Aplikasi Transformasi Base 64 pada Kriptografi

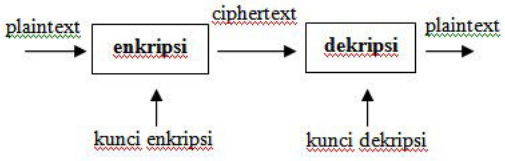
Yulisdin Mukhlis, Tjahjo Dwinurti, Akeda Bagus

Jl. Margonda Raya 100 Pondok Cina, Depok 16424, West Java, Indonesia
ymukhlis@staff.gunadarma.ac.id, dwinurti@staff.gunadarma.ac.id,
akeda@staff.gunadarma.ac.id

Ringkasan

Keamanan data merupakan hal yang sangat penting dalam menjaga kerahasiaan informasi, terutama yang berisi informasi sensitif yang hanya boleh diketahui isinya oleh pihak tertentu, sehingga perlu dilakukan penyandian data supaya beberapa pihak yang tidak memiliki kewenangan tidak akan dapat membuka informasi yang dikirim. Salah satu cara yang digunakan untuk pengamanan data adalah menggunakan sistem kriptografi yaitu dengan menyediakan isi informasi (plaintext) menjadi isi yang tidak dipahami melalui proses enkripsi (encipher), dan untuk memperoleh kembali informasi yang asli, dilakukan proses deskripsi (decipher), dengan menggunakan kunci yang benar. Cukup banyak algoritma pada kriptografi, salah satunya adalah algoritma Base64. Transformasi base64 digunakan untuk *Encoding dan Decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metoda yang digunakan untuk melakukan *encoding* terhadap data biner. Aplikasi ini akan menyajikan implementasi dari proses enkripsi dan dekripsi suatu data baik bersifat text maupun file dengan menggunakan Visual C#.

Kata kunci : Algoritma, kriptografi, base64, encoding, decoding



Gambar 1: Metode enkripsi dekripsi

1 Pendahuluan

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan yang bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak bertanggung jawab.Terdapat dua konsep utama pada kriptografi yaitu enkripsi dan dekripsi. Enkripsi adalah proses dimana informasi / data yang hendak dikirim berupa data jelas (plaintext) diubah menjadi bentuk yang hampir tidak dikenali berupa data random (ciphertext) sebagai informasi awalnya dengan menggunakan algoritma tertentu. Sedangkan dekripsi adalah kebalikan dari enkripsi yaitu mengubah kembali bentuk yang tersamar (ciphertext) tersebut menjadi informasi awal (plaintext).

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen teks terang /plaintext dan yang berisi elemen teks sandi/ciphertext. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. Apabila elemen-elemen teks terang dinotasikan dengan P, elemen-elemen teks sandi dinotasikan dengan C, sedang untuk proses enkripsi dinotasikan dengan E, dekripsi dengan notasi D[6].

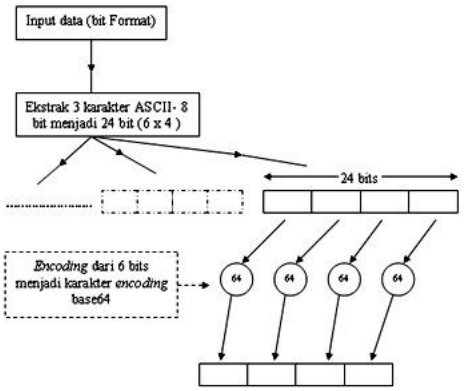
Enkripsi : E(P) = C (1)

Dekripsi : D(C) = P atau D(E(P)) = P (2)

