Studi dan Implementasi Steganografi Pada Data Biner Dengan Berkas XAML Sebagai *Cover-Object*

Puja Pramudya

Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Cisitu Lama 3D/160B, Bandung, Jawa Barat
e-mail: puja.pramudya@gmail.com
if16058@students.if.itb.ac.id

Abstrak – *Masalah keamanan data merupakan salah satu masalah yang terdapat dalam pengiriman dan penerimaan pesan. Aspek keamanan data pada pengiriman dan penerimaan pesan dapat dicapai salah satunya dengan menggunakan teknik steganografi, yaitu sebuah teknik penyembunyian pesan sehingga pesan tidak dapat dideteksi keberadaannya dan tidak memunculkan kecurigaan bagi orang yang tidak berhak melihatnya. Makalah ini menguraikan teknik steganografi dengan menggunakan berkas teks sebagai cover-object. Berkas yang dipilih adalah berkas Xaml, yaitu format berkas untuk mendukung desain antarmuka pada framework .NET 3.5 yang merupakan produk dari vendor Microsoft. Studi yang dilakukan adalah pengujian kelayakan berkas Xaml untuk digunakan sebagai cover-object. Diusulkan pula sebuah teknik yang disebut teknik perubahan posisi atribut yang dapat dan layak dijadikan salah satu alternatif penggunaan steganografi pada penyembunyian pesan.*

Kata Kunci *: Steganografi, Xaml, Atribut, Perubahan posisi atribut*

1. PENDAHULUAN

Dewasa ini penyembunyian pesan tidak hanya dapat dilakukan dengan menyandikan pesan. Melainkan dapat pula menyisipkan pesan tersebut dalam media lain sehingga tidak akan memunculkan kecurigaan terhadap pesan yang dikirimkan karena pesan tersebut tidak terlihat, yang terlihat hanyalah media penampung pesan tersebut.

Sebagai contoh, jika diinginkan pengiriman pesan kepada seseorang melalui email. Karena dikhawatirkan pesan akan diketahui orang lain maka pesan tersebut disisipkan ke dalam sebuah media lain yang berukuran lebih besar. Misalnya dalam media berkas citra. Sehingga orang lain tidak akan curiga akan gambar yang dikirimkan. Hal ini tentunya akan lebih praktis ketimbang mengirimkan pesan dalam bentuk berkas yang terenkripsi. Hal ini tentunya akan membuat orang lain curiga dan mulai melakukan *attack* untuk mengetahui isi pesan yang dikirimkan.

Teknik penyisipan pesan dalam media lain yang lebih besar ini dinamakan Steganografi. Media penyimpan yang dapat digunakan dalam steganografi dapat berupa berkas lagu, citra, atau berkas-berkas lain yang berukuran besar dan dapat dimasukkan pesan yang ingin disembunyikan.

Dengan menggunakan steganografi maka orang tidak akan menjadi curiga kalau ternyata terdapat pengiriman pesan rahasia. Tetapi terdapat harga yang harus dibayar dalam steganografi yaitu besarnya ukuran berkas yang disisipi pesan rahasia. Seringkali diperlukan transfer data dalam jumlah besar padahal data penting yang diperlukan hanya berukuran kecil.

2. STEGANOGRAFI

Steganografi adalah teknik menyisipkan pesan kedalam suatu media . Penyisipan ini dilakukan agar pihak ketiga tidak menyadari keberadaan pesan tersebut. Pada masa kini, steganografi lebih banyak digunakan pada data dijital, dengan media teks, gambar, audio dan video dengan algoritma yang beragam.

Beberapa contoh media penyisipan (*cover-object*) yang digunakan dalam teknik steganografi adalah :

1. Teks
Dalam teknik steganografi yang menggunakan teks sebagai media penyisipan biasanya digunakan teknik NLP sehingga teks yang telah disisipi pesan rahasia tidak akan dicurigai mengandung sebuah pesan.

