BAB II

LANDASAN TEORI

2.1. Kriptografi

Kata kriptografi berasal dari bahasa Yunani, *crypto* (rahasia) dan *graphia* (tulisan). Secara umum makna dari kata kriptografi adalah tulisan rahasia, dan arti sebenarnya dari kriptografi adalah suatu metode untuk mengacak data (enkrip) dengan teknik atau metode algoritma tertentu sehingga data tidak bisa dibaca secara normal, dan merapikannya kembali seperti semula (dekrip) dengan kunci yang tepat (Putranto, 2010).

Jadi, secara khusus kriptografi merupakan teknik pengamanan informasi yang dilakukan dengan cara mengolah informasi awal (*plaintext*) dengan suatu kunci tertentu melalui proses enkripsi sehingga menghasilkan suatu informasi baru (*ciphertext*) yang tidak dapat dibaca secara langsung. Informasi baru (*ciphertext*) tersebut dapat dikembalikan menjadi informasi awal (*plaintext*) melalui proses dekripsi (Rambe, 2010).

Algoritma kriptografi, atau sering disebut *cipher*, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Konsep algoritma kriptografi berdasarkan kuncinya secara umum dibagi menjadi dua bagian, yaitu algoritma kriptografi kunci rahasia dan algoritma kriptografi kunci publik (Stiawan, 2005).

Sebuah sistem kriptografi (*cryptosystem*) dapat digambarkan dengan lima *tuple* (P, C, K, E, D), di mana kondisi berikut terpenuhi (Stinson, 2006):

1) P adalah himpunan berhingga dari *plaintext* yang mungkin,

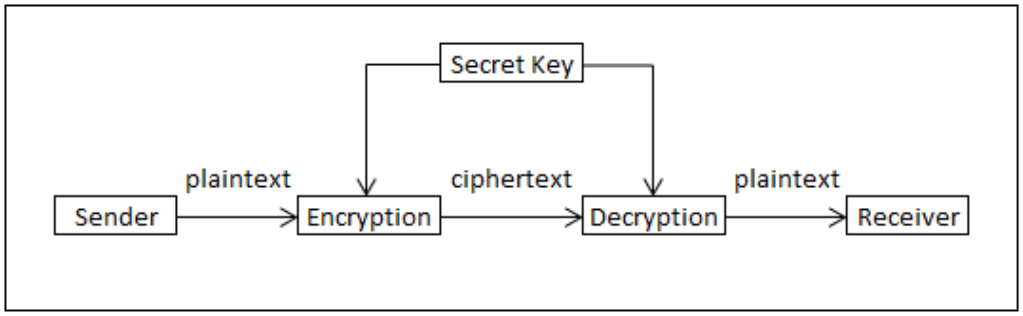
2) C adalah himpunan berhingga dari *ciphertext* yang mungkin,

3) K adalah *keyspace*, yaitu himpunan berhingga dari *key* yang mungkin,

4) Untuk setiap k ∈ K, terdapat aturan enkripsi e_k ∈ E dan aturan dekripsi yang bersesuaian d_k ∈ D. Setiap e_k: P→ C dan d_k: C → P adalah fungsi sehingga $d_k(e_k(x)) = x$ untuk setiap elemen *plaintext* x ∈ P.

2.1.1. Algoritma Kunci Rahasia

Algoritma kunci rahasia dikenal juga dengan kriptografi simetris (*symmetric cryptography*), karena proses enkripsi dan dekripsinya hanya menggunakan satu kunci yang unik. Oleh karena itu, antara pengirim dan penerima saling berbagi kunci yang harus dijaga kerahasiaannya (Arregoces & Portolani, 2004).



Gambar 2.1. *Symmetric Cryptography* (Arregoces & Portolani, 2004)

Karena kesamaan kunci untuk melakukan enkripsi dan dekripsi, maka sebelum berkomunikasi harus ditentukan metode yang aman untuk bertukar kunci. Setiap orang yang mengetahui kunci dan algoritma dapat melakukan dekripsi untuk pesan yang di enkripsi dengan kunci tersebut. Kriptografi simetris biasanya sangat cepat, namun memiliki beberapa kekurangan (Amon, 2004).

