BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

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

Berdasarkan penelitian yang dilakukan oleh (Pandia Gunawan Situmorang, 2017) dengan judul "Perancangan Aplikasi Penyandian Pesan *Chatting Client* dan *Server* dengan Algoritma RC5" diperoleh kesimpulan bahwa Metode RC5 merupakan kriptografi primitif yang merupakan sasaran pengkajian RC5. Data *dependent rotations* merupakan suatu teknik yang merotasi data secara sirkuler sebanyak N rotasi. Didalam metode *enkripsi* algoritma RC5 menggunakan metode simetrik dan pengolahan dalam bentuk *blok chipper*. Proses penyandian pesan *chatting client* dan *server* dapat dilakukan dengan algoritma *rivest code* 5 sehingga pesan yang dikirim dan diterima tidak dapat dibaca dan dimengerti oleh sembarang pihak. Cara kerja algoritma *rivest code* 5 dalam proses penyandian pesan dalam penyisipan kata kunci yang ingin disisipkan dapat merubah pesan *chatting m*enjadi berbentuk karakter-karakter simbol dan huruf yang tidak sesuai dengan pesan yang dikirim oleh pengirim.

Berdasarkan penelitian yang dilakukan oleh (Somya, 2018) dengan judul "Perancangan Aplikasi *Chatting* Berbasis *Web* di PT. Pura Barutama Kudus Menggunakan *Socket*. IO dan *Framework Foundation*" diperoleh kesimpulan

bahwa aplikasi *chatting* dapat dibuat dengan menggunakan *Framework CodeIgniter, SocketIO* dan *Framework Foundation*. Pemanfaatan *Framework CodeIgniter* dapat diterapkan konsep MVC (*Model View Controller*) sehingga penulisan kode menjadi lebih terstruktur dan terorganisir. Pemanfaatan *SocketIO* cocok untuk pembuatan aplikasi *realtime* seperti aplikasi *chatting* dan *framework foundation* membuat tampilan aplikasi menjadi *responsive* yang dapat menyesuaikan diberbagai resolusi layar. Aplikasi *chatting* ini juga dapat membantu karyawan EDP keuangan atau *user* lainnya untuk berkomunikasi dan menyebarkan informasi jika sewaktu-waktu terjadi penambahan atau perubahan program yang dikembangkan oleh karyawan EDP keuangan. Spesifikasi *software* maupun *hardware* aplikasi *chatting* ini tergolong aplikasi yang ringan karena merupakan apliksi berbasis *web*, *user* hanya membutuhkan *web browser* kemudian mengakses aplikasi *chatting* yang sudah *diupload* ke *server.*

Berdasarkan penelitian yang dilakukan oleh (Khairil, 2020) dengan judul "Rancang Bangun Aplikasi *Chatting* Keluarga Menggunakan Fitur *Device Location* Berbasis Android" diperoleh kesimpulan bahwa Sistem aplikasi menggunakan sistem *client-server* secara *online* dan menggunakan sistem API. Aplikasi diperuntukkan untuk masyarakat umum yang ingin melakukan *chatting* bersama keluarga dan khusus orang tua yang ingin melihat lokasi keluarganya. Terciptanya aplikasi *chatting* keluarga menggunakan fitur *device location* berbasis android dengan menggunakan *tools software Android Studio, Visual Studio Code, Hosting*, dan *MySQL Database*. Untuk fitur *device location* nya penulis menggunakan API *Maps* SDK yang tersedia dari *google*. Dan aplikasi diberi nama dengan "CLK (*Chatting* Lokasi Keluarga)".

