Steganografi pada *DOS text file* dan *rich text format file* dengan memanfaatkan atribut file yang tidak terlihat

Alfan Farizki Wicaksono – 135 06 067
Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha nomor 10, Kota Bandung, Indonesia
e-mail: farizki@comlabs.itb.ac.id, website : http://students.itb.ac.id/~alfan_fw

ABSTRAK

Steganografi merupakan sebuah teknik yang digunakan untuk penyembunyian pesan di dalam sebuah pesan. Keuntungannya adalah orang tidak akan merasa curiga terhadap pesan yang dikirimkan bahwa pesan tersebut mengandung pesan rahasia.
Makalah ini memperkenalkan sebuah metode yang dapat kita lakukan jika kita ingin melakukan steganografi pada DOS text file dan rich text format file. Kita dapat memanfaatkan *EOF (end of file)* yang ada pada DOS text file untuk menyisipkan sebuah pesan rahasia. Kita juga dapat memanfaatkan atribut warna dan ukuran huruf pada steganografi rich text format file.
Makalah ini juga akan membahas tentang rekayasa penyisipan pesan agar tidak diketahui dengan mudah oleh orang. Rekayasa ini meliputi rasio jumlah penyisipan pesan rahasia dengan pesan palsu (covertext), pemilihan warna, dan pemilihan ukuran huruf.

Kata kunci: Steganografi, DOS text file, rich text file.

1. PENDAHULUAN

Rich text format merupakan file text yang mengandung atribut lain selain semantik dari text yang sebenarnya. Atribut itu mengandung *style* atau gaya penulisan dari text yang dikandung seperti ketebalan huruf, kemiringan huruf, warna huruf, ukuran huruf, hyperlink, dan fitur yang lainnya. Atribut ini diletakkan dalam file text itu juga tetapi dalam format tertentu. Jika rich text format ini kita buka dengan aplikasi *word processor* seperti *Microsoft office word*, kita akan melihat text yang dikandung dengan gaya penulisannya. Kita tidak bisa melihat format text yang menspesifikasikan gaya penulisannya (tidak dapat dilihat di *word processor*).

Begitu pula dengan *DOS text file*. File text ini merupakan file text yang distandarkan oleh Microsoft untuk *Disk Operating System atau Command Prompt*. Kita dapat membaca file text ini dengan menggunakan program TYPE pada DOS. File ini mempunyai beberapa atribut yang tidak terlihat jika ditampilkan ke layar dengan program TYPE. Salah satu atribut yang terpenting adalah *end-of-file* yang berupa *ASCII control-Z*. Kita akan memanfaatkan atribut ini untuk menyisipkan pesan ke dalam *covertext* bertipe *DOS text file*. Hal ini tentu saja akan sangat berguna untuk para pengguna DOS / Command prompt dan DOS programmer.

Ide yang digunakan untuk menyisipkan pesan rahasia pada file *rich text format* adalah dengan memanfaatkan nilai atribut file tersebut. Kita dapat mengubah atribut gaya penulisan file dengan maksud menyisipkan pesan rahasia kedalamnya. Misalnya, kita dapat mengubah warna sebuah huruf dan huruf yang lainnya. Warna huruf itu nantinya akan menjadi sebuah kode untuk pesan rahasia. Begitu pula dengan *DOS text file*, kita dapat meletakkan sebuah pesan rahasia setelah atribut *ASCII control-Z* tersebut. Akibatnya, pesan tersebut tidak akan ditampilkan ke layar karena program akan berhenti menampilkan isi file setelah menemui *end-of-file*. Intinya adalah kita dapat memanfaatkan atribut yang tersembunyi pada file untuk penyisipan pesan rahasia.

Aspek psikologis, aspek ukuran file, dan aspek penglihatan mata dalam melakukan rekayasa terhadap penyisipan pesan rahasia ke dalam *covertext* juga merupakan sebuah kajian yang penting. Misalan, pengenkripsian pesan yang disisipkan setelah *end-of-file* pada *DOS text file*, orang hanya akan mengira itu adalah kerusakan pada disk atau semacamnya. Begitu pula dengan mode pengubahan warna huruf menjadi warna yang tidak jauh berbeda dengan warna huruf sekitarnya. jika dilihat oleh mata manusia, tulisan yang berubah warna tersebut tidak akan terlihat berbeda dengan tulisan lain yang tidak berubah warna. Padahal, pengubahan warna huruf itu merupakan sebuah penyisipan pesan rahasia ke dalam *covertext*. Aspek ukuran file lebih terkait dengan rasio panjang pesan yang disisipkan dengan panjang covertext. Orang yang membaca pesan akan curiga dengan ukuran file yang besar tetapi tulisan yang ditampilkan hanya sedikit. Ia akan mengira ada pesan tersembunyi di dalamnya. Jadi, kita harus punya aturan perkiraan untuk menentukan pesan rahasia dengan

panjang tertentu disisipkan ke dalam covertext dengan panjang tertentu.