Menurut Ariyus (2008), algoritma yang menggunakan konsep kriptografi simetris diantaranya adalah:

1) *Data Encryption Standard* (DES),
2) RC2, RC4, RC5, RC6,
3) *International Data Encryption Algorithm* (IDEA),
4) *Advanced Encryption Standard* (AES),
5) *One Time Pad* (OTP),
6) A5, dan lain sebagainya.

Menezes dkk (1997), menjabarkan kelebihan dan kelemahan kriptografi simetris sebagai berikut.

Kelebihan:

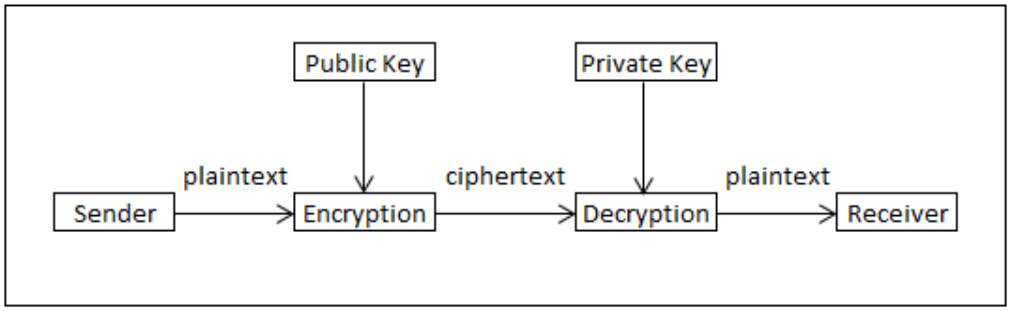
1) Dapat dirancang untuk memiliki kecepatan proses (enkripsi / dekripsi) yang tinggi.
2) Ukuran kunci relatif pendek.
3) Dapat digunakan sebagai dasar untuk membangun mekanisme kriptografi yang lain.
4) Dapat digunakan untuk menghasilkan algoritma kriptografi (*cipher*) lain yang lebih kuat.
5) Dipercaya memiliki nilai sejarah yang tinggi.

Kelemahan:

1) Kunci harus selalu bersifat rahasia.
2) Untuk komunikasi yang lebih luas, dibutuhkan lebih banyak kunci pula.
3) Kunci harus sering diubah, bahkan mungkin pada setiap kali berkomunikasi.
4) Untuk *digital signature* membutuhkan ukuran kunci yang besar.

2.1.2. Algoritma Kunci Publik

Algoritma kunci publik dikenal juga dengan kriptografi asimetris (*asymetric cryptography*), karena menggunakan dua kunci yang berbeda untuk proses enkripsi dan dekripsinya. Satu kunci disebut kunci pribadi / *private key* dan dijaga kerahasiaannya oleh pengirim. Kunci lainnya disebut kunci publik / *public key* dan diberitahukan kepada semua orang yang ingin berkomunikasi dengan pengirim. Secara umum, kripotgrafi asimetris lebih lambat daripada kriptografi simetris pada saat melakukan enkripsi. Sebaliknya, kripotgrafi asimetris lebih baik dalam sisi pendistribusian kunci (Arregoces & Portolani, 2004).



Gambar 2.2. *Asymmetric Cryptography* (Arregoces & Portolani, 2004)

Algoritma kunci publik dapat diimplementasikan dengan dua cara. Pertama, jika pesan dienkripsi dengan menggunakan kunci publik, maka akan diperoleh fungsi dari banyak ke satu. Hanya yang memiliki kunci pribadi yang dapat melakukan proses dekripsi. Kedua, jika pesan dienkripsi dengan menggunakan kunci pribadi, maka akan diperoleh fungsi dari satu ke banyak. Setiap orang yang memiliki kunci publik dapat melakukan proses dekripsi (Komputer, 2010).

Menurut Ariyus (2008), algoritma yang menggunakan konsep kriptografi asimetris diantaranya adalah:

1) *Digital Signature Algorithm* (DSA),

2) RSA,
3) Diffie-Hellman (DH),
4) *Elliptic Curve Cryptography* (ECC),
5) Kriptografi Quantum, dan sebagainya.

Menezes dkk (1997), menjabarkan kelebihan dan kelemahan kriptografi asimetris sebagai berikut.