2. Audio
Format audio relatif sering dipilih sebagai *cover-object* karena biasanya berukuran cukup besar. Ukuran ini akan menambah daya tampung *cover-object* terhadap pesan yang disisipkan

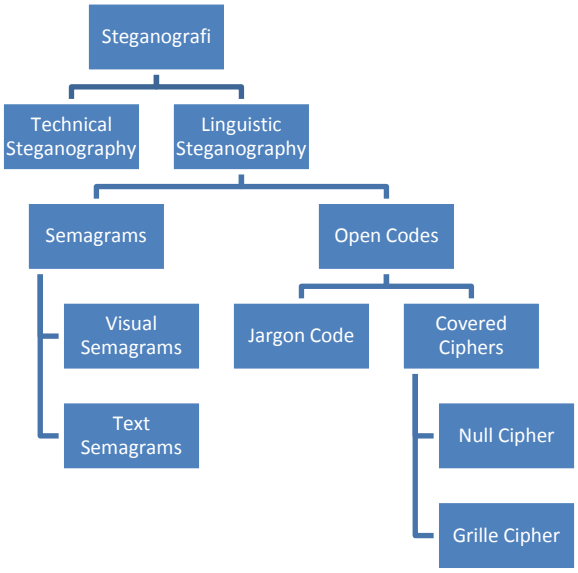
3. Citra

Format citra adalah yang paling sering digunakan dikarenakan format ini merupakan salah satu format berkas yang sering dipertukarkan dalam jaringan internet. Alasan lainnya adalah terdapat banyak algoritma steganografi untuk media penampung yang berupa citra.

4. Video

Format ini merupkanan format berkas yang berukuran cukup besar sehingga layak dijadikan media penampung. Hanya saja video jarang digunakan karena ukurannya terlalu besar sehingga mengurangi kepraktisan dan juga algoritma yang mendukung format ini relatif lebih sedikit.

Selain berdasarkan format berkas penampung yang digunakan, menurut Bauer, steganografi dapat dikelompokkan menjadi sebuah taksonomi sebagai berikut :



Gambar 1 Taksonomi Steganografi

1. Technical Steganography

Teknik ini menggunakan metode sains untuk menyembunyikan pesan. Contohnya adalah penyembunyian pesan di dalam *chip* mikro.

2. Linguistic Steganography

Teknik ini menggunakan penyembunyian pesan dengan cara-cara yang tidak lazim secara kebahasaaan. Dibagi menjadi Semagrams dan Open Codes.

3. Semagrams

Teknik ini menyembunyikan informasi dengan menggunakan simbol atau tanda. Contoh penggunaannya adalah dengan mengganti ukuran teks atau mengganti ukuran font. Pergantian ukuran atau tipe tersebut yang digunakan sebagai media penyisipan pesan. Algoritma steganografi yang termasuk tipe ini adalah Algoritma Gifshuffle.

4. Open Codes

Teknik ini menyembunyikan pesan dengan cara yang tidak umum namun tetap tidak menimbulkan kecurigaan.

5. Jargon Code

Teknik ini sesuai dengan namanya menggunakan bahasa yang hanya dimengerti oleh sebagian orang. Sebagai contoh adalah *warchalking*, *underground terminology* atau percakapan biasa yang hanya diketahui oleh pembicara.

6. Covered Ciphers

Teknik ini menyembunyikan pesan dalam media pembawa sehingga pesan kemudian dapat diekstrak dari media pembawa tersebut oleh pihak yang mengetahui bagaimana pesan tersembunyi tersebut disembunyikan.

Penilaian sebuah teknik steganografi yang baik dapat dinilai terhadap beberapa faktor yaitu :

1. Imperceptiable

Keberadaan pesan dalam media penampung tidak dapat dideteksi

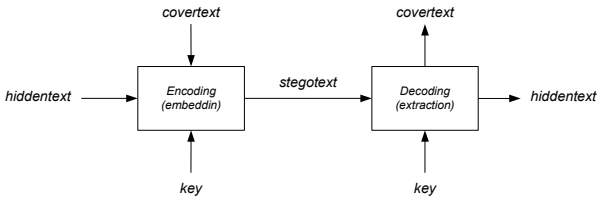
2. Fidelity

Mutu media penampung setelah ditambahkan pesan rahasia tidak jauh berbeda dengan mutu media penampung sebelum ditambahkan pesan.

3. Recovery

Pesan rahasia yang telah disisipkan dalam media penampung harus dapat diungkap kembali. Hal ini merupakan syarat mutlak dalam sebuah algoritma steganografi karena ada banyak cara penyisipan pesan yang tidak terdeteksi namun sulit dalam pembacaan kembali.

