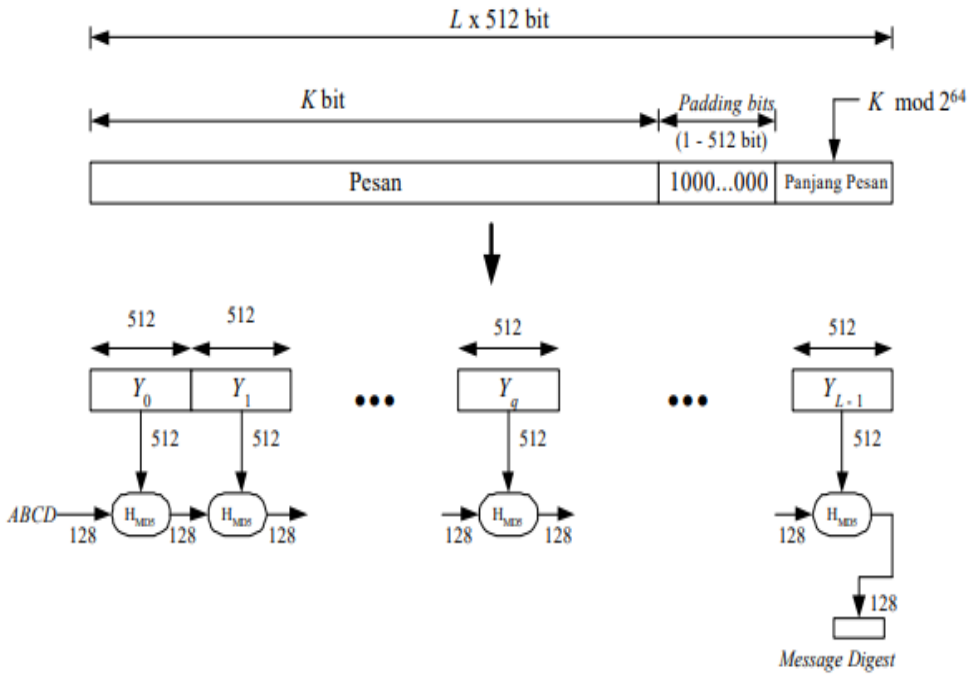
2.1 MD 5 (Message Digest Algorithm 5)

Algoritma Message Digest 5 merupakan algoritma yang dirancang oleh Ron Rivest dimana algoritma ini dipublikasi pada tahun 1992. Masukan dari MD5 adalah pesan yang memiliki panjang data sembarang, namun MD5 menghasilkan keluaran berupa *hash value* atau *message digest* yang memiliki panjang data tetap yaitu 128 bit[5].

MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan, dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah berkas[6]. *Enkripsi* menggunakan MD5 masih mendominasi sebagian besar aplikasi *PHP*. *Enkripsi* MD5 dianggap kuat karena *enkripsi* yang dihasilkannya bersifat *one way hash*. Berapapun string yang di *enkripsi* hasilnya tetap sepanjang 32 karakter. Untuk mengerjakan algoritma ini membutuhkan perhitungan yang tidak mudah, dimana banyak pesan diproses hingga menghasilkan *hash value* yang spesifik untuk tiap pesannya. Algoritma MD5 didesain untuk mesin 32-bit.

2.1.1 Cara Kerja

Sebelum melakukan proses MD5 adalah mengubah data menjadi data bit. Setelah proses awal ini selesai, maka MD5 memproses data menjadi blok-blok data yang terdiri dari 512 bit blok, dan 512 bit ini dibagi ke dalam 16 sub-blok yang terdiri dari 32 bit. Hasil proses dari blok-blok ini akan menghasilkan hash value 128 bit[7].