Berdasarkan yang dilakukan oleh (Neforawati, 2017) dengan judul "Rancang Bangun Aplikasi *Chatting* Berbasis *Web* Menggunakan *Docker*" diperoleh kesimpulan bahwa selama perancangan sampai implementasi teknologi *docker* pada *cloud computing* ini Aplikasi *Chatting* dapat berjalan dengan baik menggunakan teknologi *Docker*. Berberapa aplikasi hanya dapat dibuka jika computer *user* terhubung dengan internet. Jika *server* yang satu sedang *down* dan ingin memindahkan aplikasi kita ke *server* yang lain dengan *docker* akan menjadi lebih mudah. Aplikasi *chatting* dalam penelitian hanya dapat mengirim pesan teks, gambar, pesan suara maupun video call berbasis *web*. Dalam melakukan konfigurasi dalam melakukan implementasi teknologi *docker* pada *cloud computing* ini masih menggunakan *CLI*.

Berdasarkan penelitian yang dilakukan oleh (Saipul Anwar, 2020) dengan judul "Aplikasi Perbandingan Metode *Vernam* Dan Kombinasi OTP dengan *Rot* 13 pada *Wirelles Chatting* LAN dan WLAN" diperoleh kesimpulan bahwa Hasil perbandingan menunjukkan bahwa kecepatan pengiriman pesan setelah dilakukannya *enkripsi* ternyata metode *vernam cipher* memiliki keunggulan dibandingkan dengan kombinasi metode OTP dengan *Rot* 13. Dengan menggunakan teknik kriptografi dan menerapkan metode *vernam cipher* dan kombinasi metode OTP dengan ROT13 maka pesan *chatting local area network* dapat diamankan. Dengan menggunakan pemrograman *visual basic* 2010 maka dapat menghasilkan Aplikasi Perbandingan Metode Vernam Dan Kombinasi *OTP* Dengan ROT3 Pada *Wirelles Chatting* LAN Dan WLAN. Hasil enkripsi ditentukan dari kunci yang digunakan, semakin sulit kunci yang digunakan maka semakin kuat

hasil penyandian pesan yang didapatkan. Penggunaan kombinasi metode dapat menghasilkan hasil enkripsi yang sulit dipecahkan oleh pencuri informasi.

II.2. Landasan Teori

II.2.1. Kriptografi

Kriptografi merupakan ilmu dan seni untuk menjaga kerahsiaan berita (Bruce Schneier – *Applied Cryptography*). Selain itu kriptografi juga diartikan sebagai ilmu yang mendalami teknik – teknik matematika yang kaitannya dengan aspek keamanan informasi misalnya kerahasiaan data, keabsahan data, kredibilitas, data, serta keaslian data. Tetapi tidak segala aspek keamanan dan keselamatan informasi diatasi oleh kriptografi (Musfiroh, 2017)

Sedangkan menurut (Simargolang, 2017) Kriptografi merupakan ilmu mentransformasikan data jelas (*plaintext*) kedalam bentuk sandi (*ciphertext*) yang tidak dapat dikenali. *Ciphertext* inilah yang kemudian dikirim oleh pengirim kepada penerima. Setelah sampai di penerima, *ciphertext* tersebut ditransformasikan kembali kedalam bentuk *plaintext*.

Di dalam kriptografi sering dijumpai berbagai istilah atau terminologi, berikut ini adalah istilah-istilah dalam kriptografi : (Musfiroh, 2017)

1. Pesan (*message*) adalah data atau informasi yang dapat dibaca atau dimengerti maknanya. Nama lainnya untuk pesan adalah *plainteks* atau teks jelas (*clear text*).
2. Pengirim (*sender*) adalah entitas yang melakukan pengiriman pesan kepada entitas lainnya.
3. Kunci (*cipher*) adalah aturan atau fungsi matematika yang digunakan untuk melakukan proses enkripsi dan dekripsi pada *plaintext* dan *ciphertext*.

4. Enkripsi adalah mekanisme yang dilakukan untuk merubah *plaintext* menjadi *ciphertext*.
5. Dekripsi adalah mekanisme yang dilakukan untuk merubah *ciphertext* menjadi *plaintext*.
6. Penerima (*recipient*) adalah entitas yang menerima pesan dari pengirim/entitas yang berhak atas pesan yang dikirim.