Ada dua buah proses dalam steganografi yakni proses penyisipan pesan dan proses ekstraksi pesan. Proses penyisipan pesan membutuhkan masukan media penyisipan, pesan yang akan disisipkan dan kunci. Keluaran dari proses penyisipan ini adalah media yang telah berisi pesan. Proses ekstraksi pesan membutuhkan masukan media yang telah berisi pesan. Keluaran dari proses ekstraksi pesan adalah pesan yang telah disisipkan.



Gambar 2 Proses Steganografi

3. MEDIA PENYISIPAN (*COVER-OBJECT*)

Seperti yang telah dibahas sebelumnya bahwa salah satu berkas yang dapat digunakan sebagai *cover-object* adalah berkas teks.

Beberapa algoritma atau teknik steganografi menggunakan berkas teks sebagai *cover-object* adalah :

Teknik LSB

Teknik ini mengunakan modifikasi langsung nilai *byte* dari *cover-object*. Bit-bit pesan digunakan untuk mengganti bit-bit kurang berarti (*least significant bit* atau LSB) dari *cover-object*. Akan tetapi karena diterapkan di dalam berkas teks, berkas keluaran (*stego-object*) akan sangat mencurigakan karena perubahan sedikit saja *bit* dari karakter pada berkas teks akan mengakibatkan perubahan yang cukup besar.

Teknik Manipulasi Elemen Teks

Teknik ini menggunakan manipulasi elemen-elemen di dalam teks untuk menyisipkan pesan. Misalnya dengan pengubahan huruf, spasi antar baris, jarak antar kalimat dan sebagainya.

Teknik NLP

Teknik NLP atau *Natural Language Processing* diterapkan pada manipulasi atribut leksikal,sintaks dan semantik dari teks dengan mengusahakan seminimal mungkin untuk mengubah arti sebenarnya. Teknik ini lebih kokoh dibandingkan menggunakan manipulasi elemen teks.

4. BERKAS XAML

Seperti telah dijelaskan bahwa berkas berformat teks dapat digunakan sebagai *cover-object* untuk penyembunyian pesan. Salah satu berkas berformat teks yang jarang digunakan adalah berkas XAML.

XAML (dibaca "zammel") atau *Extensible Application Markup Language* adalah sebuah bahasa *markup* yang didefinisikan Microsoft untuk antarmuka statis maupun dinamis dalam aplikasi .NET yang dapat mendukung *declarative programming*. XAML sebenarnya ditujukan untuk Windows Vista namun dapat juga diimplementasikan dengan Windows XP atau Wndows 2003.

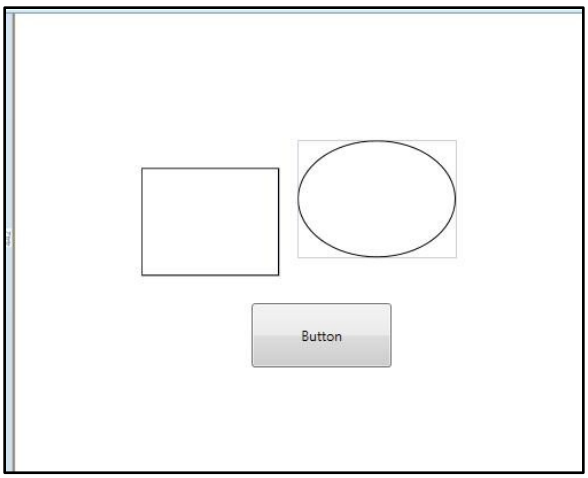
XAML berguna untuk memisahkan kode antarmuka dengan kode lojik aplikasi dan sangat mirip dengan konsep MVC. XAML dibundel di dalam *Windows Presentation Foundation* (WPF) untuk membangun antarmuka di dalam aplikasi .NET 3.0 dan Vista. Pada dasarnya, XAML merupakan ekstensi dari XML. Setiap kode XAML harus berbentuk *well-formed* dan XAML mewarisi semua definisi XML dan aturannya.Yang membedakan XAML dan XML adalah apa yang direpresentasikan oleh XAML. Setiap elemen pada berkas XAML merepresentasikan sebuah kelas pada .NET CLR (*Common Language Runtime*).