Kelebihan:

1) Hanya kunci pribadi / *private key* yang harus dijaga kerahasiaannya (namun, otentikasi kunci publik / *public key* tetap harus terjamin)
2) Pasangan kunci publik dan kunci pribadi tidak harus diubah, bahkan untuk jangka waktu yang cukup panjang.
3) Untuk *digital signature* membutuhkan ukuran kunci yang lebih kecil daripada kriptografi simetris.
4) Untuk komunikasi yang lebih luas, jumlah kunci mungkin lebih sedikit daripada kriptografi simetris.

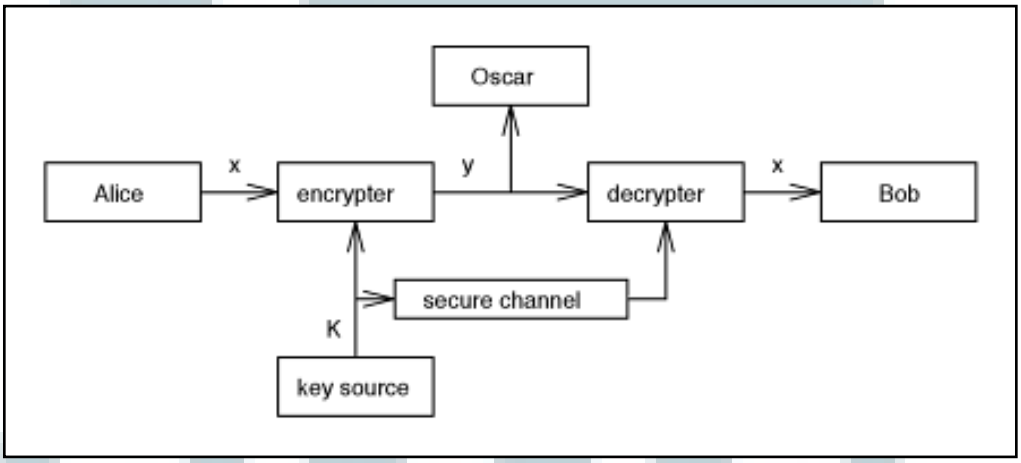
Kelemahan:

1) Kecepatan proses (enkripsi / dekripsi) lebih lambat daripada kriptografi simetris.
2) Ukuran kunci lebih besar.
3) Tidak ada kunci publik yang terbukti aman. Akan tetapi dapat diperkuat dengan sulitnya memecahkan persoalan aritmetik yang menjadi dasar pembentukan kunci publik.

2.1.3. Tujuan Kriptografi

Tujuan utama dari kriptografi adalah untuk memungkinkan dua orang berkomunikasi melalui kanal yang aman, sehingga musuh tidak mengerti apa yang sedang dikatakan (Stinson, 2006).

Informasi yang dikirimkan (*plaintext*) tersebut dienkripsi dengan menggunakan kunci yang telah ditentukan, kemudian hasil enkripsi (*ciphertext*) tersebut dikirim melalui kanal. Musuh mungkin mengetahui bahwa terdapat informasi yang sedang dikirim, namun itu adalah *ciphertext*. Musuh tidak dapat mengetahui isi informasi tersebut. Hanya yang memegang kunci enkripsi yang dapat melakukan dekripsi pada *ciphertext* dan mengetahui isi informasi (*plaintext*) yang dikirimkan (Stinson, 2006).



Gambar 2.3. Kanal Komunikasi (Stinson, 2006)

Dari Gambar 2.3. terlihat bahwa dengan kriptografi memungkinkan Alice untuk berkomunikasi dengan Bob melalui sebuah kanal yang aman dengan pesan yang sudah terenkripsi. Meskipun Oscar mengetahui bahwa ada pesan dari Alice kepada Bob, Oscar tidak dapat mengetahui isi pesan tersebut karena tidak

memiliki kunci untuk mendekripsikan pesan tersebut. Hanya Bob yang dapat mengetahui isi dari pesan yang dikirimkan Alice.

Maka, menurut Menezes dkk (1997), tujuan umum kriptografi dibagi menjadi 4 bagian berikut.

1) Kerahasiaan (*confidentiality*), adalah sebuah layanan yang digunakan untuk menjaga isi informasi dari siapapun, kecuali yang memiliki kewenangan untuk memilikinya.
2) Integritas data (*data integrity*), adalah sebuah layanan penjagaan pada perubahan data yang tidak sah. Untuk menjaga integritas data, maka harus dimiliki kemampuan untuk mendeteksi manipulasi data oleh pihak yang tidak memiliki kewenangan.
3) Autentikasi (*Authentication*), adalah sebuah layanan yang berhubungan dengan identifikasi. Ini berlaku baik untuk entitas dan informasi itu sendiri. Kedua pihak yang berkomunikasi harus saling mengidentifikasi satu sama lain. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi, waktu pengiriman, dan lain-lain.
4) *Non-repudiation*, adalah sebuah layanan yang mencegah adanya penyangkalan terhadap aksi yang telah dilakukan. Ini berlaku ketika adanya penyangkalan salah satu pihak telah melakukan sebuah aksi.