II.2.2. Algoritma Rot 13

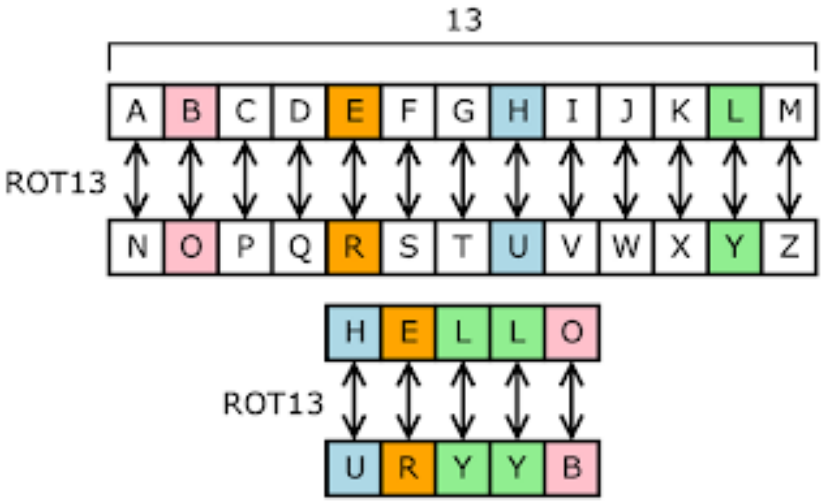
ROT13 (Rotate 13) merupakan perkembangan atau modifikasi dari metode *Caesar Cipher* dalam segi perubahan jumlah geseran dan juga arah geseran. Pada sistem *enkripsi ROT*13 sebuah huruf digantikan dengan huruf yang letaknya di atas 13 posisi darinya. *ROT* 13 adalah fungsi yang menggunakan kode *Caesar* dengan pergeseran k=13. *ROT13* didesain untuk keamanan pada sistem operasi *UNIX* yang sering digunakan pada forum *online*, berfungsi untuk menyelubungi isi artikel sehingga hanya orang yang berhak yang dapat membacanya. (zuhri, 2018)

Algoritma ROT13 adalah algoritma yang mudah untuk diimplementasikan dalam melindungi informasi. Algoritma ini bekerja sangat cepat dan akurat. Tidak membutuhkan kunci khusus dalam melakukan proses *enkripsi* dan *dekripsi*. Kelemahan algoritma ini adalah *ciphertext* dapat ditebak dengan beberapa kali percobaan sehingga algoritma ini akan lebih baik jika dikombinasikan dengan algoritma lain atau ditambah penggunaan kunci khusus. (HASIBUAN, 2020)

ROT13 merupakan pergeseran *key* 13, sehingga abjad ‘A’ menjadi ‘N‘ dan sebaliknay abjad ‘N‘ menjadi ‘A‘. Misalkan *plaintext* ‘AKU’. Setelah dilakukannya *enkripsi* menggunakan *Rot* 13, maka *plaintext* ‘AKU’ menjadi ‘NXH’. (Redho Maland Aresta, 2020)

ROT13 (Rotate 13) adalah *substitution enkripsi cipher* yang biasa diperlukan di dalam sistem operasi *UNIX*. Pada sistem *enkripsi* ROT13 digantikan dengan huruf yang sebuah huruf letaknya di atas 13 posisi darinya (Manullang, 2020)

Untuk lebih jelasnya dapat dilihat pada Gambar 2.1



Gambar II. 1. Rumus ROT13

(Redho Maland Aresta, 2020)