Contoh dibawah ini adalah potongan kode XAML yang dibangkitkan dengan menggunakan kakas *Microsot Expression Blend* .

```
<Window
      xmlns="http://schemas.microso
ft.com/winfx/2006/xaml/presentation
"
      xmlns:x="http://schemas.micro
soft.com/winfx/2006/xaml"
      x:Class="CoverText.Window1"
      x:Name="Window"
      Title="Window1"
      Width="640" Height="480"
Activated="Window_Activated">

      <Grid x:Name="LayoutRoot">
            <Rectangle
Fill="#FFFFFFFF" Stroke="#FF000000"
HorizontalAlignment="Left"
Margin="112,160,0,188"
Width="120"/>
            <Ellipse
Fill="#FFFFFFFF" Stroke="#FF000000"
Margin="248,136,232,204"/>
            <Button
Margin="208,0,288,108"
VerticalAlignment="Bottom"
Height="56" Content="Button"/>
      </Grid>
</Window>
```

Kode diatas akan diinterpretasi menjadi bentuk berikut :



Gambar 3 Berkas XAML

Pada dasarnya tidak dapat disisipkan kode lain atau pesan apapun di dalam berkas XAML. Apapun yang terdapat di dalam berkas XAML akan terlihat pada Editor XAML (misalnya Microsoft Visual Studio atau Expression Blend), atau terlihat dari kode sumber sehingga penggunaan steganografi jadi tidak berguna. Akan tetapi, posisi dan urutan atribut pada deklarasi elemen dapat diubah tanpa mempengaruhi penampakan berkas dan ukuran berkas. Dengan cara ini dapat dilakukan proses penyisipan pesan tanpa diketahui oleh siapapun. Penulis akan mengusulkan teknik ini yang penulis sebut sebagai Teknik Perubahan Posisi Atribut (*Attribute Position Changing Technique*).

5. TEKNIK PERUBAHAN POSISI ATRIBUT

Teknik perubahan posisi atribut didasarkan pada perubahan posisi dari atribut-atribut elemen berkas XAML yang bersesuaian. Pada berkas XAML tiap elemen atau kelas pada CLR didefinisikan menjadi pasangan *tag* yang saling melengkapi (*well formed*). Setiap *tag* akan mengandung atribut-atribut untuk mendefinisikan *style* atau *format* antarmuka dari elemen tersebut. Perubahan posisi atribut pada *tag* ini tidak akan mengubah hasil dari interpretasi kakas Blend maupun aplikasi yang dihasilkan nantinya. Dengan demikian tidak terdapat modifikasi *byte* sama sekali terhadap berkas *cover-object*.

Perhatikan contoh berikut :

```
<Rectangle          Fill="#FFFFFFFF"
Stroke="#FF000000" />
<Ellipse            Fill="#FFFFFFFF"
Stroke="#FF000000" />
```

```
<Rectangle          Stroke="#FF000000"
Fill="#FFFFFFFF"   />
<Ellipse            Stroke="#FF000000"
Fill="#FFFFFFFF"/>
```

Potongan kode XAML diatas menampilkan dua variasi posisi atribut yang mengasilkan konten yang sama. Sekarang dapaf didefinisikan kunci yang sederhana untuk bentuk diatas

Atribut elemen yang berkorespondensi adalah :

```
Fill   Stroke
```

```
if (Fill before stroke)
      {
       tag encodes a "1" bit}
else  {
       tag encodes a "0" bit
       }
```

Dengan demikian, setiap kombinasi dari dan `Fill` dan `Stroke` akan merepresentasikan 1 bit. Akan diperlukan 80 *tag* elemen *Shape* pada XAML untuk menyembunyikan 10 karakter dari pesan teks. Hal ini mengesankan terlalu besar ukuran *cover-object* yang diperlukan untuk menyembunyikan pesan yang berjumlah sedikit. Akan tetapi perlu diketahui bahwa berkas XAML memiliki banyak lagi kombinasi atribut yang bersesuaian, terutama apabila menggunakan teknik-teknik desain dengan penggunaan elemen *Storyboard*, *Style*, *Resource* dan sebagainya.