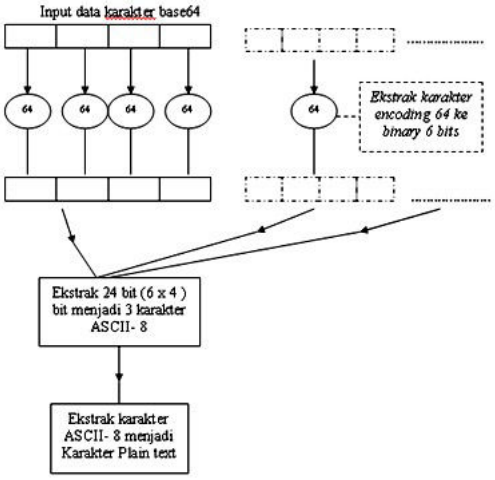
2 Algoritma Base 64

Transformasi base64 merupakan salah satu algoritma untuk Encoding dan Decoding suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metoda yang digunakan untuk melakukan encoding (penyandian) terhadap data binary. Karakter yang dihasilkan pada transformasi Base64 ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data binary atau pengisi pad. Karakter simbol yang akan dihasilkan akan tergantung dari proses algoritma yang berjalan.

Pada transformasi Base 64, digunakan dua buah



Gambar 2: Algoritma Encoding



Gambar 3: Algoritma decoding

tabel, yaitu tabel data encoding dengan 64 radix dan tabel ASCII. Tabel data encoding berfungsi untuk proses pengkorversian dari data biner ke karakter base 64 dan sebaliknya dari karakter base 64 ke data binary.

2.1 Algoritma Base 64 encoding [4]

Proses encoding pada transformasi base 64 diilustrasikan seperti pada gambar dibawah :

Dari gambar diatas, data input biner (yang merupakan hasil ekstrak dari karakter dengan menggunakan tabel ASCII) dimana satu karakter diwakili 8 bits kemudian kumpulan 8 bits tersebut di ekstrak menjadi kumpulan per 6 bits yang mewakili satu karakter yang disusun membentuk 4 bagian per blok (jadi tersusun beberapa blok dimana satu blok terdapat 24 bit data). Untuk kemungkinan jika terdapat data bit yang tidak mencapai 6 bits setelah proses ekstrak tadi, solusinya adalah dengan menambahkan bit 0 pada bit 6 hingga mencapai 6 bit dan bit 1 sisanya hingga mencapai 24 bit pada blok yang tersisa. Setelah kumpulan bit tersusun menjadi 24 bits dimana setiap blok nya dibagi menjadi 4 bagian, yang terdiri dari susunan-susunan 6 bits, barulah dari setiap bagian 6 bits tersebut data-data binary dapat dikonversikan menjadi karakter encoding base64 berdasarkan tabel data Encoding 64 radix diatas dimana satu bagian 6 bits mewakili satu karakter encoding base64.

2.2 Algoritma Base 64 decoding

Proses decoding pada transformasi base 64 diilustrasikan seperti pada gambar dibawah :

Pada proses dekripsi berdasarkan gambar algortima decoding diatas, terlihat input data berupa data karakter base64 hasil encoding pada proses sebelumnya, pertama-tama dengan menggunakan tabel Data Encoding 64 Radix, data karakter di ubah menjadi binary, dimana satu karakter di wakili oleh 6 bits data yang dikelompokkkan dalam blok yang berisi 24 bits data, setelah semua kumpulan bits data tersusun, kumpulan-kumpulan bits tersebut akan diekstrak menjadi kumpulan 8 bits data, dimana satu blok berisi 24 bits data akan di ekstrak menjadi 3 karakter ASCII 8 bits. Selanjutnya dengan menggunakan tabel ASCII, kumpulan 8 bits data tersebut di ekstrak menjadi karakter plaintext (8 bits data mewakili satu karakter Plaintext).

2.3 Algoritma Program

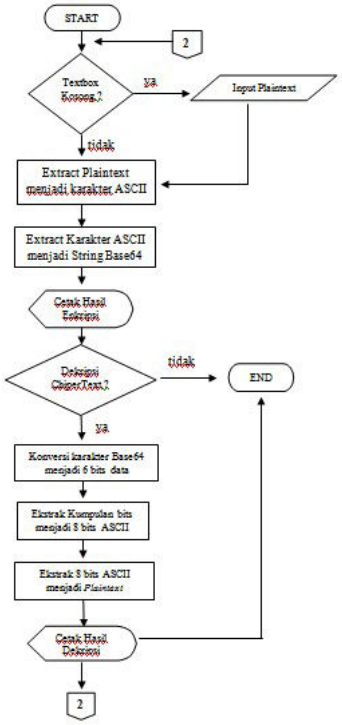
Perancangan program enkripsi dan dekripsi meliputi proses enkripsi dan dekripsi untuk text dan file.

