# BAB II

# TINJAUAN PUSTAKA

## II.1. Penelitian Terdahulu

Menurut E.Z Adnan Kashogi (2012) pada penelitian dengan judul penelitian "Algoritma Message Digest 5 (MD5)". Di dalam mengirimkan suatu pesan pada jaringan, kita menghadapi beberapa persoalan yaitu kerahasiaan (confidentiality), integritas data (menjamin pesan tidak diubah oleh orang lain), keaslian pesan (authentication), dan tak terbantahkan (non-repudiation). Harapan penulis dengan makalah ini adalah dapat memberikan informasi tentang fungsi hash MD5, sehingga pembaca dapat lebih memahami fungsi hash MD5 dan juga mengetahui kekuatan dan kelemahannya.

Berdasarkan penelitian yang dilakukan oleh Rusdianto (2016) dengan judul Implementasi Algoritma Md5 Untuk Keamanan Dokumen. Keamanan data dan informasi menarik banyak perhatian orang memastikan keaslian data atau dokumen masih terjaga, masalah ini begitu urgen untuk dan menyentuh berbagai bidang termasuk saluran komunikasi yang aman, teknik enkripsi data yang kuat dan dipercaya dibutuhkan untuk menjaga database. Data chipertex yang dihasilkan akan berubah-ubah sesuai masukan data kunci password yang diberikan. Sistem ini dibuat dengan bahasa pemprokraman Visual Basic.Net.

Berdasarkan penelitian yang dilakukan oleh Yudi Prayudi (2015) dengan judul Kompleksitas Waktu Untuk Algoritma MD5. Integritas bukti digital adalah salah satu issue penting dalam aktivitas digital forensics. Secara umum, bukti

digital tidak boleh mengalami perubahan apapun dalam setiap tahap digital forensics. Dalam hal ini, fungsi hash secara umum dalam digital forensics telah digunakan untuk kepentingan menjaga integritas bukti digital.

## II.2. Landasan Teori

Untuk mendukung keberhasilan penelitian ini, penyusun melakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitian yang dilakukan. Beberapa tinjauan pustaka pada penelitian ini yaitu:

### II.2.1. Pengertian Aplikasi

Program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi software yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu (Rahmatillah ; 2011: 3).

### II.2.2. Fungsi Hash

Fungsi hash kriptografi adalah fungsi hashyang memiliki beberapa sifat keamanantambahan sehingga dapat dipakai untuk tujuan keamanan data.

Umumnya digunakan untuk keperluan autentikasi dan integritas data. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai hash. Fungsi hashkriptografi memiliki beberapa sifat antara lain sebagai berikut: a. Tahan preimej (Preimage resistant): bila diketahui nilai hash h maka sulit (secara komputasi tidak layak) untuk mendapatkan m dimana h = hash (m). b. Tahan preimej kedua (Second preimage resistant): bila diketahui input m1 maka sulit mencari input m2 (tidak sama dengan m1) yang menyebabkanhash(m1) = hash (m2). c. Tahan tumbukan (Collision-resistant): sulit mencari dua input berbeda m1 dan m2 yang menyebabkan hash(m1) = hash(m2). (sumber:http://books.google.co.id/books?id=CFq5qdse2j4C&pg=PA19&dq=algoritma+md5&hl=id&sa=X&ei=XUEcUoe6I4KRrAfSnYCgDg&sqi=2&ved=0CD4Q6AEwAw#v=onepage&q=algorit ma%20md5&f=false).

Hash Tertutup (Closed Hash) didefinisikan sebagai cara hash dimana data langsung disimpan dalam tabel hash dengan ukuran tabel tertentu (dapat ditentukan sendiri). Fungsi hash tersebut dapat didefinisikan sebagai berikut, H(X) = X mod [ukuran tabel] dengan :X= nilai data H(X) = hasil fungsi hash. Implementasi metode hash dalam pencarian arsip. Membentuk Tabel No Index Hash Huruf Tabel no index hash ini digunakan untuk memberikan no index dari masing-masing huruf sebagai identitas pengenal untuk perhitungan hash, sebagai contoh dapat dilihat di bawah ini :

Lanjutan

| 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 |

### II.2.3. Data

Data adalah sebagai bahan keterangn tentang kejadian nyat atau fakta-fatkat yang dirumuskan dalam sekelompok lambang tertentu yang tidak acak yang menunjukan jumlah, tindakan, atau hal". Data dapat perupa catatan-catatan dalam kertas, buku, atau tersimpan sebagai file dalam basis data (Hermansyah Sembiring ; 2012 : 14).