Misalnya pada penggunaan animasi dengan *Storyboard* seperti berikut :

```
<DoubleAnimationUsingKeyFrames
BeginTime="00:00:00"
Storyboard.TargetName="button"
Storyboard.TargetProperty="(UIEleme
nt.RenderTransform).(TransformGroup
.Children)[0].(ScaleTransform.Scale
X)">

      <SplineDoubleKeyFrame
KeyTime="00:00:01.1000000"
Value="2.438"/>

      <SplineDoubleKeyFrame
KeyTime="00:00:01.4000000"
Value="2.438"/>

      <SplineDoubleKeyFrame
KeyTime="00:00:01.6000000"
Value="1.901"/>
```

```
        </DoubleAnimationUsingKeyFrames>
```

Untuk satu animasi seperti diatas yang melakukan pergerakan rotasi,translasi dan dilatasi ukuran, terdapat atribut `KeyTime` dan `Value` yang berpasangan dan dapat digunakan untuk penyembunyian data dengan pengubahan posisi atribut. Dapat dikatakan semakin banyak atau semakin kompleks animasi yang terdapat pada berkas XAML akan semakin banyak pula tempat untuk menyembunyikan data.

Kunci untuk operasi ini terletak pada tabel atau daftar atribut yang didefinisikan. Untuk itu diperlukan pembangunan kunci berupa tabel atribut yang dapat disimpan menjadi berkas lain, misalnya berkas XML,

Dengan melakukan perubahan pada setiap atribut yang didefinisikan pada berkas XAML dapat dilakukan penyembunyian terhadap pesan yang tidak ingin diketahui oleh orang lain. Implementasi penggunaan teknik ini akan dibahas pada bagian selanjutnya.

6. IMPLEMENTASI

Untuk implementasi steganografi pada berkas XAML pertama-tama dibutuhkan dua kelas untuk menyimpan *tag* XAML dan atributnya. Atribut tidak memiliki banyak *properties*, cukup nama dan *value*-nya saja. Setiap atribut di dalam *tag* dapat digunakan untuk satu bit pesan. Aplikasi harus mampu menandai apakah sebuah atribut sudah ditangani atau belum. Selanjutnya, sebuah *tag* pada XAML memiliki nama dan sejumlah atribut. Konstruktor akan melakukan pencarian teks pada *tag* untuk atribut dan nilai dari atribut tersebut.

Selanjutnya, operasi penyembunyian data akan melakukan pendaftaran seluruh *tag* XAML, melakukan iterasi terhadap seluruh *tag* dan atributnya. Atribut yang telah di-*handle* akan diabaikan. Jika atribut masih belum di-*handle* dan belum digunakan dilakukan operasi *look-up* ke tabel kunci atribut.

Algoritma penyembunyian pesan adalah sebagai berikut :

1. Baca berkas XAML yang akan dijadikan *cover-object*
2. Daftarkan seluruht *tag* XAML
3. Untuk setiap tag XAML pada daftar *tag*
    - Jika atribut belum di-*handle* :
        - cari baris kunci pada tabel *look-up*
        - lakukan penukaran posisi atribut
    - Jika atribut sudah di-*handle* diabaikan

Algoritma ekstraksi adalah sebagai berikut :

1. Baca dokumen *stegano-object*
2. Daftarkan seluruh *tag* XAML
3. Untuk setiap *tag* XAML pada daftar *tag*
    - Untuk setiap atribut pada daftar atribut
    - Jika atribut belum di-*handle* :
    - Cari baris kunci pada tabel *look-up*
    - Bandingkan dengan atribut posisi
    - Tukar jika terdapat perbedaan dengan tabel
    - Tandai bahwa atribut sudah di-*handle*

Untuk impelementasi penulis mengembangkan prototipe aplikasi dengan menggunakan framework .NET 3.5 menggunakan bahasa C#. Berikut tabel kebutuhan kelas desain untuk implementasi teknik steganografi dengan perubahan posisi atribut :

Tabel 1 Daftar Kebutuhan Kelas

No	Kelas	Fungsi
1	XamlAttribute.cs	Pemodelan object atribut Xaml
2	XamlAttributeCollection	Kelas untuk menampung list of Xaml
3	XamlTag.cs	Pemodelan objek tag Xaml
4	XamlTagCollection.cs	Kelas untuk menampung list of tag Xaml
5	XamlUtility.cs	Kelas untuk melakukan operasi yang berhubungan dengan penyembunyian dan ekstraksi pesan
6	MainForm.cs	Kelas untuk antarmuka