Gambar 2.1 Gambaran umum fungsi *hash* MD5

Pada gambar 2.1 dijelaskan fungsi *hash* MD5 ini menerima pesan yang ukurannya *K* bit selanjutnya pesan ditambah dengan *padding bits* sepanjang 1 sampai 512 bit. Kemudian disambung lagi dengan bit yang menyatakan panjang pesan, jika panjang pesannya lebih dari 2^{64} bit maka disebutkan *K* mod 2^{64}. Selanjutnya pesan yang telah disambung akan menjadi sejumlah blok yang setiap blok berukuran 512-bit. Pada Blok Y_0,Y_1 dan Y_{L-1} akan masuk pada proses H_{MD5} dimana akan menerima 2 buah input yaitu blog itu sendiri kemudian sebuah *buffer* yang bernama A, B, C, D yang panjangnya 128 bit hasil dari proses H_{MD5}. Hasil tersebut menghasilkan luaran 128 bit masuk kembali pada proses H_{MD5} berikutnya yang memproses blok. Terakhir dihasilkan keluaran sepanjang 128 bit, inilah yang disebut dengan *messagest digest* dari pesan tersebut. Jadi semua proses akan saling berkaitan dari H_{MD5} ini tergantung kepada blok dan nilai dari H_{MD5} sebelumnya. Sebuah blog akan mengalami perubahan jika keluaran dari MD5 ini berubah nilainya dan nilai tersebut akan merambat ke operasi berikut pada H_{MD5} sehingga hasilnya akan berbeda dari pesan sebelumnya.

Terdapat 5 langkah yang dibutuhkan untuk pembuatan *message digest*. Langkah tersebut akan dijelaskan sebagai berikut:

1. Penambahan bit – bit penganjal

Pesan akan ditambahkan bit – bit tambahan sehingga panjang bit akan kongruen dengan 448, mod 512. Hal ini berarti pesan akan mempunyai panjang yang hanya kurang 64 bit dari kelipatan 512 bit. Penambahan bit selalu dilakukan walaupun panjang dari pesan sudah kongruen dengan 448, mod 512 bit[5].

2. Penambahan nilai panjang pesan

Setelah penambahan bit, pesan masih membutuhkan 64 bit agar kongruen dengan kelipatan 512 bit. 64 bit tersebut merupakan perwakilan dari b (panjang pesan sebelum penambahan bit dilakukan). Bit – bit ini ditambahkan ke dalam dua word (32 bit) dan ditambahkan dengan *low-order* terlebih dahulu. Penambahan pesan ini bisa disebut juga ***MD Strengthening*** atau ***Penguatan MD*** [5].

3. Inialisasi MD5

Pada MD5 terdapat empat buah word 32 bit register yang berguna untuk menginisialisasi *message digest* pertama kali. Register – register ini di inisialisasikan dengan bilang hexadesimal. Buffer empat kata (A, B, C, D) digunakan untuk menghitung intisari pesan. Di sini masing-masing A, B, C, D adalah register 32 bit. Register ini diinisialisasi ke nilai berikut dalam heksadesimal sebagai berikut:

Word A: 01234567

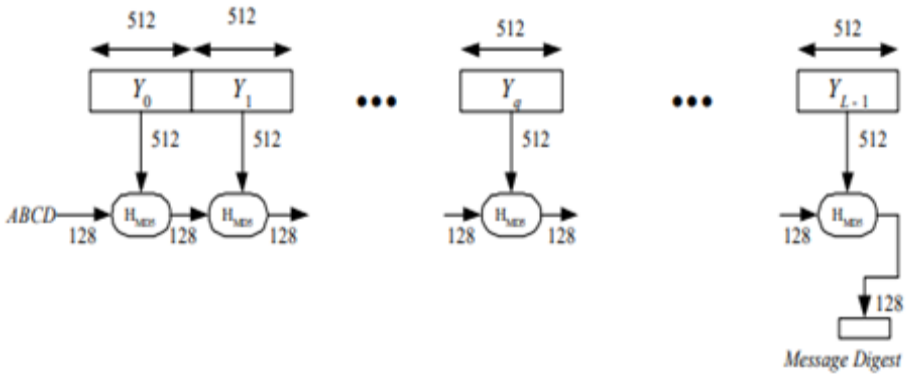
Word B: 89ABCDEF

Word C: FEDCBA98

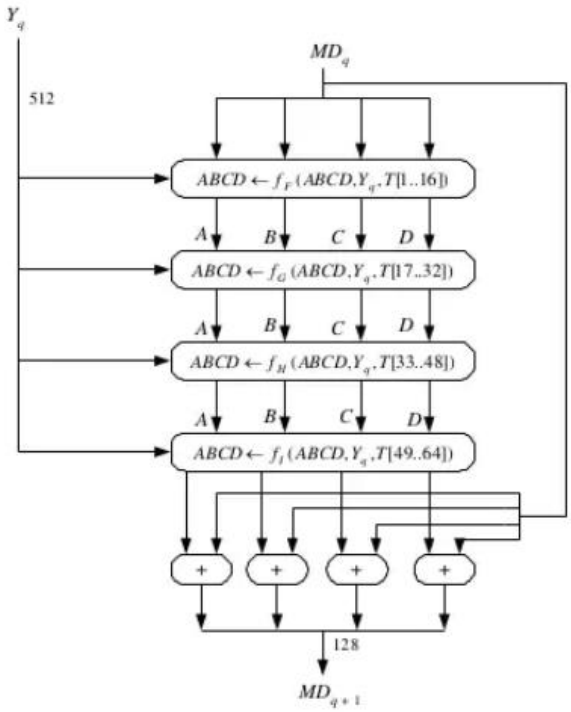
Word D: 76543210

4. Pengolahan pesan dalam blok berukuran 512 bit

Pesan akan dibagi menjadi L buah blok yang masing – masing dimana Y_0 sampai dengan Y_{L-1} panjanganya 512 bit yang terlihat pada gambar 2.2. Setiap blok akan diproses denganya penyangga MD menjadi 128 bit dan disebut dengan proses $H_{MD5.}$

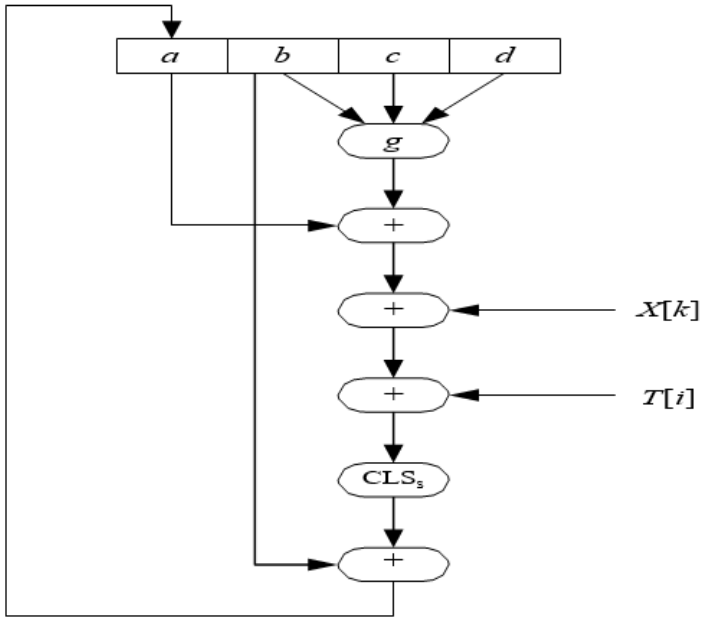


Gambar 2.2 Pengolahan Pesan Dalam Blok 512 Bit



Gambar 2.3 Proses H_{MD5}

Pada gambar 2.3 merupakan proses H_{MD5} n bahwa proses H_{MD5} terdiri dari 4 buah putaran, dan masing-masing putaran melakukan opersi dasar MD5 sebanyak 16 kali *Yq* menyatakan blok 512-bit ke-*q* dari pesan yang ditambah bit pengganjal dan tambahan 64 bit nilai panjang pesan semula. MD_q adalah nilai *message digest* 128-bit dari proses H_{MD5} ke-*q* [7]. Pada awal proses, MD_q berisi nilai inisialisasi penyangga MD. Fungsi-fungsi f_F, f_G, f_H, dan f_I masing-masing berisi 16 kali operasi dasar terhadap masukan. Operasi dasar MD5 diperlihatkan pada Gambar 2.4 berikut:



Gambar 2.4 Operasi dasar MD5

Operasi dasar MD5 yang diperlihatkan gambar 2.5 dapat dituliskan dengan persamaan (2.1) berikut ini:

$$a \leftarrow b + CLS_s(a + g(b,c,d) + X[k] + T[i]) \quad (2.1)$$

Keterangan:

a, b, c, d	:	Empat buah peubah Penyangga A, B, C, D
g	:	Salah satu fungsi F,G,H,I
CLSs	:	Circular left shift sebanyak *s* bit
X[K]	:	Kelompok 32-Bit ke-K dari blok 512 bit message ke-q. Nilai k : 0 sampai 15
T[i]	:	Elemen tabel T ke-i (32-bit)
+	:	Operasi penjumlahan module

Fungsi f_F, f_G, f_H, dan f_I adalah fungsi untuk memanipulasi masukan *a*, *b*, *c*, dan *d* dengan ukuran 32-bit [7]. Masing-masing fungsi dapat dilihat pada Tabel 2.1 berikut:

Tabel 2.1 Fungsi Dasar MD5

Nama	**Notasi**	**g(a, b, c, d)**
f_F	F(b, c, d)	(b ∧ c) ∨ (~b ∧ d)
F_G	G(b, c, d)	(b ∧ d) ∨ (c ∧ ~d)
f_H	H(b, c, d)	b ⊕ c ⊕ d
f_I	I(b, c, d)	c ⊕ (b ∧ ~ d)

Keterangan :

∧ : AND

∨ : OR

~ : NOT

⊕ : XOR

Dari operasi dasar MD5 dapat dilihat bahwa masing- masing fungsi f_F, f_G, f_H, dan f_I melakukan 16 kali operasi dasar. Bila Mj menyatakan pesan sub-blok ke-j (l bernilai mulai 0 sampai dengan 15), dan <<<s menyatakan pergeseran ke kiri sebanyak s bit, 4 operasi yang dijalankan yaitu:

$$ff(a,b,c,d,Mj,s,ti) menyatakan\ a = b + ((a + f(b,c,d) + Mj + Ti) \lll s$$

$$gg(a,b,c,d,Mj,s,ti) menyatakan\ a = b + ((a + f(b,c,d) + Mj + Ti) \lll s$$

$$hh(a,b,c,d,Mj,s,ti)\ menyatakan\ a = b + ((a + f(b,c,d) + Mj + Ti) \lll s$$

$$ii(a,b,c,d,Mj,s,ti)\ menyatakan\ a = b + ((a + f(b,c,d) + Mj + Ti) \lll s$$

2.2 Sensor Fingerprint

Penggunaan sensor *fingerprint* mencakup proses perekaman data sidik jari maupun proses pencocokan data sidik jari mengunakan *NodeMCU*. Modul sensor *fingerprint* yang akan di gunakan yaitu modul sensor *Z70* [8].

Fingerprint menerapkan sensor *scanning* untuk mengetahui sidik jari seseorang guna keperluan verifikasi identitas. Sensor *Fingerprint* seperti digunakan pada beberapa peralatan elektronik seperti *smartphone*, pintu masuk, alat absensi karyawan dan berbagai macam peralatan elektronik yang membutuhkan tingkat keamanan yang tinggi, dan hanya bisa di akses oleh orang-orang tertentu saja. Sebelum sensor *Fingerprint* ditemukan, dahulu sebuah data diamankan dengan menggunakan password atau ID, ada juga yang menggunakan pola guna mengamankan suatu data. Bentuk dari modul *fingerprint* ini ditunjukkan pada gambar 2.5 berikut ini:

Gambar 2.5 Bentuk Fisik Modul sensor *fingerprint Z70*

Modul ini memiliki 6 buah pin yaitu *Vtouch, Sout, Vin, TD, RD,* dan *GND* [8]. Deskripsi setiap pin ada di Tabel 2.2 di bawah ini:

Tabel 2.2 Fungsi pin Z70

Nama	Tipe	Fungsi
Vtouch	*In*	*Touch induction* power Input (kabel biru)
Sout	*out*	*Induction signal* output (kabel kuning)
Vin	*In*	Power input (kabel merah)
TD	*out*	Data output : *TTL logic* level (kabel hijau)
RD	*In*	Data input : *TTL logic* level (kabel putih)
GND		*Ground*

Modul Z70 adalah sebuah modul sensor sidik jari (*fingerprint*). Modul ini membutuhkan tegangan antara 3,8V, -7V *DC* agar dapat bekerja. Komunikasi untuk pengambilan data yaitu komunikasi serial [8]. Parameter utama ada pada tabel 2.3 berikut:

Tabel 2.3 Parameter modul *fingerprint* Z70

No	**Parameter**	**Keterangan**
1	Tegangan Operasi	3,8V - 7V DC
2	Arus operasi	Typical:<65mA *Peak*: <95Ma
3	Baud rate	(9600*N)bps

		N=1 ~ 12 (default N=6)
4	Waktu pengambilan gambar	<1s
5	Kapasitas penyimpanan	1000
6	*FAR*	<0.001%
7	Rata-rata waktu pencarian	<1s(1:500)
8	*Working environment*	-20^{0}C s/d +60ºC
		RH: 40% - 85%
9	Antarmuka	UART (TTL logic level)
10	Mode pencocokan	1:1 dan 1:N
11	Ukuran karakter	256 bytes
12	Ukuran template	512 bytes
13	Level keamanan	5(1,2,3,4,5(highest))
14	*FRR*	<1%
15	Ukuran kaca penampang	14,5mm*19,4mm
16	*Storage environment*	-40^{0}C s/d +85^{0}C
		RH: <85%+A2:CA3:C21