2.2. Algoritma Kriptografi ElGamal

ElGamal adalah algoritma kriptografi yang ditemukan oleh ilmuwan Mesir, yaitu Taher ElGamal pada tahun 1985. Algoritma ElGamal mendasarkan kekuatannya pada fakta matematis sulitnya menghitung logaritma diskrit.

Algoritma ElGamal terdiri atas tiga proses, yaitu proses pembentukan kunci, proses enkripsi, dan proses dekripsi. Kunci yang dibentuk dengan algoritma ElGamal berupa tiga bilangan untuk proses enkripsi dan satu bilangan untuk proses dekripsi (Rochmat dkk, 2012).

Logaritma ini disebut logaritma diskrit karena nilainya berhingga dan bergantung pada bilangan prima yang digunakan. Karena bilangan prima yang digunakan adalah bilangan prima besar, maka sangat sulit bahkan tidak mungkin menurunkan kunci privat dari kunci publik yang diketahui walaupun serangan dilakukan dengan menggunakan sumber daya komputer yang sangat besar (Zelvina dkk, 2012).

Proses pembentukan kunci pada ElGamal mengadopsi algoritma kunci publik Diffie-Hellman. Pada algoritma Diffie-Hellman, masing-masing pihak harus melakukan perhitungan untuk membuat kunci private. Hal ini akan sulit dilakukan mengingat banyaknya hambatan yang mungkin terjadi. Algoritma ElGamal dibuat untuk memperbaiki permasalahan tersebut, dimana tidak membutuhkan interaksi dari kedua pihak yang akan berkomunikasi (Meier, 2005).

Kelebihan dari algoritma ElGamal dapat dijabarkan sebagai berikut (Zelvina dkk, 2012).

1) Plainteks yang sama dapat diubah menjadi chiperteks yang berbeda, karena bilangan bulat pada algoritma Elgamal dapat dipilih secara acak untuk menentukan kunci.
2) Pada algoritma ElGamal tidak hanya kunci privat yang perlu dijamin kerahasiannya, tetapi autentikasi kunci publik juga harus tetap dijaga.

3) Kunci publik dan kunci privat pada algoritma ElGamal tidak perlu diubah dalam periode waktu yang panjang.
4) Algoritma ElGamal bisa dimanfaatkan untuk mengirimkan sebuah pesan rahasia, yaitu dengan menentukan kunci dari sebuah kriptografi simetris.

Kelemahan dari algoritma ElGamal dapat dijabarkan sebagai berikut (Caroline, 2011).

1) Memiliki panjang *ciphertext* dua kali panjang *plaintext*-nya.
2) Membutuhkan resource yang baik dan processor yang mampu untuk melakukan komputasi yang besar.

2.2.1. Cara Kerja Algoritma ElGamal

Menurut Buchmann (2004), terdapat 3 proses untuk menjalankan algoritma ElGamal. Proses tersebut dapat dijabarkan sebagai berikut.

```
ElGamal_Key Generation
{
Select a prime p
Select d such that 1 ≤ d ≥ p-2.
Select e1 to be prime root of p
 e2←e1^d mod p
Public_key← (e1, e2 , p)
Private_key← d
Return Public_key and Private_key
}

ElGamal_Encryption(e1, e2, p, P)
{
Select random number r in the group G=<Zp*, x>
// P is the plain Text.
C1 ← e1^r mod p          // C1 and C2 are Ciphertexts
C2 ← (P x e2^r) mod p
return // C1 and C2
}

ElGamal_Decryption
{
P ← [C2 (C1^d)^-1 ] mod p
return P
}
```

Gambar 2.4. *Pseudocode* Algoritma Elgamal (Suresha & Nalini, 2011)

1) Proses pembuat kunci dengan memilih sembarang bilangan prima p. kemudian pilih dua buah bilangan acak, g dan x yang nilainya kurang dari nilai p. Hitung $y = g^x \bmod p$. Kunci publik adalah p, g, dan y. Kunci rahasia adalah x.

