# BAB II

# LANDASAN TEORI

## Pengertian Kriptografi

Kriptografi merupakan suatu ilmu yang menjelaskan bagaimana cara seseorang untuk mengirimkan sebuah pesan kepada seseorang yang dituju tanpa di interupsi dan hanya orang tersebut lah yang mengetahui *password* yang dikenakan pada pesan tersebut. (Chuck, 2016). Banyak dari kita yang sering mengartikan kriptografi merupakan hal yang sama dengan kriptologi. Kriptologi lebih komperhensif (formal) dan mencakup dari kriptografi dan kriptanalisis.

Berikut ini penulis akan jelaskan berbagai macam pengertian yang sering kita jumpai pada Kriptografi:

### *Cipher*, sebuah algoritma yang digunakan untuk merubah sebuah teks biasa menjadi teks yang sudah dimodifikasi (*cipher text*).

### *Cipher Text*, merupakan hasil dari *cipher* yang merupakan teks yang sudah dimodifikasi.

### *Cryptanalysis*, (biasa dikenal sebagai *Code Breaking*) merupakan studi tentang prinsip dan metode yang mengartikan *cipher text* tanpa mengetahui kunci algoritma.

### *DeCipher* (*Decrypt*), merupakan cara yang digunakan untuk merubah teks yang telah dimodifikasi (*cipher text*) menjadi teks aslinya (p*lain text*).

### *Encipher* (*Encrypt*), merupakan cara yang digunakan untuk merubah teks biasa (*plain text*) menjadi teks termodifikasi (*cipher text*).

### *Key*, merupakan kata kunci yang digunakan oleh algoritma Kriptografi dalam melakukan *encrypt* maupun *decrypt* data.

Setelah kita mengetahui pengertian dari kriptografi. Berikutnya akan dijelaskan mengenai cara kriptografi bekerja, Lihat Gambar 2.1:

Gambar 2.1

Model *Standard* dari *Cryptosystem*

Sumber: (Martin, 2017)

### *Sender* mengirimkan *plain text* yang harus dilindungi selama transmisi dari pengirim ke penerima.

### *Plain text* di enkripsi dengan algoritma enkripsi, dengan menggunakan kunci enkripsi agar *plain text* menjadi *cipher text*. Tujuannya adalah bahwa pada akhir proses hanya pengirim dan *receiver* akan mengetahui *plain text*. Secara khusus, *interceptor* tidak dapat menentukan *plain text*.

### *Cipher text* bukanlah rahasia dan dapat diperoleh oleh siapa saja yang memiliki akses ke komunikasi. Termasuk *Interceptor*. Tetapi dibutuhkan algoritma dan kunci yang tepat agar *cipher text* dapat diubah menjadi *plain text*.

### Dengan penerima yang tepat (memiliki algoritma deskripsi & kunci dekripsi) *cipher text* yang dikirmkan dapat diubah menadi *plain text*. Sehingga *receiver* mengerti apa yang dikirimkan oleh *sender*.

Terdapat dua teknik penguncian pada kriptografi yaitu:

## *Symmetric Keys* (Kunci Simetris)

Kunci simetris merupakan kunci yang digunakan pada algoritma enkripsi dan dekripsi dimana kunci yang digunakan untuk enkripsi sama dengan kunci yang digunakan untuk dekripsi. Untuk lebih jelas dapat dilihat pada Gambar 2.2.



Gambar 2.2

Diagram Kunci Simetris

Sumber: (Dunkerley, 2017)

Kemanan dari model enkripsi ini tergantung dari pengguna untuk mengamankan *secret key.* Jika ada orang yang tidak bertanggung jawab mendapatkan kunci, dia dapat membaca seluruh pesan yang sudah di enkripsi dengan cara mendekripsi pesan tersebut dengan kunci yang telah didapatkan.