Pemroresan sidik jari mencakup dua bagian yaitu *fingerprint enrollment* (perekaman) dan *fingerprint matching* (pengenalan sidik jari). Ketika proses *enrollment* dilakukan penggunaan harus menempatkan sidik jari ke sensor sebanyak dua kali. Sistem ini akan memproses gambar jari, menghasilkan sebuah template terkait jari yang direkam dan menyimpannya [8].

Kekurangan mesin *fingerprint* adalah pada kondisi tertentu mesin ini dapat mengalami error atau lambat dalam mendeteksi sidik jari. Data yang disimpan dalam mesin berasal dari scan sidik jari yang dilakukan pada saat proses registrasi. Bila jari yang terdeteksi berada dalam kondisi basah, berkeringat atau kotor maka proses pengenalan dapat sulit dilakukan. Namun ada beberapa kelebihan menggunakan sensor *fingerprint* yaitu lebih cepat, praktis, data lebih akurat dan kapasitas besar.

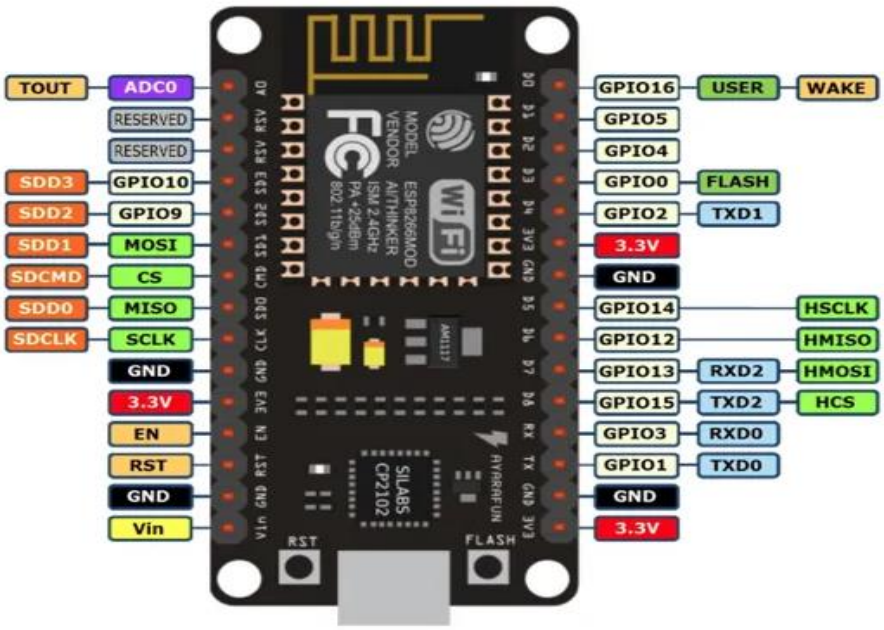
2.3 *NodeMCU ESP8622*

NodeMCU adalah sebuah *firmware* interaktif berbasis *LUA Ekspressif* ESP8622 *wifi SoC* [9]. *NodeMCU* diimplementasikan dalam bahasa dalam C dan

di lapisi pada *Ekpressif NON SDK*. Gambar bentuk fisik *NodeMCU* ESP8266 ada pada gambar 2.6 dibawah ini:

Gambar 2.6 *NodeMCU* ESP8266

Pemetaan pin modul *wifi NodeMCU* ESP8266 ditunjukkan pada gambar 2.7 dibawah ini:



Gambar 2.7 Pemetaan pin *NodeMCU* V.1

NodeMCU ESP8266 ini bisa diprogram menggunakan *ARDUINO IDE* dan menggunakan bahasa C arduino, maka perlu *mengupdate firmware* modul ini. Untuk melakukan *update firmware* sama langkah pada proses *update firmware* modul *WiFi* ESP8266-01.

