# BAB II

# TINJAUAN PUSTAKA DAN DASAR TEORI

## 2.1 Tinjauan Pustaka

Penelitian sebelumnya yang ditulis oleh Alen Dwi Priyanto Here(2010) dibuat dengan menggunakan bahasa pemrograman Borland C++ Builder. Berupa aplikasi kriptografi menggunakan kombinasi metode *stream cipher* dan *viginere cipher* untuk pengamanan data.

Sedangkan dalam penelitian ini menggunakan metode *Shift Cipher*, *Stream Cipher*, ECB, dan kombinasinya sehingga kriptografinya lebih kuat lagi, dan bahasa pemrogrannya menggunakan Java.

## 2.2 Dasar Teori

### 2.2.1 Kriptografi

Kriptografi berasal dalam bahasa yunani dari kata "*crypto*" yang berarti rahasia dan "*graphia*" yang berarti tulisan. Kriptografi adalah suatu ilmu dan seni untuk menjaga keamanan tulisan atau pesan ketika pesan dikirim dari suatu tempat ke tempat lain ( Dony Ariyus, 2008).

Kriptografi klasik menitikberatkan kekuatan pada kerahasiaan algoritma yang digunakan (yang artinya apabila algoritma yang digunakan telah diketahui maka pesan sudah jelas "bocor" dan dapat diketahui isinya oleh siapa saja yang mengetahui algoritma tersebut), kriptografi modern lebih menitikberatkan pada kerahasiaan kunci yang digunakan pada algoritma tersebut (oleh pemakainya) sehingga algoritma tersebut dapat saja disebarkan ke kalangan masyarakat tanpa takut kehilangan kerahasiaan bagi para pemakainya.

Berikut adalah istilah-istilah yang digunakan dalam bidang kriptografi:

- *Plaintext* adalah pesan yang hendak dikirimkan (berisi data asli).
- *Ciphertext* adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
- Enkripsi adalah proses pengubahan *plaintext* menjadi *ciphertext*.
- Dekripsi adalah kebalikan dari enkripsi yakni mengubah *ciphertext* menjadi *plaintext*, sehingga berupa data awal/asli.
- Kunci adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.

Kriptografi itu sendiri terdiri dari dua proses utama yakni proses enkripsi dan proses dekripsi. Seperti yang telah dijelaskan di atas, proses enkripsi mengubah *plaintext* menjadi *ciphertext*

(dengan menggunakan kunci tertentu) sehingga isi informasi pada pesan tersebut sukar dimengerti.

Peranan kunci sangatlah penting dalam proses enkripsi dan dekripsi (disamping pula algoritma yang digunakan) sehingga kerahasiaannya sangatlah penting, apabila kerahasiaannya terbongkar, maka isi dari pesan dapat diketahui.

### 2.2.2 *Shift Cipher*

Algoritma *Shift Cipher* (Sandi geser) ini merupakan suatu algoritma kriptografi klasik. *Shift Cipher* merupakan generalisasi dari sandi *Caesar*, yaitu tidak membatasi pergeseran sebanyak tiga huruf. Teknik pergeseran akan dilakukan dengan menggunakan kode *American Standard Code for Information Interchange (*ASCII) dengan modulus 256 sehingga nantinya pembaca harus mengikuti tabel ASCII 256 (Cahyo Ardianto, 2012). Secara umum dapat dituliskan dengan persamaan berikut ini:

Proses enkripsi yaitu :

$$Ci = E(Pi) = (Pi + Ki) \bmod 256 \quad \text{..... (i)}$$

Proses dekripsi yaitu :

$$Pi = D(Ci) = (Ci - Ki) \bmod 256 \quad \text{..... (ii)}$$

Keterangan: Ci=karakter ke-i dari *ciphertext*, Pi= karakter ke-i dari *plaintext*, Ki= karakter ke-i dari kunci.

Contoh :

*Plaintext* : Dew! Kunci :@

**Enkripsi**

*Plaintext* : D:68, e:101, w:119, !:33 Kunci : @:64

C = (P + K) mod 256
D = (68 + 64) mod 256 = 132 → „ (Double low)
E = (101 + 64) mod 256 = 165 → ¥ (Yen sign)
W = (119 + 64) mod 256 = 183 → · (Middle dot-Georgian comma)
! = (33 + 64) mod 256 = 97 → a
*Cipertext* : „¥·a

**Dekripsi**

*Ciphertext* : „:132, ¥:165, ·: 183, a:97 Kunci : @:64

C = (P - K) mod 256
D = (132 - 64) mod 256 = 68 → D
E = (165 - 64) mod 256 = 101 → e
W = (183 - 64) mod 256 = 119 → w
! = (97 - 64) mod 256 = 33 → !
*Plaintext* : Dew!

### 2.2.3 *Stream Cipher*

*Stream Cipher* algoritma yang dalam operasinya bekerja dalam suatu pesan berupa bit tunggal atau terkadang dalam

suatu *byte*, jadi format data berupa aliran dari bit untuk kemudian mengalami proses enkripsi dan dekripsi. *Stream chiper* merupakan suatu teknik enkripsi data dengan cara melakukan transformasi dari tiap bit secara terpisah berdasarkan posisi tiap bit dalam aliran data yang biasanya dikendalikan menggunakan operasi XOR. Enkripsi aliran data merupakan hasil dari operasi XOR antara setiap bit *plaintext* dengan setiap bit kuncinya (Cahyo Ardianto, 2012).