Algoritma enkripsi yang menggunakan Kunci Simetris antara lain:

#### *Data Encryption Standard* (DES)

#### *Advanced Encryption Standard* (AES)

#### *Blowfish*

#### *Twofish*

#### *International Data Encryption Algorithm* (IDEA)

#### *Rivest Cipher* (RC4)

Berikut ini merupakan tabel mode operasi yang dilakukan pada enkripsi maupun dekripsi dari Kunci Simetris

Table 2.1

*Modes of Operation Symmetric Key*

|  |  |  |
| --- | --- | --- |
| Nama | Akronim | Deskripsi |
| *Electronic Codebook* | ECB | Nilai *plaintext* yang sama akan selalu menghasilkan nilai *ciphertext* yang sama |
| *Cipher Block Chaining* | CBC | Mode yang mana proses enkripsinya menggabungkan blok *plaintext* dengan hasil *Ciphertext* sebelumnya. |
| *Cipher Feedback* | CFB | Mode yangmana memasukan hasil unpan balik sukses *Ciphertext* sebagai *input block* untuk menghasilan output lalu dilakukan *eXclusive OR* dengan plainteks untuk menghasilkan Cipherteks. |
| *Output Feedback* | OFB | Mode yang mana merupakan iterasi dari *cipher* untuk menghasilkan urutan *output* lalu dilakukan *eXclusive OR* dengan plainteks untuk menghasilkan *Ciphertext* |
| *Counter* | CTR | Mode yang mana memiliki alat penghitung (*counter)* untuk memasukan *block set input* untuk menghasilan urutan *output* lalu dilakukan *eXclusive OR* dengan plain teks lalu menghasilkan Cipherteks |
| *Galois Counter Mode* | GCM | GCM memberikan keamanan data dengan menggunakan variasi dari mode perhitungan untuk enkripsi. |

Sumber: (Dunkerley, 2017)

## *Asymmetric Keys* (Kunci tidak simetris)

Pada *asymmetric keys* terdapat 2 kunci, yaitu kunci publik (Key A) yang biasa digunakan untuk enkripsi data dan kunci privat (Key B) untuk dekripsi data. Dapat dilihat pada Gambar 2.3.

Gambar 2.3

Diagram Kunci Asimetris

Sumber: (Dunkerley, 2017)

Meskipun kunci secara matematis mirip, mereka tidak dapat diturunkan satu sama lain. Dua kunci dibuat untuk enkripsi dan tujuan dekripsi: satu kunci adalah kunci publik, yang diketahui semua pengguna, sedangkan kunci pribadi tetap rahasia dan diberikan kepada pengguna untuk menjaga privasi. Untuk menggunakan sistem ini, pengirim akan mengenkripsi sebuah pesan atau file dengan kunci publik penerima yang dimaksud. Untuk mendekripsi pesan ini, penerima akan menggunakan sebuah kunci pribadi yang hanya dia miliki. Tidak ada orang lain yang dapat mendekripsi pesan tanpa kunci privat ini. Publik kunci dapat dilewatkan secara langsung di antara pengguna atau ditemukan di direktori kunci publik. Kunci ketiga digunakan untuk mengenkripsi kunci pribadi, yang kemudian disimpan di lokasi yang aman. Kunci ketiga ini dapat digunakan untuk membuka kunci salinan terenkripsi kunci pribadi dalam kasus kehilangan atau pencurian kunci asli.

## *Data Encryption Standard (DES)*

*Data Encryption Standard* (DES) merupakan algoritma *cipher block* *symmetric key* yang dirancang oleh Pemerintah Amerika Serikat pada tahun 1977 sebagai standar kriptografi yang diusulkan untuk semua komunikasi pemerintah. Karena dinyatakan tidak aman oleh Pemerintah Amerika Serikat, maka pemerintah federal Amerika Serikat mulai meninggalkan Algoritma DES dan digantikan menjadi *Advanced Encryption Standard* pada Desember 2001 (Madria & Hara, 2016).

Sebelum algoritma DES, *International Business Machines Corporation* (IBM) membuat algoritma yang bernama Lucifer yang memiliki panjang *key* sebanyak 128*bit*. Lalu kemudian selanjutnya *key* dipersingkat menjadi 56-*bit* dan diganti namanya menjadi *Data Encryption Standard* dan diadopsi oleh Pemerintah U.S. (Ciampa, 2015)

DES merupakan *block cipher* yang membagi plainteks menjadi 64-*bit* *block* dan digunakan sebagai *input* enkripsi yang dilakukan sebanyak 16 kali. Terdapat 4 model dari Algoritma DES yang masih digunakan. Walaupun algoritma DES pernah diterapkan secara luas, tetapi panjang kunci yang digunakan hanya 56*bit*, sehingga tidak lagi dianggap aman dan tidak direkomendasikan untuk digunakan.