Secara sistematis algoritma *ROT13* dapat didefinikan menjadi dua, pertama untuk *enkripsi* menggunakan rumus: **C ROT13** = **(M)** atau **C** = **(i + 13)** *mod* 26 dan jika hasil lebih dari 26 maka **C** = **(i + 13)** – 26. Untuk dekripsi menggunakan rumus: **M** = ***ROT13*** **(*ROT13*(M))** atau **P** = **(C – 13)** ***mod*** **26** dan jika hasil minus maka **C** = **(i - 13)** + 26 (Redho Maland Aresta, 2020)

II.2.3. Algoritma Vigenere Cipher

Vigenere Cipher adalah salah satu metode penyandian dalam *kriptografi*. Metode *Vigenere* melakukan penyandian dengan menggunakan tabel diagram dengan huruf *alfabet* yang terurut secara diagonal. (Zulfatul Aulia, 2021)

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Tabel II. 1. Tabel *Vigenere Cipher*

Zulfatul Aulia, 2021)

Untuk menyandikan suatu pesan, digunakan sebuah tabel alfabet yang disebut Tabel *Vigenere*. Tabel *Vigenère* berisi alfabet yang dituliskan dalam 26 baris, masing-masing baris digeser satu urutan ke kiri dari baris sebelumnya, membentuk ke-26 kemungkinan sandi *Caesar*. Setiap huruf disandikan dengan menggunakan baris yang berbeda-beda, sesuai kata kunci yang diulang. Kata kunci yang digunakan dibuat secara otomatis di dalam script. Ada rumus untuk enkripsi dan dekripsinya yaitu:

Rumus *Enkripsi Vigenere Cipher*: ***Ci = (Pi+Ki) mod 26*** atau **jikas hasil lebih dari 26 maka Ci = (Pi+Ki) - 26**. Rumus *Dekripsi Vigenere Cipher*: ***Ci = (Pi-Ki) Mod 26*** atau jika hasil minus maka ***Ci = (Pi-Ki)+26*** .

Vigenere memproses informasi dalam bentuk teks antara teks informasi pesan dan teks kata kunci, sehingga *vigenere* juga dikategorikan sebagai *polyalphabetic cipher* . Kunci dalam *Vigenere* digunakan secara berulang sebanyak jumlah teks informasi yang akan disandikan. Proses pada *Vigenere Cipher* digambarkan seperti proses *Caesar Cipher* dengan nilai pergeseran karakter yang berbeda. (E. Ardhianto, 2021)

Vigenere cipher merupakan pengembangan dari *caesar cipher*. Pada *caesar cipher*, setiap huruf pada *plaintext* digantikan dengan huruf lain yang memiliki perbedaan tertentu pada urutan alfabet. Sedangkan pada *vigenere cipher*, setiap karakter pesan pada *plaintext* berkorespondensi dengan lebih dari satu karakter pada *ciphertext*. Misalnya, huruf A pada *plaintext* dapat menjadi huruf K atau M pada *ciphertext* yang berkaitan, tergantung pada kunci yang digunakan. (Adryan Rachmadsyah, 2020)

Vigenere cipher merupakan algoritma kriptografi klasik yang dibuat pada abad ke 16 atau pada tahun 1986. Algoritma kriptografi sendiri ini dipublikasikan oleh seorang diplomat dan juga kriptologis yang berasal dari prancis, yaitu *Blaise deVigenere*. Cara kerja *vigenere cipher* ini mirip dengan *Caesar cipher*, yaitu mengenkripsi *plaintext* pada pesan dengan cara menggeser huruf pada pesan tersebut sejauh nilai kunci pada deret *alphabet.* (Gyna Rahmi Fajri, 2020)

II.2.4. *Android*

Android adalah sebuah sistem operasi perangkat mobile berbasis *linux* yang mencangkup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, *Google Inc*. membeli *Android Inc*. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel atau *smartphone*. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, *konsorsium* dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk *Google*, *HTC, Intel, Motorola, Qualcomm, T-Mobile,* dan *Nvidia.*

Pada saat perilisan perdana *Android*, 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada

perangkat *mobile*. Di lain pihak, *Google* merilis kode-kode *Android* di bawah *lisensi Apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler. (Joni Karman, 2017)