Maka persamaan enkripsi di tulis:

$$Ci = Pi \oplus Ki \quad \text{..... (iii)}$$

Dan proses dekripsi menggunakan persamaan:

$$Pi = Ci \oplus Ki \quad \text{..... (vi)}$$

Contoh :

*Plaintext* : Lia

**Enkripsi**

*Plaintext* : L :76, i:105, a:97 Kunci : 3, 76, 105

*Plaintext* : L
76 = 0100 1100
3 = 0000 0011 ⊕
0100 1111 =79 →O

Plaintext : i
105 = 0110 1001
76 = 0100 1100 ⊕
0010 0101 = 37 →%

*Plaintext* : a
97 = 0110 0001
105 = 0110 1001 ⊕
0000 1000 = 4 →EOT (End of Transmission)

*Ciphertext* : O%EOT

**Dekripsi**

*Ciphertext* : O: 79, %: 37, EOT:4 Kunci : 3, 76, 105

*Ciphertext* : O

79 = 0100 1111

3 = 0000 0011 ⊕

0100 1100 = 76 → L

*Ciphertext* : %

37 = 0010 0101

76 = 0100 1100 ⊕

0110 1001 =105 → i

*Ciphertext* : EOT

4 = 0000 1000

105 = 0110 1001 ⊕

0110 0001 = 97 → a

*Ciphertext* = Lia

### 2.2.4 *Electronic Codebook* (ECB)

Pada metode ECB ini suatu blok kode yang panjang dibagi dalam bentuk ukuran binari menjadi satu blok tanpa mempengaruhi blok-blok lain. Satu blok terdiri dari 64 bit atau 128 bit. Setiap blok merupakan bagian dari pesan yang dienkripsi. Kata *code book* di dalam ECB muncul dari fakta bahwa blok teks-asli yang sama selalu dienkripsi menjadi blok teks-kode yang sama maka secara teoretis dimungkinkan untuk membuat buku kode teks-asli dan teks-kode yang berkorespondensi. Namun semakin besar ukuran blok, semakin besar pula ukuran buku kodenya (Dony Ariyus, 2008).

Maka persamaan enkripsi di tulis:

$$Ci = Ek ( Pi ) \quad \text{..... (v)}$$

Dan proses dekripsi menggunakan persamaan:

Pi = Dk ( Ci ) ..... (vi)

Contoh :

*Plaintext* : Dew! Kunci :Li@

**Enkripsi**

*Plaintext* : D:68, e:101, w:119, !:33

Kunci : L:76, i:105, @:64

*Plaintext* : D

68 = 0100 0100

76 = 0100 1100 ⊕

8 = 0000 1000

105 = 0110 1001 ⊕

97 = 0110 0001

64 = 0100 0000 ⊕

33 = 0010 0001

→ digeser ke kiri 1 bit

0100 0010 = 66 → B

*Plaintext* : e

101 = 0110 0101

76 = 0100 1100 ⊕

41 = 0010 1001

105 = 0110 1001 ⊕

64 = 0100 0000

64 = 0100 0000 ⊕

0 = 0000 0000

→ digeser ke kiri 1 bit

0000 0000 = 0 → NUL

*Plaintext*: w

119 = 0111 0111

76 = 0100 1100 ⊕

59 = 0011 1011

105 = 0110 1001 ⊕

82 = 0101 0010

64 = 0100 0000 ⊕

18 = 0001 0010

→ digeser ke kiri 1 bit

0010 0100 = 36 → $

*Plaintext*: !

33 = 0010 0001

76 = 0100 1100 ⊕

109 = 0110 1101

105 = 0110 1001 ⊕

4 = 0000 0100

64 = 0100 0000 ⊕

68 = 0100 0100

→ digeser ke kiri 1 bit

1000 1000 = 136 → ^

*Ciphertext* : BNUL$^

**Dekripsi**

*Ciphertext* : B: 66, NUL: 0, $: 36, ^:136

Kunci : @:64, i:105, L:76

*Ciphertext* B

0100 0010 = 66

→ digeser ke kekanan 1 bit

33 = 0010 0001

64 = 0100 0000 ⊕

97 = 0110 0001

105 = 0110 1001 ⊕

8 = 0000 1000

76 = 0100 1100 ⊕

68 = 0100 0100 → D

*Ciphertext* NUL

0000 0000 = 0

→ digeser ke kekanan 1 bit

0 = 0000 0000

64 = 0100 0000 ⊕

64 = 0100 0000

105 = 0110 1001 ⊕

41 = 0010 1001

76 = 0100 1100 ⊕

101 = 0110 0101 → e

Ciphertext $

0010 0100 = 36

→ digeser ke kekanan 1 bit

18 = 0001 0010

64 = 0100 0000 ⊕

82 = 0101 0010

105 = 0110 1001 ⊕

59 = 0011 1011

76 = 0100 1100 ⊕

119 = 0111 0111 → w

Ciphertext ^

1000 1000 = 136

→ digeser ke kekanan 1 bit

68 = 0100 0100

64 = 0100 0000 ⊕

4 = 0000 0100

105 = 0110 1001 ⊕

109 = 0110 1101

76 = 0100 1100 ⊕

33 = 0010 0001 → !

*Plaintext* : Dew!