DES menggunakan *series* panjang dari *eXclusive OR* (XOR) dalam operasionalnya untuk menghasilkan Cipherteks. Proses ini dilakukan selama 16 kali setiap enkripsi maupun dekripsi. (Stewart, Chapple, & Gibson, 2015)

## *International Data Encryptopn Algorithm*

*International Data Encryption Algorithm* (IDEA) merupakan algoritma *cipher* *block* *symmetric key* yang dirancang oleh Xuejia Lai dan James L. Massey dari ETH-Zürich dan pertama kali dideskripsikan pada tahun 1991. Ini adalah revisi minor dari *cipher* sebelumnya, PES (*Proposed Encryption Standard*). IDEA pada awalnya disebut IPES (*Improved PES*). IDEA digunakan sebagai *cipher* simetris dalam versi awal dari kriptosistem privasi cukup baik. (Schneier, 2015)

IDEA dibuat dengan tujuan agar dapat menggantikan algoritma enkripsi *Data Encryption Standard* yang dikembangkan di Amerika Serikat pada tahun 70-an. Berikut ini merupakan kelebihan dari algoritma IDEA (Mandal, Kar, Das, & Panigrahi , 2015):

1. Memberikan keamanan tingkat tinggi tidak berdasarkan menjaga kerahasiaan algoritma, tetapi lebih pada ketidaktahuan kunci rahasia
2. Sepenuhnya ditentukan dan mudah dipahami
3. Cocok untuk digunakan dalam berbagai aplikasi
4. Dapat diimplementasikan secara ekonomis dalam komponen elektronik (VLSI Chip)
5. Dapat diekspor ke seluruh dunia
6. Dilindungi paten untuk mencegah penipuan dan pembajakan

IDEA beroperasi pada blok 64-*bit* menggunakan kunci 128-*bit* dan terdiri dari seri delapan transformasi identik (putaran, lihat ilustrasi) dan output transformasi (setengah putaran). Proses untuk enkripsi dan dekripsi serupa. IDEA memperoleh banyak keamanannya dengan operasi *interleaving* dari grup yang berbeda, penambahan dan penggandaan modular, dan *bitwise eXclusive OR* (XOR) yang secara aljabar “tidak kompatibel” dalam arti tertentu. Lebih detail, operator ini, yang semuanya berhubungan dengan kuantitas 16-*bit* (Schneier, 2015).

## *Message Digest* – 5 (MD5)

*Message Digest* merupakan salah satu algoritma enkripsi *one-way­ hash* (hash satu arah) yang memiliki tiga jenis versi yang berbeda. *Message Digest 2* (MD2) merupakan algoritma enkripsi *Message Digest* yang pertama. *Message Digest* merubah seluruh plain teks yang panjangnya berbeda dan membuat panjangnya menjadi *digest* *128-bit*. MD2 membagi plainteks menjadi beberapa sector yang panjangnya 128-*bit*, jika plainteks ternyata kurang dari 128-*bit*, maka akan ada penambahan hingga mencapai 128-*bit*. (Madria & Hara, 2016)

MD2 pertama kali dibentuk pada tahun 1989 dan dioptimasi agar dapat berjalan pada prosesor intel mikrokomputer yang menjalankan *8-bit* per waktu. Sekarang MD2 sudah tidak begitu aman lagi, maka dibentuklah *Message Digest 4* (MD4) sebagai pengganti MD2 pada tahun 1990 untuk digunakan pada komputer yang dapat memproses 32-*bit* per waktu. Cara kerja *Message Digest 4* yaitu membuat digest 128-*bit*, dan plainteks dijadikan 512-*bit* dari yang sebelumnya (Algoritma MD2) 128-*bit*.