2.4 *Solenoid Door Lock*

Solenoid Door Lock adalah salah satu *solenoid* yang difungsikan khusus sebagai *solenoid* untuk pengunci pintu secara elektronik [10]. *Solenoid* ini mempunyai dua sistem kerja, yaitu *Normaly Close (NC)* dan *Normaly Open (NO)*. Perbedaan dari keduanya adalah jika cara kerja *solenoid NC* apabila diberi tegangan, maka *solenoid* akan memanjang (tertutup). Dan untuk cara kerja dari *Solenoid NO* adalah kebalikannya dari *Solenoid NC*. Biasanya kebanyakan *solenoid Door Lock* membutuhkan input atau *tegangan* kerja 12V DC tetapi ada juga solenoid Door Lock yang hanya membutuhkan input tegangan 5V *DC* dan sehingga dapat langsung bekerja dengan *tegangan* output dari pin *IC* digital. Bentuk fisik *solenoid door lock* di gambar 2.8 berikut:

Gambar 2.8 Bentuk fisik *Selenoid Door Lock*

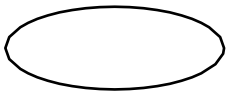
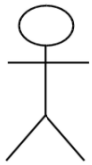
2.5 *UML (Unified Modeling Language)*

UML adalah bahasa spesifikasi standar yang digunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak[11]. Alat Bantu yang digunakan dalam perancangan berorientasi objek berdasarkan *UML* adalah sebagai berikut

1. *Use Case Diagram*

Use case diagram merupakan pemodelan untuk 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* dapat dilihat pada tabel 2.4 berikut:

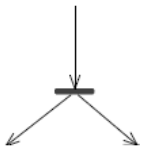
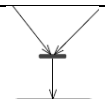
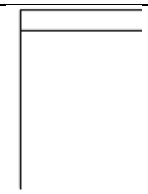
Tabel 2.4 Diagram *Use Case*

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistemsebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi 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 kontrol 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 mengindikasika n data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasika n bila aktor berinteraksi secara pasif dengan sistem
	Include, merupakan di dalam *use case* lain (*required*) atau panggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi

2. Diagram *Activity*

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah sebuah aliran kerja sistem sebelumnya (internal *processing*). Diagram ini lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Tabel 2.5 *Activity* Diagram

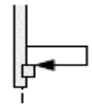
Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities,* menggambar kan suatu proses/kegiatan bisnis
	*Fork,* digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points*, menggambar kan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

3. *Diagram Sequence*

Diagram Sequence merupakan salah satu diagram *interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Simbol yang digunakan dalam Diagram *Sequence* seperti tabel 2.6 berikut:

Tabel 2.6 Diagram *Sequence*

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 *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan form *entry* 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* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline merupakan garis titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

4. *Class* Diagram

Class Diagram menggambarkan keadaan secara luas dari tiap-tiap kelas suatu sistem dan menghubungkan kelas tersebut. *Class* diagram menggambarkan struktur dan deskripsi atribut sebuah kelas yang dikoneksikan. *Class* Diagram secara khas meliputi, Kelas (*Class*), Relasi *Assosiations*, *Generalitation*, *Aggregation*, attribut (*Attributes*), operasi (*operation*), dan *visibility*. Tingkat akses objek eksternal kepada suatu operasi atau attribute dan Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality*.

2.6 *PHP (Personal Home Page)*

PHP (Personal Home Page) dimulai pada tanggal 1995 saat seseorang pengembang software independen bernama *Rasmus Lerdorf* mengembangkan *Perl/CGI script* dengan tujuan siapa yang mengunjungi website [9]. Konsep *php* diawali oleh suatu permintaan halaman web browser. Alamat dari web server

mengidentifikasi alamat yang dikehendaki dan menyampaikan informasi oleh Web Server. Web Server akan mengirim hasil kode *html* untuk menyampaikan ke *client* server [9].

2.7 *MySQL*

Merupakan database server bersifat open *source* berbasis database. *MySQL* dipakai untuk database pribadi berskala kecil hingga besar. Menggunakan *SQL* untuk mendukung penyimpanan akses data. Terdapat dua bagian dalam *MySQL* yaitu *DDL* (*Definition Data Language*) dan DML (*Data Manipulation Language*). *DDL* yang biasa sebagai bahasa pemograman yang merelasikan database maupun tabel untuk menciptakan database, membuat table, struktur table, merubah struktur table, menghapus table, dan menghapus database. DML sekumpulan sintaks yang digunakan untuk memanipulasi data seperti memilih, memasukan, menghapus dan memperbarui data yang terdapat pada database [9].