Pada penelitian ini pesan yang akan disisipkan *dibatasi* hanya berasal dari masukan pengguna melalui papan kunci. Pengguna dapat memilih tabel *look-up* yang ingin dijadikan kunci. Pengguna juga dapat melakukan penyuntingan terhadap kunci yang ingin digunakan. Pesan yang akan disisipkan akan diubah menjadi bentuk biner (*binary data*) di dalam memori dan kemudian dilakukan perubahan posisi

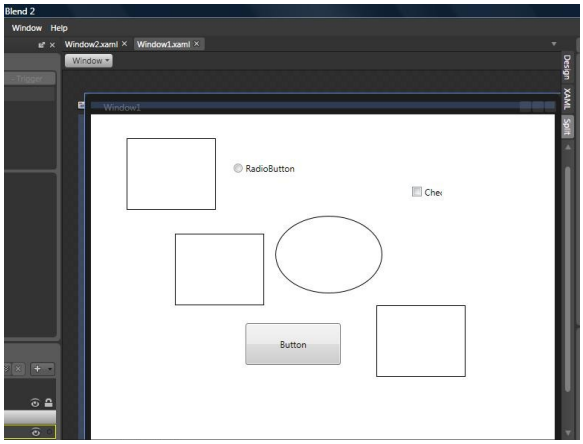
atribut pada *cover-object* untuk penyembunyian pesan.

7. HASIL DAN PEMBAHASAN

Berdasarkan hasil implementasi, perangkat lunak yang memiliki fungsi untuk penyisipan pesan dan ektraksi pesan dengan berkas XAML sebagai *cover-object* telah berhasil dikembangkan. Proses selanjutnya adalah memeriksa kelayakan berkas XAML sebagai *cover-object* dengan melihat kriteria *imperceptibility, fidelty* dan *recovery*. Berkas yang akan digunakan adalah **Window1.xaml** yang dihasilkan dengan kakas Blend dan pesan yang akan disisipkan adalah **pramudya**. *Stego-object* akan disimpan sebagai berkas **Window2.xaml**

7.1 Pengujian Kriteria *Imperceptibility*

Proses dari pengujian ini adalah menyisipkan pesan ke dalam berkas XAML, kemudian dilakukan usaha pendeteksian keberadaan pesan secara kualitatif, yaitu melihat hasil interpretasi kakas dan kode sumber.

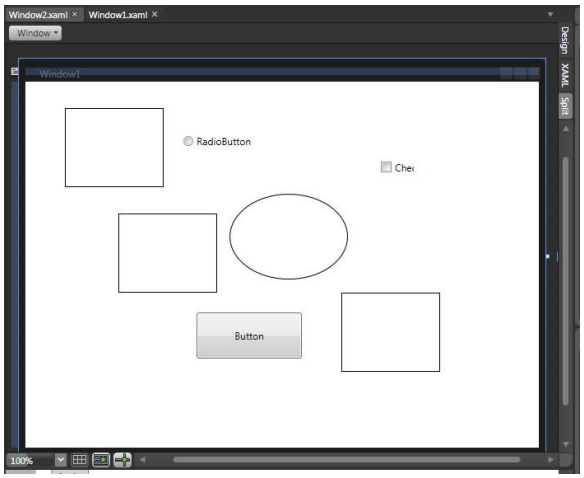


Gambar 4 Berkas Asli

```
<Storyboard x:Key="Test">

      <DoubleAnimationUsingKeyFrame
s BeginTime="00:00:00"
Storyboard.TargetName="rectangle"
Storyboard.TargetProperty="(UIEleme
nt.RenderTransform).(TransformGroup
.Children)[3].(TranslateTransform.X
)">

      <SplineDoubleKeyFrame
KeyTime="00:00:00.3000000"
Value="223"/>
```



Gambar 5 Berkas Stego-object

```
<Storyboard x:Key="Test">

      <DoubleAnimationUsingKeyFrame
s BeginTime="00:00:00"
Storyboard.TargetName="rectangle"
Storyboard.TargetProperty="(UIEleme
nt.RenderTransform).(TransformGroup
.Children)[3].(TranslateTransform.X
)">

      <SplineDoubleKeyFrame
Value="223"
KeyTime="00:00:00.3000000">

      </DoubleAnimationUsingKeyFram
es>
```