2. METODE

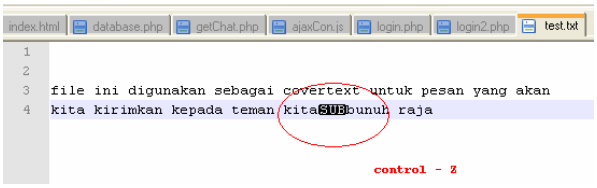
Bagian ini akan menjelaskan mengenai berbagai macam metode yang digunakan jika kita ingin melakukan steganografi pada DOS text file dan juga rich text format file.

2.1 Steganografi pada DOS text file

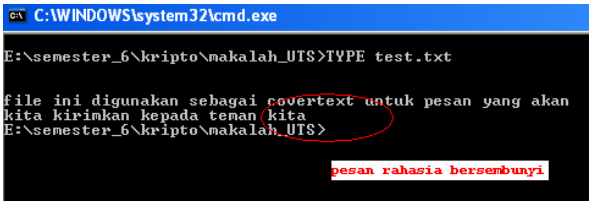
2.1.1 Peletakkan pesan rahasia setelah EOF

Sebenarnya, DOS menyimpan sebuah isi sebuah file tanpa menggunakan penanda EOF, sistem operasi ini hanya mengkalkulasi EOF melalui informasi direktori. Bagaimanapun, banyak aplikasi pembacaan teks berbasiskan DOS yang membaca file hingga karakter Control-Z. Hal ini menyebabkan terdapat sebuah end of file yang khusus didefinisikan oleh program – program pembaca teks bawaan DOS ini. Contoh aplikasi ini adalah TYPE, MORE, dan FIND. Ketiga aplikasi ini membaca file dalam mode teks. Sebaliknya, perintah COPY, XCOPY, dan yang lainnya membaca file dalam mode binary.

Prinsip dari penggunaan karakter control-Z disini adalah sebagai pembatas antara covertext dengan pesan rahasia. Jadi, jika kita membuka file teks dengan mode binary pada DOS dan kemudian kita gabungkan dengan sebuah pesan teks dengan secara eksplisit kita tulis karakter ASCII control-Z sebagai pembatas, hasil yang didapat adalah pesan tersebut akan tersembunyi di belakang end of file. Berikut adalah simulasi penggabungan pesan dimana sebuah pesan covertext digabung dengan pesan rahasia menggunakan pemisah atau *delimiter* sebuah karakter ASCII control-Z.



Gambar 1. end of file sebagai penanda akhir sebuah file



Gambar 2. Pesan rahasia tidak ditampilkan di layar

Gambar satu dan gambar dua memperlihatkan bagaimana caranya penyisipan pesan rahasia kedalam DOS text file. Steganografi ini sangat berguna sekali untuk para DOS programmer yang kesehariannya berkecimpung di dunia DOS.

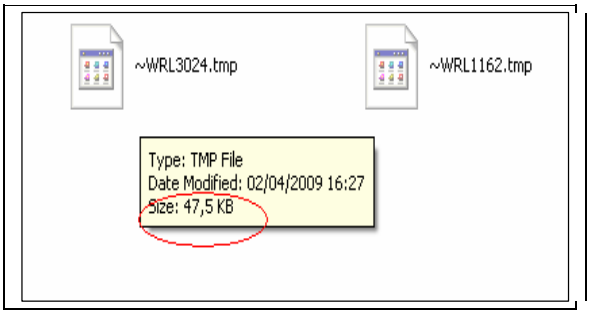
2.1.2 Rekayasa ukuran pesan rahasia

Permasalahan kemudian timbul ketika seseorang menyadari file teks yang ditampilkan di layar hanya sedikit, tetapi ukuran file tersebut terlihat besar. Orang tersebut mungkin akan berfikir ada yang tidak beres dengan file tersebut dan mencoba untuk mencari tahu. Akibatnya adalah peluang terbukanya pesan rahasia menjadi besar.