3 Perancangan Program

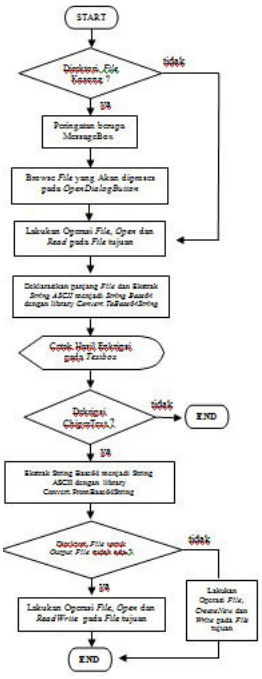
Dalam Source Code dengan Library terdapat dua buah form yang akan dijelaskan, yaitu Form1.cs.pada form pengoperasian enkripsi dekripsi input text dan Form2.cs.pada form pengoperasian enkripsi dekripsi input file.

3.1 Source code form1 cs [7]

- using System;
- using System.Collections.Generic;
- using System.ComponentModel;
- using System.Data;
- using System.Drawing;



Gambar 4: Flowchart encoding/decoding Text Base64



Gambar 5: Flowchart encoding/decoding file Base64

- using System.Text;
- using System.Windows.Forms;

Kode-kode diatas merupakan kumpulan kode keyword standar untuk pemograman dalam Visual C#. Kode using System menandakan bahwa program ini membutuhkan sistem yang ada pada Visual C# beserta komponen komponennya. Using System juga berhubungan dengan fungsi pembuatan class.

```
Private void
    button1_Click(object sender,
    EventArgs e)
{
 if (this.textBox1.Text == )
 {
  MessageBox.Show(PlainText harus
     diisi terlebih dahulu);
  return;
 }
  byte[]bytes=System.Text.
      Encoding.ASCII.GetBytes
      (this.textBox1.Text);
  string encoding64 = Convert.
         ToBase64String(bytes);
  this.textBox2.Text = encoding64;
}
```

Kode kode diatas, merupakan class yang digunakan untuk memanggil button btnEncode ketika button tersebut mengalami action click, Kemudian di dalam class diatas terdapat listing yang diguanakan untuk mengenkripsi suatu data dengan menggunakan fungsi library yang telah disediakan oleh .Net Framework 3.0 untuk keperluan encoding decoding base64 transformation cryptography. Pertama-tama diberikan kondisi bahwa akan ada peringatan berupa message box jika textbox1 (tempat user menginput Plaintext) masih kosong. Kemudian hasil input an yang masih berupa karakter String ASCII dikonversikan menjadi binary 8 bits sebelum akhirnya dikonversikan langsung menjadi karakter base64 dengan menggunakan fungsi Library .Net Framework, Convert.ToBase64String(byte[]). Hasil konversi encoding diletakkan pada textBox2.

```
Private void
 button2_Click(object sender,
 EventArgs e)
{
 if (this.textBox2.Text == )
 {
   MessageBox.Show(Data encoding base
      64 harus diisi terlebih dahulu);
   return;
 }
```

```
    byte[]bytes=Convert.
        FromBase64String
        (this.textBox2.Text);
    stringplaintext=
    System.Text.
      Encoding.ASCII.
      GetString(bytes);
    this.textBox3.Text = plaintext;
}
```

Kode pada class diatas digunakan untuk mengembalikan karakter base64 menjadi plaintext (Decoding). Sama halnya seperti pada proses encode, pertama-tama diberikan kondisi bahwa akan ada peringatan berupa message box jika textbox2 (tempat sumber inputi an Chipertext) masih kosong. Kemudian chipertext berupa karakter base64 langsung di konversikan menjadi kumpulan bit-bit data dengan menggunakan fungsi library yang telah disediakan oleh .Net Framework 3.0, yaitu Convert.FromBase64String(string base64), kemudian bit-bit data tersebut di konversikan lagi menjadi karakter String ASCII, sebelum akhirnya hasil decoding tersebut diletakkan pada textbox3.