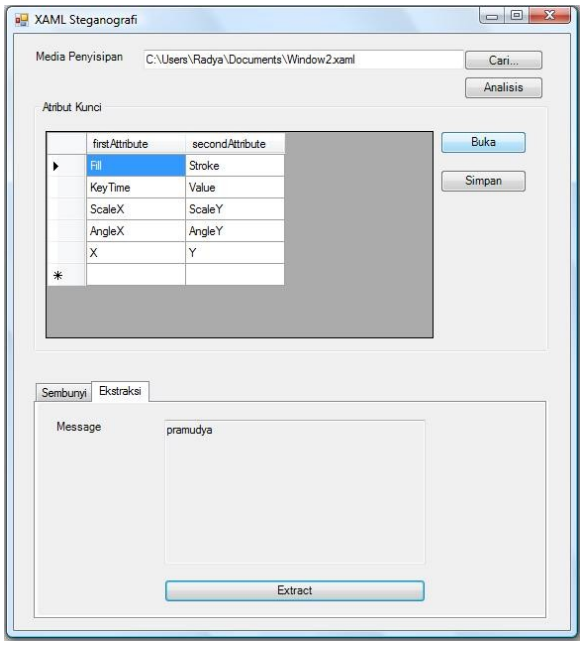
Terlihat bahwa berkas asli dan berkas stego-object tidak dapat dibedakan karena ketika ditampilkan pada kakas Blend terlihat sama seluruhnya. Begitu juga kode sumber tidak akan memperlihatkan kecurigaan karena urutan atribut elemen pada penulisan kode XAML tidak memiliki aturan tertentu sehingga setiap pemrogram dapat memiliki gayanya masing-masing. Tidak dapat dideteksi apakah terdapat pesan rahasia di dalam berkas XAML.

7.2 Pengujian Kriteria *Fidelity*

Proses pengujian terhadap kriteria *fidelity* dilakukan secara kualitatif dengan membandingkan berkas asli dan berkas *stego-object* dari aspek ukuran dan penampakan pada kakas setelah dikenai proses steganografi. Ukuran berkas tidak berubah, keduanya berukuran sama dan penampakan pada kakas tidak berubah sama sekali.

7.3 Pengujian Kriteria *Recovery*

Proses pengujian terhadap kriteria *recovery* dilakukan dengan mengekstraksi kembali pesan yang telah disisipkan. Proses memerlukan berkas *stego-object* dan tabel kunci yang sama pada proses penyembunyian pesan. Pesan **pramudya** dapat dihasilkan kembali.



Gambar 6 Hasil Ekstraksi

8. KESIMPULAN

Berikut beberapa kesimpulan yang dapat diambil :

1. Berkas XAML layak dijadikan *cover-object* karena memenuhi kriteria *imperceptibility, fidelity* dan *recovery*.
2. Steganografi dengan teknik Perubahan Posisi Atribut dapat diterapkan pada berkas XAML.
3. Perangkat lunak yang mengimplementasikan steganografi dengan teknik Perubahan Posisi Atribut pada berkas Xaml berhasil dibangun. Kebutuhan fungsional dari perangkat lunak seperti penyisipan, ekstraksi dan pembangunan tabel kunci dapat dilakukan dengan benar.
4. Jumlah pesan yang dapat disisipkan sangat tergantung kepada jumlah pasangan atribut yang terdapat di dalam berkas XAML.
5. Teknik Perubahan Posisi Atribut pada berkas XAML dapat dijadikan alternatif pengiriman pesan dengan memperhatikan aspek keamanan.

DAFTAR PUSTAKA

[1] Munir, Rinaldi, Diktat Kuliah IF5054 Kriptografi, Penerbit ITB 2006

[2] Cole,Eric. *Hiding in Plainsight : Steganography and the Art of Covert Communication*. Wiley Publishing, Inc. 2004

[3] Introduction to XAML, http://aspalliance.com/1019_introduction_to_xaml

[4] XAML, http://msdn.microsoft.com/en-us/library/ms747122.aspx

[5] http://homes.cerias.purdue.edu/~mercan/spie06_6072-9_paper.pdf