II.2.5. *Chatting*

Chatting merupakan suatu jenis dari perkembangan suatu teknologi. Dengan kecanggihan teknologi saat ini, fungsi aplikasi *chating* tidak hanya sebagai alat komunikasi biasa, tetapi manusia juga dapat mengirimkan foto dan lain-lainya. Dampak yang ditimbulkan dari *chatting* mungkin tidak disadari sama sekali. Selain memudahkan dalam berkomunikasi sebagai dampak *positif* yang didapatkan manusia, terdapat pula dampak *negatif* yang didapatkan manusia sebagai akibat menggunakan *chatting* ini. (Pandia Gunawan Situmorang, 2017)

Kerahasiaan dan keamanan saat melakukan pertukaran pesan adalah hal yang sangat penting didalam berkomunikasi, baik untuk tujuan keamanan bersama, maupun untuk privasi individu. Mereka yang menginginkan agar pesannya tidak diketahui oleh pihak- pihak yang sama sekali tidak berkepentingan selalu berusaha menyiasati cara mengamankan informasi yang akan dikomunikasikan. Perlindungan terhadap kerahasiaan pesan pun meningkat, salah satu caranya dengan menyandikan pesan atau *enkripsi*. Skema *enkripsi* yang akan dibangun pada masalah ini menerapkan teknik pada *kriptografi modern*, yang menganut kerahasiaan pada kunci (*key*), sehingga keamanan *enkripsi* hanya tergantung pada kunci (*key*) dan tidak tergantung apakah algoritmanya diketahui orang atau tidak. (Pandia Gunawan Situmorang, 2017)

II.2.6. Java

Java merupakan pemrograman yang bersifat lintas *platform*. Artinya, bahasa ini dapat dipakai untuk menyusun program pada berbagai *system* operasi (*Linux, Windows, UNIX*) dan sebuah bahasa pemrograman yang dapat digunakan di berbagai *platform*. Bahasa pemrograman java menjadi bahasa untuk pengembangan berbasis jaringan besar, karena dengan menggunakan java seorang pengembang dapat dengan mudah menjalankan program di semua *computer* yang support dengan java. (Afrizal, 2017)

II.2.7. *Firebase*

Layanan *firebase* merupakan sebuah *platform* yang tujuannya untuk memudahkan pengembangan sistem yang menggunakan sebuah *resource REST API*. Dalam pengembangan aplikasi berbasis *android* untuk berkomukasi dengan *server* umumnya menggunakan *REST API* akan tetapi dalam proses pembuatannya sangatlah lama karena beberapa faktor seperti keamanan, kecepatan, dan kemudahan akses, *firebase* hadir untuk memangkas kegiatan pengembangan *REST API* tersebut sehingga memudahkan pengembang aplikasi dalam pembuatan aplikasi. (Teguh Kurniawan, 2021)

Adapun layanan *firebase* yang digunakan pada penelitian ini adalah sebagai berikut:

a. *Firebase Authentication*

   Sebuah layanan dari *firebase* yang berfungsi untuk melakuka proses *Authentication* pada aplikasi, *firebase authentication* sendiri menyediakan cukup banyak metode autentikasi (pengenalan identitas user), pada saat penelitian ini dibuat adapun metode tersebut adalah *email, phone number, facebook account, github account, google*

account, yahoo account, microsoft account, apple account, twitter account.

b. *Firebase Realtime Database*

salah satu layanan dari *firebase* yang bertujuan untuk melakukan *manajemen database*, bersifat *NoSQL* dan dalam bentuk JSON. Layanan ini sangat optimal untuk digunakan karena kemampuannya dalam melakukan proses komunikasi dengan *Client* sangat cepat.