Masalah ini dapat kita redam dengan cara memperhatikan ukuran covertext yang kita gunakan. Misalkan, jika kita ingin menyembunyikan pesan sebesar 10 byte, kita dapat sembunyikan pesan ini kedalam covertext sebesar 100 kilobyte. Dengan begitu, orang sepertinya tidak akan memperdulikan pesan sebesar 10 byte dibandingkan dengan pesan sebesar 100 kilobyte. Jadi, metode ini menjelaskan sebuah nilai selisih antara ukuran covertext dengan pesan rahasia.

Salah satu metode penentuaan ukuran yang dapat kita gunakan untuk mendapatkan nilai ini adalah dengan memanfaatkan ketelitian nilai ukuran file yang ditampilkan pada file explorer di windows (dengan asumsi pengguna DOS berada di lingkungan windows). Satuan ukuran terkecil file yang digunakan pada file explorer adalah byte. Jadi, jika sebuah nilai mempunyai satuan byte, ia merupakan bilangan bulat. Satuan berikutnya adalah kilobyte. Jika kita perhatikan dan kita teliti satu persatu file yang mempunyai ukuran dengan satuan kilobyte, nilai ukuran tersebut hanya mempunyai ketelitian sampai satu angka di belakang koma.

Dengan adanya pengamatan ini, kita dapat memanfaatkan sisa jumlah byte yang tidak akan ditampilkan pada file explorer jika ditambahkan pada sebuah file berukuran kilobyte. Oleh karena itu, metode kita kali ini hanya akan berkutat seputar jumlah ukuran yang tidak akan ditampilkan di layar file explorer.



Gambar 3. Ketelitian satuan kilobyte pada tampilan file explorer windows.

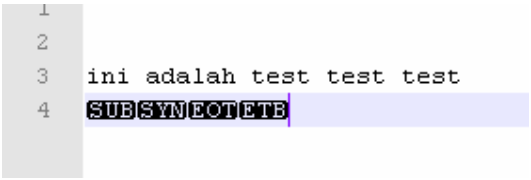
Sekarang, kita akan menyederhanakan solusi yaitu hanya dengan meneliti file dengan satuan ukuran file kilobyte. Jika ketelitian satuan kilobyte adalah satu angka di belakang koma, nilai ukuran terbesar yang tidak akan menghasilkan perubahan tampilan di file explorer adalah 99 Byte. Misal, sebuah ukuran file adalah 1500 Byte, file explorer akan menampilkan ukuran file tersebut dengan tampilan 1.5 KB. Jika kita menambah ukuran file tersebut menjadi 1599 Byte, layar pada file explorer masih menampilkan ukuran file dengan tampilan 1.5 KB. Jadi, panjang covertext disarankan lebih dari 1000 byte dengan penyisipan pesan rahasia mempunyai batas atas 99 byte. Dengan adanya ketetapan ini, kita dapat turunkan rumus untuk menentukan ukuran maksimal pesan rahasia,

$$PE <= 99 - (PC \bmod 100) \qquad (1)$$

Dimana PE adalah panjang pesan rahasia dan PC adalah panjang covertext. Jika kita menggunakan aturan ini, kemungkinan besar pesan rahasia kita tidak akan diketahui oleh pembaca pesan.

2.1.3 Enkripsi pesan rahasia

Setelah kita memperkuat metode steganografi kita dengan rekayasa ukuran file, kita juga dapat memperkuat lagi sistem steganografi kita dengan meng-enkripsi pesan rahasia tersebut. Kita dapat memilih salah satu algoritma enkripsi yang kita suka untuk mengenkripsi pesan rahasia tersebut. Hal ini mengakibatkan tingkat keamanan metode steganografi ini meningkat. Dengan adanya enkripsi pesan rahasia yang diletakkan di belakang ini akan terlihat seperti kumpulan huruf kacau. Sebagian besar orang akan mengira hal ini merupakan kesalahan *system error* atau *disk error*.

