BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Nurul Yalisa, et al., 2018 melakukan penelitian berjudul "Algoritma *Elgamal* Dengan Pertukaran kunci *Diffie Hellman* Pada Aplikasi Keamanan Citra Sidik Jari Berbasis Android". Penelitian tersebut menghasilkan sebuah aplikasi berbasis android yang berisikan tentang pengamanan citra dengan penerapan algoritma *elgamal* dengan pertukaran kunci *diffie hellman* untuk proses pengamanannya. Yang menjadi perbedaan dengan penelitian yang akan dilaksanakan adalah pada aplikasi akan diterapkan algoritma *data encryption standard* (*des*) untuk proses pengaman pesan chat.

Hermansyah Alam, et al., 2019 melakukan penelitian dengan judul "Rancang Bangun Aplikasi Penyandian Data Text Menggunakan Algoritma *Diffie Hellman* Dan Algoritma RC4". Penelitian tersebut menggunakan algoritma RC4 dan algoritma pertukaran kunci *diffie hellman* untuk pengamanan data text. Terdapat perbedaan dengan penelitian yang akan dibangun yaitu, pada penelitian yang akan dilaksanakan akan dibangun sebuah pengamanan pesan chat dengan menerapkan algoritma pertukaran kunci *diffie hellman* pada algoritma *data encryption standard (des)* untuk digunakan pada perangkat android.

Ana Wahyuni, et al., 2011 melakukan penelitian dengan judul "Keamanan Pertukaran Kunci Kriptografi dengan Algoritma Hybrid : Diffie Hellman Dan RSA". Penelitian tersebut menggunakan algoritma RSA dan algoritma pertukaran

kunci *diffie hellman* untuk pengamanan kriptografi. Terdapat perbedaan dengan penelitian yang akan dibangun yaitu, pada penelitian yang akan dilaksanakan akan dibangun sebuah pengamanan pesan chat dengan menerapkan algoritma pertukaran kunci *diffie hellman* pada algoritma *data encryption standard (des)* untuk digunakan pada perangkat android.

Rifkie Primatha, et al., 2011 melakukan penelitian dengan judul "Penerapan Enkripsi Dan Deskripsi File Menggunakan Algoritma Data Encryption Standard (DES)". Penelitian tersebut menggunakan algoritma DES untuk pengamanan data file. Terdapat perbedaan dengan penelitian yang akan dibangun yaitu, pada penelitian yang akan dilaksanakan akan dibangun sebuah pengamanan pesan chat dengan menerapkan algoritma pertukaran kunci *diffie hellman* pada algoritma *data encryption standard (des)* sedangkan pada penelitian terdahulu bertujuan untuk mengamankan data file.

Luthfiatun Nisa, et al., 2020 melakukan penelitian dengan judul "Aplikasi Enkripsi Citra Dan Teks Menggunakan Algoritma Diffie-Hellman Dan ElGamal". Penelitian tersebut menggunakan algoritma Elgamal dan algoritma pertukaran kunci *diffie hellman* untuk enkripsi citra dan teks. Terdapat perbedaan dengan penelitian yang akan dibangun yaitu, pada penelitian yang akan dilaksanakan akan dibangun sebuah pengamanan pesan chat dengan menerapkan algoritma pertukaran kunci *diffie hellman* pada algoritma *data encryption standard (des)*.

II.2. Uraian Teoritis

II.2.1. Algoritma

Algoritma adalah sistem kerja komputer memiliki *brainware*, *hardware*, dan *software*. Tanpa salah satu dari ketiga sistem 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*, diperlukannya 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 = a ditulis sebagai blog a = c (b disebut basis). (Maulana, Gun Gun ; 2017 : 70)

II.2.2. Algoritma *Data Encryption Standard* (DES)

DES (*Data Encryption Standard*) adalah algoritma *cipher* blok yang popular karena dijadikan standard algoritma enkripsi kunci simetri. DES (*Data*