2) Proses enkripsi merupakan proses mengubah pesan asli (plaintext) menjadi pesan rahasia (*ciphertext*). Pada proses ini digunakan kunci publik (p, g, y).
   a) Potong *plaintext* menjadi blok-blok m1, m2, …, nilai setiap blok sama dalam rentang [0, …, p – 1].
   b) Pilih bilangan acak k, dengan syarat $1 \leq k \leq p - 2$.
   c) Setiap blok di enkripsi dengan rumus:

      $e1 = g^k \bmod p$

      $e2 = y^k m \bmod p$
   d) Pasangan e1 dan e2 adalah *ciphertext* untuk blok pesan m. Jadi, ukuran *ciphertext* dua kali ukuran *plaintext*.
3) Proses dekripsi merupakan proses mengubah pesan rahasia (*ciphertext*) menjadi pesan asli (*plaintext*). Pada proses ini digunakan kunci pribadi (x).
   a) Hitung *plaintext* m dengan rumus:

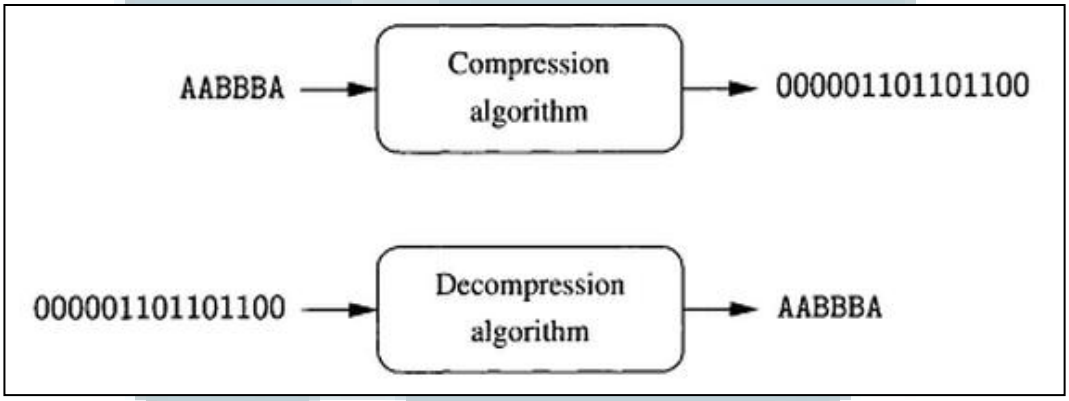
      $m = e2.e1^{(p-1-x)} \bmod p$
   b) Susun *plaintext* m.

2.3. Kompresi

Secara umum, kompresi merupakan proses mereduksi ukuran suatu data untuk menghasilkan representasi digital yang padat atau mampat (*compact*) namun tetap dapat mewakili kuantitas informasi yang terkandung pada data tersebut. Tujuan dari kompresi tiada lain adalah untuk mengurangi data berlebih sehingga ukuran data menjadi lebih kecil dan lebih ringan dalam proses transmisi (Putra, 2010).

2.3.1. Kompresi *Lossless*

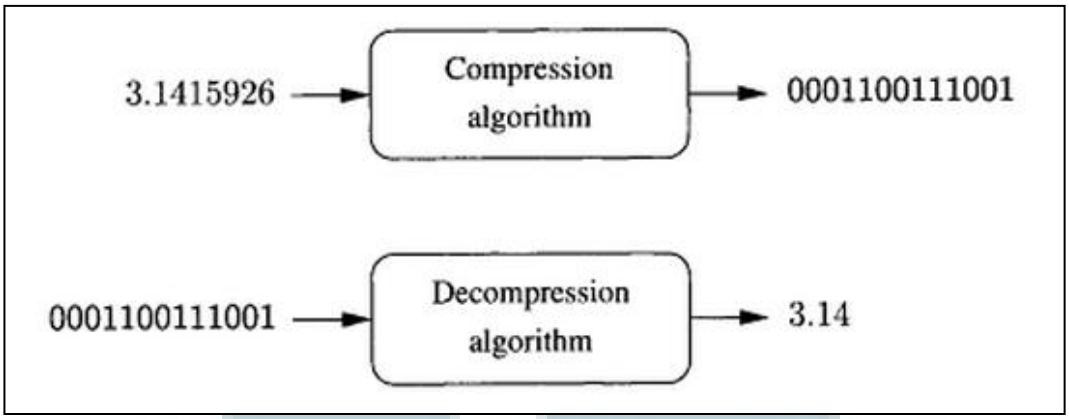
Kompresi *lossless* merupakan teknik kompresi yang memungkinkan data tersusun kembali dari data kompresi, dengan sama persis (tidak ada data yang dihilangkan) seperti data yang asli. Karena data dapat tersusun kembali persis seperti data yang asli, maka kompresi *lossless* dapat disebut juga kompresi *reversible*. Kompresi *lossless* digunakan ketika keaslian data dirasakan sangat penting dan tidak dapat kehilangan kehilangan data (Pu, 2006).