Definisi Data secara Etimologis merupakan bentuk jamak dari DATUM yang berasal dari Bahasa Latin dan berarti "Sesuatu Yang Diberikan". Dalam pengertian sehari-hari DATA dapat berarti Fakta dari suatu objek yang diamati, yang dapat berupa angka-angka maupun kata-kata. Sedangkan jika dipandang dari sisi Statistika, maka DATA merupakan Fakta-fakta yang akan digunakan sebagai bahan penarikan kesimpulan (Dodiet Aditya ; 2013 : 1).

Data adalah kumpulan file-file yang saling berelasi, relasi tersebut biasa ditunjukan dengan kunci dari tiap file yang ada. Satu basis data menunjukkan kumpulan data yang dipakai dalam satu lingkup informasi. Dalam satu file terdapat record-record yang sejenis, sama besar, sama bentuk, merupakan satu kumpulan entity yang seragam. Satu record terdiri dari fieldfield yang saling berhubungan untuk menunjukan bahwa field tersebut dalam satu pengertian yang lengkap dan direkam dalam satu record. Suatu sistem manajemen basis data berisi

satu koleksi data yang saling berelasi dan satu set program untuk mengakses data tersebut. Jadi sistem manajemen basis data dan set program pengelola untuk menambah data, menghapus data, mengambil data dan membaca data (Mhd Bustanur Rahmad ; 2014 : 1333).

### II.2.4. Algoritma MD5

Message Digest 5 (MD5) adalah salah satu dari serangkaian algoritma Message Digest yang didesain oleh Professor Ronald Rivest dari MIT. Saat kerja analitik menunjukkan bahwa pendahulu MD5 -MD4- mulai tidak aman, MD5 kemudian di desain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin). MD5 banyak digunakan pada bermacammacam aplikasi termasuk SSL/TLS, IPSec dan protocol-protokol kriptografi lainnya. MD5 juga biasa digunakan pada implementasi *Timestamping Mechanism, Commitment Schemes*, dan aplikasi pengecekan integritas pada *online software*. MD5 tidak memiliki sistim pengamanan seperti persamaan matematika, namun untuk setiap fungsi hash h, domain D dan range R kita membutuhkan tiga hal berikut :

1. *Pre Image Resistance* : jika diberi suatu nilai y ε R, maka kita tidak akan dapat mencari suatu nilai x ε D dimana h(x)=y.
2. *Second Pre Image Resistance* : jika diberi suatu nilai x ε D, maka kita tidak akan dapat mencari nilai x’ ε D dimana h(x)=h(x’).
3. *Collision Resistance* : kita tidak akan dapat mencari nilai x,x’ ε D dimana h(x)=h(x’) (E. Z Adnan Kashogi ; 2012 : 3)

### II.2.5. Pengertian *Android Studio*

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat open *source* 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 Cloud Platfrom* untuk setiap aplikasi yang dikembangkan (Andi Juansyah ; 2015 : 3)

### II.2.6. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

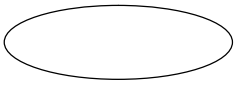
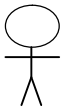
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

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

- *Use case* Diagram

*Use case* diagram merupakan pemodelan untuk kelakukan (*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 di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

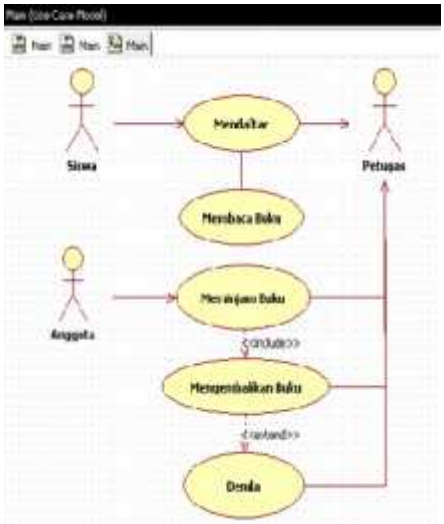
**Tabel II.1. Simbol *Use Case***

| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*. |
|  | Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan 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 control 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 mengidikasikan aliran data. |
| ⟶ | Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem. |
| - - - - - - - - - - ⟶ | *Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program. |
| ⟵ - - - - - - - - - - | *Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi. |

***(Sumber : Windu Gata ; 2013 : 4)***

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.2 berikut :

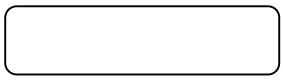
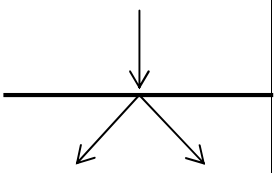
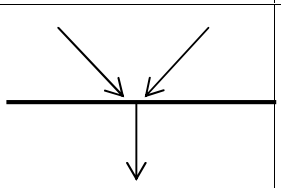
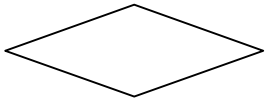
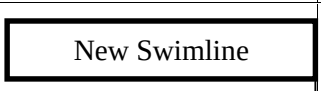