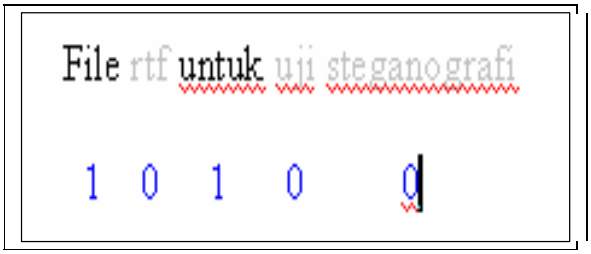


Gambar 4. Contoh hasil file yang pesan rahasianya di enkripsi

2.2 Steganografi pada rich text format file

2.2.1 Mode steganografi

Mode steganografi yang dapat kita gunakan dalam menyembunyikan pesan di rich text format antara lain dengan menggunakan representasi bit 1/0. Artinya, data yang kita simpan direpresentasikan dalam bentuk stream 1/0. Kita juga dapat merepresentasikan sebuah bit dengan sebuah huruf atau sebuah kata. Bit – bit ini nantinya diabstraksi dengan sebuah rekayasa misalkan dengan rekayasa warna. Misalkan, warna hitam menyatakan bit 1 dan warna abu-abu menyatakan bit 0.



Gambar 5. Contoh representasi bit pada steganografi di rich text format.

2.2.2 Rekayasa atribut warna

Kita dapat memanfaatkan atribut warna huruf untuk melakukan steganografi pada file rich text format. Dasar yang digunakan disini adalah dengan memilih dua warna dengan representasi nilai berbeda di komputer tetapi kedua warna tersebut sulit dibedakan oleh mata biasa. Saya asumsikan, mode yang digunakan disini adalah mode bit, jadi kita hanya membutuhkan 2 buah warna saja. Rich text format menggunakan aturan dengan format tertentu untuk menyatakan warna sebuah blok. Blok yang dimaksud disini bisa merupakan huruf atau kata. Satu blok merepresentasikan satu huruf atau kumpulan huruf yang mempunyai nilai atribut yang sama (dalam kasus ini adalah atribut warna). Contoh potongan format text untuk pengaturan warna adalah sebagai berikut,

```
{\cf2\insrsid9269114\
charrsid13181445
uji}{\insrsid9269114
```

```
}{\cf6\insrsid9269114\
charrsid859581
steganografi}{\insrsid1849726
\par }
```

Gambar 6. Contoh representasi rich text format file

Kunci yang menjadi warna pada format diatas adalah pada teks cf2, cf6, cf5, dan yang lainnya. Salah satu teknik steganografi yang dapat digunakan disini adalah dengan cara mengatur warna yang berdekatan dengan warna hitam. Kita gunakan nilai cf18, cf19, dan cf20. Kita menggunakan ketiga nilai ini karena ketiga nilai warna ini sangat dekat dengan warna hitam dan sulit dibedakan oleh mata manusia. Misalkan, cf19 merepresentasikan bit 0 dan cf20 merepresentasikan bit 1, berikut adalah contoh steganografi dengan atribut warna.

```
{\cf19\insrsid9269114\charrsid
859581    File}{\insrsid9269114
}{
\cf20\insrsid9269114\charrsid1
3181445    rtf}{\insrsid9269114
}{\cf19\insrsid9269114\charrsi
d859581  untuk}{\insrsid9269114
}{\cf20\insrsid9269114\charrsi
d13181445  uji}{\insrsid9269114
}{\cf19\insrsid9269114\charrsi
d859581
steganografi}{\insrsid1849726
\par }}\par }
```

Gambar 7. Contoh bentuk steganografi di rich text format file.

File rtf untuk cf19uji steganografi

Gambar 8. Tampilan dokumen hasil penyisipan pesan rahasia.

Kita dapat lihat pada gambar 7 bahwa warna tulisan sangat sulit dibedakan. Kita hanya dapat mengenal semua itu berwarna hitam. Padahal, tulisan tersebut mengandung sebuah pesan rahasia yaitu biner 01010.

2.2.3 Rekayasa atribut ukuran huruf

Selain atribut warna kita juga dapat menggunakan atribut ukuran huruf untuk melakukan steganografi. Prinsip yang digunakan tidak jauh berbeda dengan teknik yang menggunakan atribut warna huruf, yaitu menggunakan penyisipan nilai atribut yang tidak menyebabkan perubahan tampilan secara signifikan pada layar.

```
{\fs36\insrsid5339402\
charrsid11014184 untuk}
```

Gambar 9. Atribut ukuran huruf

```
{\fs27\insrsid5339402\charrsid11014184
File}
{\insrsid5339402  }{
\fs28\insrsid5339402\charrsid5339402 uji}
{\insrsid5339402  }
{\fs28\insrsid5339402\charrsid11014184
untuk}{\insrsid5339402
}{\fs27\insrsid5339402\charrsid5339402
steganografi}{\insrsid1849726
\par }}
```

Gambar 10. Contoh steganografi dengan atribut ukuran huruf.