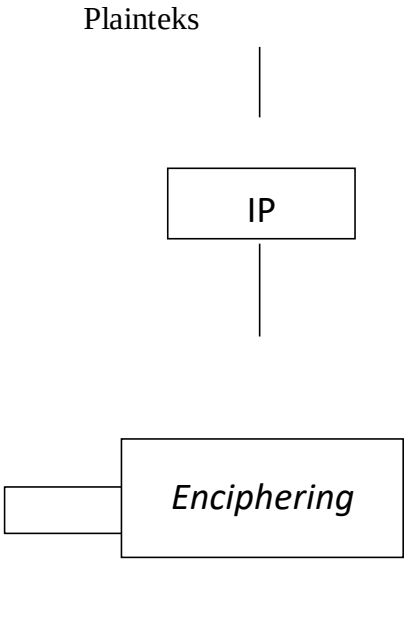
Encryption Standard) termasuk ke dalam sistem kriptografi simetri dan tergolong jenis *cipher* blok. DES (*Data Encryption Standard*) beroperasi pada ukuran blok 64 bit. Mengenskripsikan 64 bit plainteks menjadi 64 bit cipherteks dengan menggunakan 56 bit kunci internal.Kunci internal dibangkitkan dari kunci eksternal yang panjangnya 64 bit. (Muhammad Zacky, dkk, 2016).

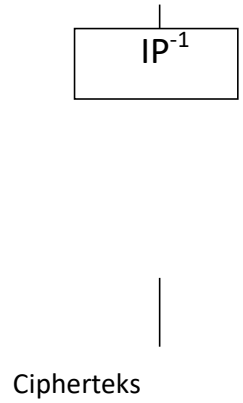
Skema global dari algoritma DES adalah sebagai berikut :

1. Blok plainteks dipermutasi dengan matriks permutasi awal (*initial permutation* atau IP).

2. Hasil permutasi awal kemudian di *enciphering* sebanyak 16 kali (16 putaran). Setiap putaran menggunakan kunci internal yang berbeda.

3. Hasil *enciphering* kemudian dipermutasi dengan matriks permutasi balikan (*invers initial permutation* atau IP^{-1}) menjadi blok cipherteks.





Gambar II.2.2. Skema Global Algoritma DES

II.2.3. Algoritma *Diffie-Hellman*

Algoritma *Deffie-Hellman* pertama kali dikenalkan oleh peneliti universitas Stanford yaitu Whitfield Diffie dan Martin Hellman pada tahun 1975. Mereka memperkenalkan algoritma ini untuk memberi solusi atas pertukaran informasi secara rahasia. Dasar dari algoritma ini adalah matematika dasar dari aljabar eksponen dan aritmatika modulus. Jumlah pengguna yang ingin menggunakan pertukaran kunci menggunakan algoritma *Diffie-Hellman* ini tidak dibatasi. Hal ini hanya berlaku jika memenuhi 2 prinsip yang harus dilakukan yaitu:

1. Bilangan p dan g yang telah disetujui oleh semua anggota.
2. Setiap anggota harus melakukan pertukaran data yang diperlukan oleh anggota lainnya sehingga semua data dapat didapatkan secara merata gabc….n.

Tingkat keamanan dari algoritma ini tinggi, jika nilai p dan g dipilih secara benar. Karena untuk mengetahui atau menebak nilai rahasia yang dimiliki

oleh penerima dan pengirim harus menyelesaikan persamaan *Diffie-Hellman* terlebih dahulu. Ini merupakan masalah logaritma diskrit yang perhitungan tersebut tidak dapat diselesaikan untuk nilai bilangan p yang sangat besar. Menghitung logaritma diskrit dari bilangan modulo p memakan waktu yang kurang lebih sama seperti dengan memfaktorkan bilangan non prima menjadi faktor primanya, seperti yang digunakan di algoritma RSA. Oleh karena itu, algoritma ini tingkat keamanannya setingkat dengan dengan algoritma RSA. (Dyas Yudi Priyanggodo, 2018).