Gambar 2.5. Kompresi *Lossless* (Pu, 2006)

2.3.2. Kompresi *Lossy*

Kompresi *lossy* merupakan teknik kompresi yang memungkinkan data tersusun kembali dari data kompresi, tetapi tidak sama persis (ada data yang dihilangkan) seperti data yang asli. Karena data tidak dapat tersusun kembali persis seperti data yang asli, maka kompresi *lossy* dapat disebut juga kompresi *ireversible*. Kompresi *lossy* lebih efektif dibandingkan dengan kompresi *lossless* karena keaslian data dapat diatur sesuai dengan keinginan, meskipun dibutuhkan keseimbangan antara kualitas data dan sulitnya perhitungan (Pu, 2006).



Gambar 2.6. Kompresi *Lossy* (Pu, 2006)

2.4. Algoritma Kompresi Lempel-Ziv-Welch (LZW)

Lempel-Ziv-Welch atau yang biasa disebut LZW ini adalah satu dari banyak algoritma kompresi. Pada awalnya algoritma kompresi ini ditemukan oleh Abraham Lempel dan Jakob Ziv pada tahun 1977, yang kemudian dikenal dengan LZ77. Pada tahun berikutnya, yakni 1978, Abraham Lempel dan Jakob Ziv kembali membuat algoritma kompresi yang dikenal dengan LZ78. Pada tahun 1984, Terry Welch mengembangkan algoritma LZ78 sehingga menghasilkan algoritma LZW yang dikenal hingga saat ini (Mulyanta, 2009).

LZW sama dengan algoritma kompresi berbasis kamus lainnya seperti Huffman, dimana urutan nilai dalam aliran data yang mengulang lebih dari sekali dieksploitasi untuk mengompres data (Terras, 2008). Kamus atau sering juga disebut table *string* (*string table*) digunakan untuk menyimpan pola-pola *string*. Kamus ini akan diperbarui selama proses kompresi dan dekompresi. Isi kamus bervariasi menurut *input* teks yang dikompresi (Putra, 2010).

Walaupun implementasi LZW begitu rumit, namun algoritmanya begitu sederhana. Konsepnya adalah mengganti *string* pada suatu karakter dengan *code word* tunggal yang tersimpan pada kamus. Sebagian besar implementasi dari LZW

menggunakan 12-bit *code word* untuk mempresentasikan 8-bit karakter *input*. Pada kamus (tabel *string*) terdapat 4096 lokasi unik (kombinasi dari 12-bit). 256 lokasi pertama digunakan untuk menyimpan karakter tunggal. Kombinasi baru yang terdapat pada deretan *input* akan ditambahkan ke dalam kamus dan akan disimpan pada lokasi 256 sampai 4096 (Putra, 2010).

2.4.1. Cara Kerja Algoritma LZW

Berikut adalah penjabaran cara kerja algoritma kompresi LZW.

1) Proses kompresi merupakan proses mengonversikan sebuah data menjadi sebuah data lain dengan ukuran data yang lebih kecil. Pada proses ini, cara kerja algoritma LZW adalah seperti berikut (Salomon, 2002).
   a) Membaca parameter dari *input*, gabungkan dengan parameter selanjutnya.
      - Jika belum ada di dalam kamus, masukkan gabungan parameter tersebut. Kemudian ulangi langkah a.
      - Jika sudah ada, gabungkan dengan parameter selanjutnya.
   b) Dilakukan hingga selesai, lalu akan dibuat data yang baru berdasarkan kamus yang sudah terbuat.

```
word ← ''
while not EOF do
  x ← read_next_character()
  if word + x is in the dictionary then
    word ← word + x
  else
    output the dictionary index for word
    add word + x to the dictionary
    word ← x
  end if
end while
output the dictionary index for word
```