3.2 Source Code Form2.cs

- using System;
- using System.Collections.Generic;
- using System.ComponentModel;
- using System.Data;
- using System.Drawing;
- using System.Text;
- using System.Windows.Forms;
- using System.IO;

Pertama-tama terdapat penggunaan Using.System yang sama dengan form1.cs pada source code with library sebelumnya. Terdapat satu komponen using System yang ditambahkan pada Form ini, yaitu using System.IO. Komponen ini berfungsi sebagai suatu system pada Visual C# yang digunakan untuk pengaksesan data file (misalnya read dan write) pada I/O File (input output file).

```
Private void
   btnEncode_Click(object sender,
   EventArgs e)
{
 if (!string.IsNullOrEmpty(txtOut.Text))
{
  FileStream fs = new FileStream(txtOut.Text,
        FileMode.Open, FileAccess.Read);
  byte[] filebytes = new byte[fs.Length];
   fs.Read(filebytes, 0,
   Convert.ToInt32(fs.Length));
  string encodedData =
Convert.
    ToBase64String(
    filebytes,
    Base64FormattingOptios.
    InsertLineBreaks);
  txtEncoded.Text = encodedData;
}
  else
  {
        MessageBox.Show(Silahkan
      Pilih File Yang Akan Di
      proses Pada Menu Browse!!!);
        return;
  }
}
```

Kode kode diatas, merupakan class yang digunakan untuk memanggil button btnEncode ketika button tersebut mengalami action click. Pertama-tama dilakukan pengecekan pada textbox txtOut.Text apakah txtOut.Text sudah memilki value atau belum, dimana value itu sendiri merupakan alamat dari directory file yang di pilih oleh user. Kondisinya jika txtOut.Text sudah memilki value, maka program akan melakukan operasi file dengan membuka (open) dan membaca (read) file yang dituju. Jika kondisinya bernilai false atau txtOut.Text belum memilki value maka program akan mengeluarkan peringatan dalam bentuk message box yang berisi string “Silahkan Pilih File Yang Akan Di proses Pada Menu Browse!!!”. Setelah file dibuka dan dibaca, program akan mendeklarasikan panjang dari isi File dan melakukan konversi String ASCII menjadi String Base64 dengan library Convert.ToBase64String. Hasil dari konversi itu sendiri akan diletakkan di txtEncoded.Text.

```
private void btnDecode_Click
   (object sender, EventArgs e)
{
        byte[]filebytes=Convert.
        FromBase64String(txtEncoded.Text);
  if (File.Exists(c:\\txtOutFile.Txt))
   {
   FileStream fs = new
   FileStream(c:\\txtOutFile.Txt,
   FileMode.Open,FileAccess.ReadWrite,
    FileShare.None);
   fs.Write(filebytes, 0,
    filebytes.Length);
```

```
    fs.Close();
  }
else
  {
        FileStream fs = new
    FileStream(c:\\txtOutFile.Txt,
    FileMode.CreateNew, FileAccess.
     Write, FileShare.None);
    fs.Write(filebytes, 0,
      filebytes.Length);
     fs.Close();
        }
}
```