*Message Digest 5* (yang digunakan sekarang) merupakan revisi dari *Message Digest* 4 yang dibuat untuk memperbaiki kelemahan dari *Message Digest 4*. Cara Kerja MD5, Panjang dari pesan dibuat menjadi 512-*bit*, menggunakan empat variabel dari 32-*bit* setiap ronde dan menciptakan *value* yang telah terkompresi untuk menciptakan *digest.*

Berikut ini merupakan penjelasan cara algoritma *Message Digest 5* bekerja (Zu & Hu, 2016):

### *Data Filling*: Dengan menambahkan angka 1 dan angka 0 sesuai setelah pesan yang asli sampai dengan 64-*digit*.

###  *Adding Length:* Dengan menambahkan 64-*digit* setelah *Data Filling*. Sampai dengan panjang maksimal yaitu kelipatan dari 512-*digit*.

### *Initialization Variable:* Menggunakan 4 variabel (Misalkan A, B, C, D) untuk menghitung *Message Digest*. Dengan menginisialisasikan 4 variabel tersebut dengan *value* 32-*bit* *Hexadecimal* (A = 01234567h, B = 89abcdefh, C = fedcba98h, D = 76543210h)

### *Data processing*: Melalui 4 fungsi, memproses pesan yang dikelompokkan dalam blok 512-*byte*, 4 fungsi tambahan ditunjukkan di bawah ini:

#### F(X,Y,Z)=(X&Y)|((~X)&Z)

#### G(X,Y,Z)=(X&Z)|(Y&(~Z))

#### H(X,Y,Z)=X^Y^Z

#### I(X,Y,Z)=Y^(X|(`Z))

Keterangan Simbol:

#### Lambang & merupakan operator “*AND*”

#### Lambang | merupakan operator “*OR*”

#### Lambang ~ merupakan operator “*Nor*” atau “*NOT*”

#### Lambang ^ merupakan operator “*eXclusive OR*”

### Output: Setelah 512-*byte* block telah diproses, ABCD akan menjadi output sebagai hasil dari *hash MD5*.

## Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain merupakan editor kode IntelliJ dan alat *development* yang baik, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android (Google LLC, 2018), misalnya:

### Sistem versi berbasis *Gradle* yang fleksibel.

### *Emulator* yang cepat dan kaya fitur.

### Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.

### *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.

### Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor contoh kode.

### Alat pengujian dan kerangka kerja yang ekstensif.

### Alat pintas untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.

### Dukungan C++.

### Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

## Dalam pemasangan aplikasi Android Studio, memerlukan *System Requierment* untuk Android Studio dapat berjalan pada sistem operasi Microsoft Windows Sebagai Berikut (Yener & Dundar, 2016):

### Microsoft Windows 10/8.1/8/7/Vista/2003 (32/64 *bit*).

### Minimal 2GB RAM, dianjurkan 4GB RAM atau lebih.

### 400MB ruang penyimpanan.

### Setidaknya 1GB untuk Android SDK, *emulator system images*, dan *caches*.

### Resolusi layar minimal 1280 x 800

### *Java Runtime Environment (JRE) 6 or higher.*

### *Java Development Kit (JDK) 7*

### *Optional for accelerated emulator: Intel processor with support for Intel VT-x, Intel EM64T (Intel 64), and Execute Disable (XD) Bit functionality.*

Android SDK merupakan perangkat lunak yang sangat dibutuhkan dalam pembuatan aplikasi Android, karena Android SDK berisikan Kumpulan fitur, alat dalam *development* Android. Serta Versi Android OS yang digunakan. Dapat dilihat pada Gambar 2.4



Gambar 2.4

Android SDK

Sumber: (Yener & Dundar, 2016)

Jika dilihat dari *System Requirement* terdapat *Java Runtime Environment* dan *Java Development Kit* berarti Android Studio menggunakan Bahasa Pemograman *Object Oriented* Java dalam pembuatan aplikasi Android.

## Java

Java adalah salah satu Bahasa Pemograman yang berbasis *Object-Oriented*. Berarti Java menggunakan konsep dari penggunaan kembali dari objek pemograman. (Horton, Android Programming for Beginners, 2015). Sejak tahun 1990-an Java telah digunakan, berevolusi dan beraneka ragam. Java juga tidak hanya digunakan pada Android, tetapi juga dapat digunakan untuk *Virtual Machine*. Selain itu Java juga cepat dan mudah untuk digunakan dikarenakan Java memiliki *memory management* yang digunakan agar proses yang dilakukan pada aplikasi dapat berjalan dengan lebih cepat (Horton, Learning Java by Building Android Games, 2015).