**Gambar. II.2. *Use Case Diagaram***
***(Sumber : Windu Gata ; 2013 : 4)***

- Diagram Aktivitas (*Activity Diagram*)

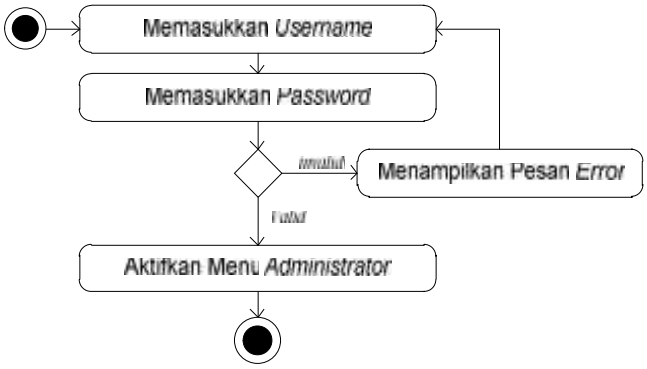
*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

**Tabel II.2. Simbol *Activity Diagram***

| Gambar | Keterangan |
|---|---|
|  | *Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | *End point*, akhir aktifitas. |
|  | *Activites*, menggambarkan suatu proses/kegiatan bisnis. |
|  | *Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu. |
|  | *Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi. |
|  | *Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*. |
|  | *Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa. |

***(Sumber : Windu Gata ; 2013 : 6)***

Contoh dari pembuatan *activity diagram* dapat dilihat pada gambar II.2 berikut :

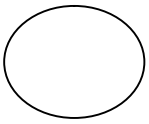
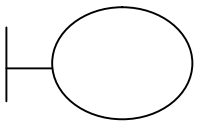
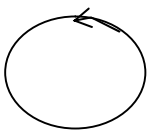


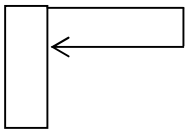

**Gambar. II.2. *Activity Diagaram***
***(Sumber : Windu Gata ; 2013 : 6)***

- 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 :

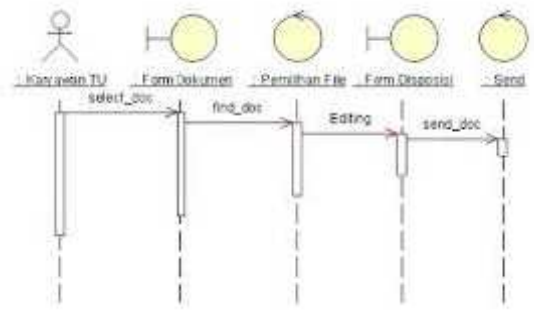
**Tabel II.3. Simbol *Sequence Diagram***

| 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 *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry 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, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi. |
|  | *Lifeline,* garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*. |

***(Sumber : Windu Gata ; 2013 : 7)***

Contoh dari pembuatan *sequence diagram* dapat dilihat pada gambar II.4 berikut :




**Gambar. II.3. *Sequence Diagaram***
***(Sumber : Windu Gata ; 2013 : 7)***

- *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

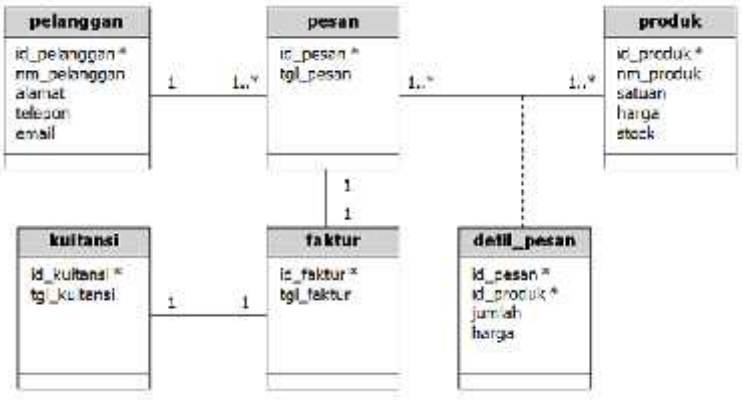
*Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*),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

**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 maksimum 4 |

***(Sumber : Windu Gata ; 2013 : 8)***

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.5 berikut :




**Gambar. II.5. *Class Diagaram***
***(Sumber : Windu Gata ; 2013 : 8)***