Gambar 2.7. *Pseudocode* Kompresi Algoritma LZW (Pu, 2006)

2) Proses dekompresi merupakan proses mengonversikan data yang sudah dikompresi menjadi data yang semula. Pada proses ini, data hasil kompresi akan berperan menjadi *input*. Cara kerja algoritma LZW pada saat dekompresi adalah seperti berikut (Pu, 2006).
   a) Membaca parameter dari *input*.
   b) Membuat data baru yang sama dengan semula berdasarkan data yang ada pada kamus.

```
read a token x from the compressed file
look up dictionary for element at x
output element
word ← element
while not EOF do
  read x
  look up dictionary for element at x
  if there is no entry yet for index x then
    element ← word + firstCharOfWord
  end if
  output element
  add word + firstCharOfElement to the dictionary
  word ← element
end while
```

Gambar 2.8. *Pseudocode* Dekompresi Algoritma LZW (Pu, 2006)

2.5. File Input

Dokumen-dokumen digital yang dapat menggunakan aplikasi ini adalah sebagai berikut.

2.5.1. *File* Teks (.txt)

Teks adalah kumpulan dari karakter-karakter atau *string* yang menjadi satu kesatuan. *File* teks merupakan *file* yang berisi informasi-informasi dalam bentuk teks. Data yang berasal dari dokumen pengolah kata, angka yang digunakan dalam

perhitungan, nama dan alamat dalam basis data merupakan contoh masukan data teks yang terdiri dari karakter, angka dan tanda baca (Pramilo, 2008).

Secara umum, format data teks dapat dibagi menjadi dua bagian sebagai berikut (Pramilo, 2008).

1. Teks terformat (*formatted text*)

   Merupakan teks yang terformat dan mengandung *styles*. Format data dokumen Microsoft Word (*.doc) merupakan contoh format data jenis ini yang paling popular.

2. Teks sederhana (*plain text*)

   Merupakan teks yang tidak mengandung *style*. Format data teks (*.txt) merupakan contoh format data jenis ini yang paling populer.

Contoh format data teks beserta perangkat lunak yang biasa digunakan diantaranya adalah sebagai berikut (Pramilo, 2008)

1. *Rich Text Format* (*.rtf)

   Format data teks ini dikembangkan oleh Microsoft dan dapat dibaca oleh berbagai macam platform seperti Windows, Linux, Mac OS, dan sebagainya.

2. Format data teks (*.txt)

   Format data teks ini merupakan format teks yang digunakan untuk menyimpan huruf, angka, karakter, kontrol (tabulasi, pindah baris, dan sebagainya) atau simbol-simbol lain yang biasa digunakan dalam tulisan. Satu huruf, angka, karakter, kontrol atau simbol pada teks memakan tempat satu byte. Berbeda dengan jenis teks terformat dimana satu huruf saja dapat memakan tempat beberapa byte untuk menyimpan format dari huruf tersebut seperti *font*, ukuran, ketebalan *font*, dan sebagainya. Kelebihan dari format data ini adalah

ukuran datanya yang kecil karena tidak adanya fitur untuk memformat tampilan teks. Saat ini perangkat lunak yang paling banyak digunakan untuk memanipulasi format data ini adalah Notepad.

3. *Hyper Text Markup Language* (*.html atau *.htm)

   Merupakan format data standard untuk tampilan dokumen web.

4. Format data dokumen (*.doc)

   Doc merupakan ekstensi dokumen perangkat lunak Microsoft Word yang paling banyak digunakan dalam penulisan laporan, makalah, dan sebagainya. Doc merupakan jenis teks terformat yang tidak hanya dapat mengatur tampilan teks seperti *styles* (*font*, ukuran huruf, dan sebagainya), namun juga dapat menyisipkan gambar. Kekurangan format data ini terletak pada ukuran datanya yang besar.