Java memiliki kelebihan dari Bahasa pemograman C++ (Bahasa pemograman yang sering digunakan hingga sekarang) sebagai berikut (Cosmina, 2018):

### Keamanan.

Di Bahasa Pemograman Java, tidak masalah dalam membuat data palsu yang melebihi ukuran *array*.

### Otomatisasi Manajemen Memori.

Pengembang tidak perlu memikirkan memori yang harus dialokasi untuk sebuah objek, hal ini telah ditangani secara otomatis oleh Manajemen Memori Java.

### *Simplicity*.

Tidak ada *pointer*, penyatuan, *templates*, maupun struktur. Semua yang ada di Bahasa pemograman Java dapat di deklarasikan sebagai *class*.

### Mendukung *multithreaded execution*.

Java didesain dari perangkat lunak yang mendukung pengembangan dari *multithreaded software.*

### *Portability*.

Java dapat mendeklarasikan suatu variabel yang dapat digunakan dimanapun dan kapanpun dengan memasukan (*include*) data dari java yang lain.

## Sejarah Android

Pertumbuhan yang mengesankan OS (*Operating System*) Android awalnya dibuat oleh Andy Rubin untuk menjadi OS untuk ponsel. Ini terjadi sekitar awal abad ke-21. Pada bulan Juli 2005, Google mengakuisisi Android dan menjadikan Andy Rubin sebagai Wakil Presiden Senior di Google, di mana dia tetap menjabat hingga November 2014. Banyak yang merasakan akuisisi OS Android ini Google sebagian besar menanggapi penampilan iPhone Apple sekitar waktu yang sama. Namun, ada cukup banyak pemain besar lainnya, seperti RIM Blackberry, Nokia Symbian, dan Microsoft Windows Mobile, yang dianggap sebagai keputusan bisnis yang cerdas untuk Google untuk membeli bakat rekayasa Android *incorporated* bersama dengan OS Android-nya hak milik intelektual. Ini memungkinkan Google memasukkan perusahaan mesin pencari Internet mereka pasar ponsel yang sedang berkembang, yang sekarang banyak disebut sebagai Internet 2.0. (Jackson, 2017).

Android memiliki tujuh rilis utama yang masing-masing memiliki beberapa revisi kecil. Untuk mengikuti versi ini lebih mudah, pengembang beri nama dengan *cookie* nama-nama. Versi Android yang populer adalah Kitkat (Android 4.4), Lollipop (Android 5.1) dan Marshmallow (Android 6.0) Nougat (Android 7.0) juga mendapatkan popularitas. Android digunakan tidak hanya di smartphone tetapi juga di *tablet*, *netbook*, *Smart TV*, perangkat permainan genggam dan bahkan dalam satu komputer. Oleh karena itu kita harus terlebih dahulu memilih target *multiplatform* dan banyak versi sebelum mengembangkan aplikasi.

Karena Android merupakan Sistem Operasi yang *Open Source*, sehingga dapat dengan mudah dimodifikasi serta *support* berbagai pengaturan *hardware* maupun *software*. Dan mendukung berbagai fitur sebagai berikut (DiMarzio, 2017):

### *Storage.*

### *Connectivity.*

### *Messaging.*

### *Media Support.*

### *Hardware Support.*

### *Multi-touch.*

### *Multi-tasking.*

### *Thetering.*

## *Unified Modeling Language*

*Unified Modeling Languange* (Disingkat sebagai UML) merupakan Bahasa grafis yang secara resmi didefinisikan oleh *Object Management Group* untuk visualisasi, spesifikasi, konstruksi, dan dokumentasi dari artefak sebuah sistem *software*. Artefak dalam *development* perangkat lunak merupakan item yang dibuat atau dikumpulkan selama proses *development*.