Kode pada class diatas digunakan untuk mendeskripsikan kembali isi file yang telah di enkripsi sebelumnya dengan menggunakan keyword library Convert.FromBase64String. Kondisi pada kode diatas menyatakan bahwa apakah file txtOutFile.Txt sudah ada sebelumnya pada directory c:\\, dimana txtOutFile.Txt sendiri merupakan file tempat diletakkannya output hasil dekripsi dari file source (dimana isi dari txtOutFile.Txt harus sama dengan file source). Jika kondisi diatas bernilai true, maka program akan melakukan operasi file berupa Open dan ReadWrite hasil dekripsi pada txtOutFile.Txt. Kemudian jika kondisi diatas bernilai false, maka program akan melakukan operasi file berupa CreateNew (membuat baru) file bernama txtOutFile.Txt pada directory c:\\ dan Write hasil dekripsi pada txtOutFile.Txt yang telah dibuat tadi.

```
Private void button1_Click
  (object sender, EventArgs e)
{
  Stream myStream;
  OpenFileDialog openFileDialog1 =
    new OpenFileDialog();
  openFileDialog1.InitialDirectory =
   C:\\; openFileDialog1.Filter =
    txt files (*.txt)|*.txt|Doc
    files (*.doc)|*.doc|
    Rtf files (*.rtf)|
    *.rtf|
    All files    (*.*)|*.*;
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.RestoreDirectory
    = true;
  if (openFileDialog1.ShowDialog() ==
    DialogResult.OK)
  {
    if ((myStream =
      openFileDialog1.OpenFile())
      != null)
  {
        txtOut.Text =
     openFileDialog1.FileName;
    myStream.Close();
  }
 }
}
```

Kumpulan kode diatas merupakan kode-kode yang digunakan untuk mendeklarasikan OpenFileDialog yang akan digunakan untuk mencari (browse) file yang akan di proses oleh user. openFileDialog1.InitialDirectory = "C:\\" merupakan deklarasi yang berfungsi untuk menentukan home direcotry ketika OpenFileDialog pertama kali dibuka, itu berarti ketika tombol browse maka default home directory pertama kali adalah drive C. Kemudian openFileDialog1. Filter digunakan untuk mendeklarasikan jenis-jenis file yang dapat diproses dalam aplikasi ini, terlihat pada kode diatas, di deklarasikan tipe-tipe file seperti .Txt, .Doc, .Rtf, dan All Files pada file of type ketika openFileDialog digunakan. Kemudian openFileDialog1.FilterIndex = 1 berfungsi sebagai default dari tipe-tipe file yang ada pada file of type pada openFileDiaog. Jadi ketika pertama kali openFileDialog digunakan, file of type akan menunjukkan index arrray pertama sesuai dengan urutan pendeklarasian tipe file diatas yaitu txt files|*.txt. kemudian pendeklarasian yang terakhir, openFileDialog1.RestoreDirectory = true akan digunakan untuk mengembalikan posisi directory pada openFileDialog ke posisi deklarasi InitialDirectory setelah digunakan oleh user. Jadi ketika openFileDialog akan digunakan lagi, posisi directory kembali berada pada default InitialDirectory yaitu drive C:\\ .

4 Kesimpulan

Algoritma Base64 dapat diimplementasikan pada .Net Framework dengan menggunakan Microsoft Visual C# 2008 melalui 2 cara, yaitu pertama dengan menggunakan library yang telah di sediakan oleh .Net Framework, sedangkan yang kedua dimana implementasi yang dibuat tanpa menggunakan library yang disediakan oleh .Net Framework. Berdasarkan pada algoritma dan perancangan program, transformasi Base64 cocok digunakan untuk aplikasi kriptografi berbasis text dan file.

Pustaka

[1] http://www.c-sharpcorner.com.

[2] http://www.codeproject.com/kb/cs/base64encdec.aspx.

[3] http://www.id.wikipedia.org/ wiki/ascii.

[4] http://www.ketepeng.wordpress.com/2008/12/01/encripsi - dan-decripsi-password-menggunakan-metode-base64/.

[5] http://www.microsoft.com/ express/2005/download/.

[6] Dony Arivus. *Pengantar Ilmu Kriptografi Teori, Analisis dan Implementasi*. ANDI, Yogyakarta, 2008.

[7] Jaenudin. *Belajar Sendiri .net dengan Visual C 2005*. ANDI, Yogyakarta, 2006.