2.5.2. *Portable Document Format* (.pdf)

International Organization for Standardization (ISO). saat ini tengah menjadi pusat perhatian. Berbagai standar untuk pembuatan dan pertukaran dokumen yang dibuat dari aplikasi perkantoran telah banyak dibuat. Dengan standar tersebut, memungkinkan dokumen untuk dibuat oleh satu aplikasi dan dapat dimengerti oleh aplikasi lain. Sebagai contoh: suatu laporan bisnis yang dibuat dengan menggunakan Microsoft Word di atas sistem operasi Windows, dimungkinkan untuk dirubah oleh orang lain yang menggunakan Pages pada Apple Mac dan kemudian finalisasi dari dokumen tersebut dilakukan oleh seseorang yang menggunakan Open Office Writer di atas Linux (RISTEK, 2009).

Dari berbagai format dokumen tersebut, terdapat format yang sudah diadopsi sebagai standar internasional oleh ISO untuk format *file* pertukaran dokumen yaitu *Portable Document Format* (PDF), adalah sebuah format *file* bentuk digital yang diciptakan oleh Adobe Systems pada tahun 1993. Digunakan untuk membaca dan pertukaran dokumen elektronik. PDF bersifat independen terhadap aplikasi perangkat lunak, perangkat keras, dan sistem operasi. PDF sebagai standar terbuka secara resmi dipublikasikan oleh ISO pada 1 Juli 2008 dengan kode ISO 32000-1:2008. Sedangkan untuk format dokumen yang bisa diedit adalah terdapat dua standar, yaitu Open Document Format dan Office Open Extensible Mark Up Language (OOXML) (RISTEK, 2009).

2.5.3. Microsoft Office

Format *file* .doc, .docx, .xls, .xlsx, .ppt, dan .pptx adalah format untuk dokumen dari Microsoft Office. Untuk .doc, .xls, dan .ppt adalah dokumen digital dari Microsoft Office 97-2003. Sedangkan .docx, .xlsx, dan .pptx adalah dokumen digital dari Microsoft Office 2007. Huruf "x" menandakan bahwa dokumen tersebut tersimpan dengan format Open XML (office.microsoft.com, tanpa tahun).

Berikut adalah manfaat dari format Open XML (office.microsoft.com, tanpa tahun):

1. File ringkas. File dipadatkan secara otomatis dan pada beberapa kasus dapat menjadi 75 persen lebih kecil. Format Open XML menggunakan teknologi pemadatan zip untuk menyimpan dokumen, menawarkan potensi penghematan biaya karena mengurangi *disk space* yang dibutuhkan untuk menyimpan file, dan mengurangi bandwidth yang diperlukan untuk

mengirimkan file melalui email, lewat jaringan, dan melintasi internet. Saat membuka, *file* akan di-*unzip* secara otomatis. Saat menyimpan, maka akan dipadatkan lagi secara otomatis.

2. Pemulihan file rusak dengan lebih baik. File disusun secara modular sehingga menyimpan berbagai komponen data berbeda dalam file yang terpisah satu sama lain. Hal ini memungkinkan file dibuka bahkan jika komponen dalam file (misalnya, *chart* atau tabel) rusak.
3. Privasi dan kontrol yang lebih baik terhadap informasi pribadi. Dokumen dapat dibagikan secara rahasia, karena informasi pribadi dan informasi bisnis, seperti nama penulis, komentar, perubahan, dan jalur file dapat diidentifikasi dan dihapus dengan mudah dengan menggunakan *Document Inspector*.
4. Integrasi dan interoperabilitas data bisnis yang lebih baik. Menggunakan Format Open XML sebagai kerangka kerja interoperabilitas data untuk rangkaian produk Office berarti bahwa dokumen, lembar kerja, presentasi, dan formulir dapat disimpan dalam format file XML yang tersedia secara gratis untuk digunakan atau dilisensikan oleh siapa saja, bebas royalti. Office juga mendukung skema XML yang sesuai kebutuhan pengguna sehingga menyempurnakan tipe dokumen Office yang ada. Ini berarti bahwa pelanggan dapat dengan mudah membuka kunci informasi dalam sistem yang ada dan melakukan tindakan sesuai informasi tersebut. Informasi yang dibuat dalam Office dapat digunakan dengan mudah oleh aplikasi bisnis lain.
5. Deteksi dokumen yang memuat makro menjadi lebih mudah. File yang disimpan dengan menggunakan akhiran "x" (seperti .docx, .xlsx, dan .pptx) tidak dapat menyimpan makro *Visual Basic for Applications* (VBA) dan

makro XLM. Hanya file yang ekstensi berakhiran "m" (seperti .docm, .xlsm, dan .pptm) yang dapat memuat makro.