Tujuan UML yaitu memberikan panduan bagaimana cara untuk membuat *blueprints* yang termasuk hal-hal yang konseptual seperti proses bisnis dan fungsi dari sebuah sistem. UML Juga bisa digunakan untuk modeling sebuah hardware dan merepresentasikan struktur organisasi. (Osis & Donins, 2017). UML memiliki berbagai jenis versi 1.1 diluncurkan pada Desember 1997 hingga yang terbaru yaitu UML versi 2.5.1 yang diluncurkan pada Desember 2017. (Object Management Group, 2017)

### *Use Case Diagram*

*Use Case Diagram* adalah diagram yang menunjukan kumpulan dari beberapa *use case*, *actors* dan hubungan antaranya. *Use Case Diagram* biasanya digunakan untuk mengorganisasi dan modeling kegiatan yang dibutuhkan dari sebuah sistem.

### *Activity Diagram*

*Activity Diagram* adalah diagram yang menampilkan model dinamis dari sistem yang berisikan urutan skema dari aktivitas pertama menuju aktivitas selanjutnya. Sebuah aktivitas menunjukan kumpulan tindakan yang berurutan atau bercabangan dari skema. *Activity Diagram* biasanya digunakan oleh organisasi dalam menggambarkan model bisnis, proses *engineering* dan alur kerja.

### Class Diagram

*Class diagram* adalah diagram yang paling sering ditemukan pada sistem yang berbasis objek dan digunakan untuk mengambarkan sudut pandang dari sebuah sistem. *Class* *diagram* menunjukan kumpulan kelas antarmuka dan hubungan antar kedua hal tersebut.

## Penelitian Terdahulu

Terdapat beberapa penelitian terdahulu yang berkaitan dengan penerapan metode *International Data Encryption Algorithm* untuk melakukan enkripsi dan dekripsi data / teks. Beberapa penelitian terdahulu yang telah dilakukan tersebut akan menjadi acuan dalam memperoleh informasi mengenai tema pembahasan penelitian yang akan dilakukan.

 Pada tahun 2016, David Miguel Lozano merupakan salah satu mahasiswa dari *Warsaw University of Technology* yang melakukan penelitian mengenai metode *International Data Encryption Algorithm* dengan judul *Software Implementation of International Data Encryption Algorithm (IDEA) Cipher With 4 Ciphering Modes*. Dalam penelitian tersebut, David Miguel Lozano menggunakan metode *Cipher Block Chaining, Cipher Feedback, Output Feedback, dan Electronic Codebook* pada sistem operasi windows. Dalam pendekripsian data oleh David Miguel dapat dilihat bahwa setiap metode *Cipher* yang digunakan memiliki hasil enkripsi dan dekripsi yang berbeda.

Pada tahun 2018, Muhammad Adi Wijaya, Wijaya Kurniawan, dan Ari Kusyanti melakukan penelitian mengenai penerapan metode *International Data Encryption Algorithm* untuk keamanan data transfer nirkabel pada *personal data assistant* dengan judul *Perancangan dan Implementasi Algoritma Enkripsi Idea pada Perangkat Kriptografi Berbasis FPGA*. Dalam penelitian tersebut Adi beserta rekan menggunakan metode *International Data Encryption Algorithm* untuk diimplementasikan ke sistem *personal data assistant* pada ponsel. Dalam hal kecepatan enkripsi data teks, suatu data teks dapat di enkripsi dengan waktu sebesar 7.788ns (nano sekon).

Penelitian tersebut akan penulis gunakan sebagai pedoman dalam pembuatan aplikasi enkripsi maupun dekripsi data berbasis Android, dimana penulis akan mempelajari bagaimana peneliti terdahulu menggunakan *International Data Encryption Algorithm* yang menurut penulis dapat diimplementasikan kedalam Android.

Hal yang akan penulis kembangkan untuk implementasi algoritma enkripsi *International Data Encryption Algorithm* dari penelitian terdahulu yaitu dengan melakukan implementasi algoritma tersebut pada sistem operasi Android dengan menggunakan satu model Cipher yaitu *Electronic Codebook*. Karena Android merupakan salah satu sistem operasi yang paling banyak digunakan pada *smartphone* pada saat peneliti melakukan penelitian ini.