II.2.8. Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi *Android* dan bersifat *open source.* Peluncuran *Android Studio* ini diumumkan oleh *Google* pada 16 mei 2013 pada event *Google* I/O *Conference* untuk tahun 2013. Sejak saat itu, *Android Studio* mengantikan *Eclipse* sebagai *IDE* resmi untuk mengembangkan aplikasi *Android*. (Joni Karman, 2017)

Android studio sendiri dikembangkan berdasarkan *IntelliJ IDEA* yang mirip dengan *Eclipse* disertai dengan ADT *plugin* (*Android Development Tools*). *Android studio* memiliki fitur :

a. Projek berbasis pada *Gradle Build*

b. *Refactory* dan pembenahan *bug* yang cepat

c. *Tools* baru yang bernama *"Lint"* diklaim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.

d. Mendukung *Proguard* dan *App-signing* untuk keamanan.

e. Memiliki *GUI* aplikasi *android* lebih mudah

f. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

II.2.9. *Unfied Markup Language*

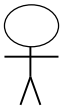
UML yaitu satu kumpulan *konvensi* permodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak yang terkait dengan objek. *UML* merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan *system* yang besar dan kompleks. *UML* tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. (Andikos, 2019)

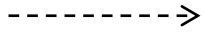
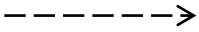
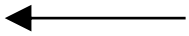
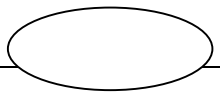
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut :

1. *Use Case* Diagram

Use case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah *sistem*. Yang ditekankan adalah "apa" yang diperbuat *sistem*, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem *use case diagram* dapat digambarkan dengan sumber-sumber pada Tabel II.2

Tabel II. 2. Simbol *Use Case*

Gambar	NAMA	Keterangan
	Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case.

	Depedency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
	Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	Include	Menspesifikasikan bahwa use case sumber secara eksplisit.
	Extend	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
	Association	Apa yang mnghubungkan antara objek satu dengan objek lainnya.
	System	Menspesifikan paket yang menampilkan sistem secara terbatas.
	Use Case	*Deskripsi* dari urutan aksi-aksi yang ditampilkan sistem yang

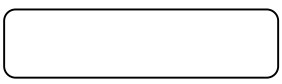
		menghasilkan suatu hasil yang terukur bagi suatu actor.
	Collaboration	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	Note	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

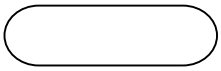
(Sumber : (Andikos, 2019): 39)

2. Diagram Aktivitas (*Activity Diagram*)

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* dapat digambarkan dengan simbol-simbol seperti pada tabel II.3.

Tabel II. 3. Simbol Activity Diagram

Gambar	Nama	Keterangan
	Activity	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.

	Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	Initial Node	Bagaimana objek dibentuk atau diawali.
	Activity Final	Bagaimana objek dibentuk dan dihancurkan
	Fork Node	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

(Sumber : (Andikos, 2019): 39)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* Diagram dapat digambarkan dengan simbol-simbol seperti pada Tabel II.4.

Tabel II. 4. Simbol *Sequence* Diagram

Gambar	Nama	Keterangan
	Lifeline	Objek entity, antarmuka yang saling berinteraksi.
	Message	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

	Message	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

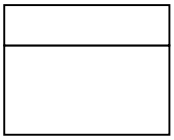
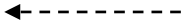
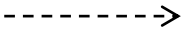
(Sumber : (Andikos, 2019): 39)

4. *Class Diagram* (Diagram Kelas)

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* diagram menggambarkan struktur dan *deskripsi class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* diagram dapat digambarkan dengan simbol-simbol seperti pada Tabel II.5.

Tabel II. 5. Class Diagram

Gambar	Nama	Keterangan
	Generalization	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
	Nary Association	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.

	Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	Collaboration	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	Realization	Operasi yang benar-benar dilakukan oleh suatu objek.
	Depedency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
	Assocation	Apa yang menghubungkan antara objek satu dengan objek lainnya

(Sumber : (Andikos, 2019): 3