II.2.4. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus *computer* eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu tehnik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu”. (Juansyah, Andi ; 2015 : 2)

II.2.5. Pemrograman Java

Java dikembangkan oleh *Sun Microsystems* pada Agustus 1991. Java disebut juga merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C dan C++. Pemrograman Java bersifat tidak bergantung pada *platform*, yang artinya, java dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Sebagaimana halnya C++, salah satu bahasa yang mengilhami Java, Java juga merupakan bahasa pemrograman berorientasi objek. Sebagai bahasa pemrograman berorientasi objek, Java menggunakan kelas untuk membentuk suatu objek. Karakteristik Java antara lain adalah berorientasi objek (*object-oriented*), terdistribusi (*distributed*), sederhana (*simple*), aman (*secure*), *interpreted*, *robust*, *multithreaded*, dan dinamis. (Annisa Rahmawati, et al. ; 2015 : 336)

II.2.6. Aplikasi *Mobile*

Aplikasi *mobile* berasal dari dua kata, yaitu aplikasi dan *mobile*. Secara istilah, aplikasi adalah program siap pakai yang dibuat untuk melaksanakan suatu fungsi untuk pengguna atau aplikasi yang lain sedangkan *mobile* adalah perpindahan dari suatu tempat ke tempat yang lain. Secara lebih lengkap, aplikasi *mobile* adalah program siap pakai yang melaksanakan fungsi tertentu yang dipasang pada perangkat *mobile*. (Mukmin Siregar ; 2016 : 83)

II.2.7. Android

Menurut Arifianto Teguh, android adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat

mobile, semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, Android merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh Google. Sedangkan Android *SDK (Software Development Kit)* menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* Android dengan menggunakan bahasa pemrograman Java.

Aplikasi *Android* ditulis dalam bahasa pemrograman *java,* yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi Android, sebuah file yang ditandai dengan akhiran *.apk.* file inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. File ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu file *.apk*, dan kemudian kita sebut sebagai sebuah aplikasi.(Ahmad : 2015 : 190-200)

II.2.8. Android *Studio*

Android *studio* adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *opensource* atau gratis. 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.

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" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
d. Mendukung *Proguard And App-signing* untuk keamanan.
e. Memiliki GUI aplikasi android lebih mudah
f. Didukung oleh Google *CloudPlatfrom* untuk setiap aplikasi yang dikembangkan. (Andi Juansyah ; 2015)

II.2.9. Android SDK (*Software Development Kit*)

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. Android SDK terdiri dari *debugger*, *libraries*, *handsetemulator*, dokumentasi, contoh kode program dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk *desktop* modern), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista. Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau lebih. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan *plugin Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML serta menggunakan peralatan *commandline* untuk menciptakan, membangun, melakukan *debug* aplikasi

Android dan pengendalian perangkat Android (misalnya *reboot*, menginstal paket perangkat lunak). (Sulihati dan Andriyani ; 2016 : 20)

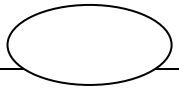
II.2.10. Pengertian UML

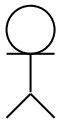
UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem (Ade Hendini ; 2016 : 108). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.2.10.1. *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. (Ade Hendini ; 2016 : 108)

Tabel II.1. *Use Case Diagram*

Gambar	Keterangan
	*UseCase* menggambarkan fungsionalitas yang disediakan sistem

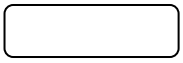
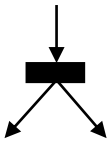
	sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktir, harus ditentukan pembagian tenaga kerja 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
 <<extends>>	*Include*, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *usecase* lain jika kondisi atau syarat

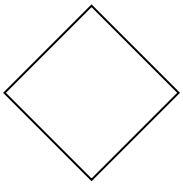
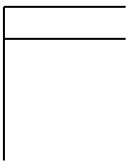
(Sumber : Ade Hendini ; 2016)

II.2.10.2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. (Ade Hendini ; 2016 : 109) Simbol-simbol yang digunakan dalam *activity Diagram* yaitu:

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

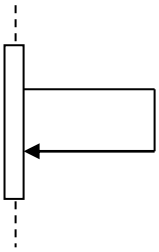
(Sumber : Ade Hendini ; 2016)

II.2.10.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Ade Hendini ; 2016 : 110). Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu:

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)

II.2.10.4. *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga

menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan obyek yang dikoneksikan. *Class diagram* secara khas meliputi:Kelas (*Class*), *Relasi, Associations, Generalization* dan *Aggregation,* Atribut *(Attributes),* Operasi *(Operations/Method),* dan *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar Kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti(Ade Hendini ; 2016 : 110).

Tabel II.4. *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)