File uji untuk steganografi

Gambar 11. Dokumen hasil steganografi.

Kita dapat lihat pada gambar 10. sulit sekali untuk mata manusia membedakan ukuran huruf yang ada. Manusia pasti akan menganggap tidak ada keanehan dengan dokumen ini.

Kode fs36 menyatakan ukuran hurufnya adalah 13. jadi, nilai kode fs adalah 2 kali dari nilai ukuran huruf. Sekarang, jika kita menggunakan dua buah kode fs yaitu fs24 dan fs23, dimana fs23 merepresentasikan bit 0 dan fs24 merepresentasikan bit 1, kita akan menghasilkan tampilan yang sulit dibedakan ukuran hurufnya.

3. IMPLEMENTASI

Implementasi program untuk steganografi pada DOS text file disarankan menggunakan bahasa pemrograman dengan level abstraksi yang rendah seperti C/C++. Bahasa pemrograman ini dapat dengan cepat beroperasi untuk pembacaan dan penulisan file. Program ini nantinya akan menerima 2 buah parameter yang bertipe FILE. File

pertama adalah menyatakan isi covertext sedangkan file yang kedua menyatakan pesan rahasia. Kemudian, program akan membaca seluruh karakter yang ada di file pertama dilanjutkan dengan menulis karakter end of file control-Z. Setelah program menulis karakter end of file, barulah program menuliskan pesan rahasia tersebut. implementasi program ini dapat dilihat di http://students.itb.ac.id/~alfan_fw/code_stegano/.

```
main(argc,argv)   /* ....this program
lists a file in binary mode; */
                  /*     ALL characters
are shown.                  */
     int    argc;
     char  *argv[];
{
  int byte;
  long count=0l;
  FILE *infl;
  if ( argc == 2 )
    {
          infl = fopen(argv[1],"rb");
          if ( infl != NULL )
            {
                while( ( byte =
getc(infl) ) != EOF )
                    {
                          count++;
                  /*      if ( byte !=
26 ) remove these comment marks to

avoid display of cntrl-Z. Writing

cntrl-Z to stdout terminates the

program on some systems.        */

putc(byte,stdout);
                    }
               close(infl);
            }
          else
            {
              printf("\n\n...cannot
open input file...\n\n");
            }
    }
  else
    {
       printf("\n\n....must supply one
file name...\n");
    }
}
```

Gambar 12. Contoh potongan kode untuk simulasi pembacaan file sampai EOF control-Z

```
main(n,params)
  int n;
  char *params[];
{
  FILE *thefile,*thedata;
  int ch=0;
  if ( n == 3 )
    {
       thefile = fopen( params[1], "ab"
);
       thedata = fopen( params[2], "rb"
);
       if ( thefile != NULL && thedata
!= NULL )
         {
           fseek(thefile, 0, 2);
           putc(26,thefile);
           while ( (ch = getc(thedata))
!= EOF )
             {
                    putc(ch,thefile);
             }
         }
       else
         {
           printf("\ncannot open a
file..\n");
         }
       fclose(thedata);
       fclose(thefile);
    /*   remove(params[2]);   option to
erase the source of the hidden data */
    }
  else
    {
      printf("\nproper format
is:\nlappend <file appended to> ");
      printf("<file to append>\n");
    }
}
```

Gambar 13. Contoh potongan kode untuk proses steganografi pada DOS text file untuk proses penyisipan pesan ke dalam covertext.

implementasi program steganografi pada rich text format file membutuhkan sebuah struktur data penampung yang merepresentasikan kumpulan bit yang ingin disembunyikan. setelah itu program mempunyai sebuah mesin yang dapat memabaca representasi internal file RTF dan mampu memilah gabungan huruf yang merupakan atribut file seperti blok cf6, fs20, dan yang lainnya. program menjalankan aksinya dengan cara melakukan pemerikasaan file dari awal sampai akhir, jika ia menemukan blok huruf cf dan fs, ia kemudian akan

menimpa angka yang ada di belakangnya sesuai angka yang sudah kita definisikan untuk bit 0 atau bit 1. Proses ini akan dilakukan terus hingga akhir pembacaan file.

4. KESIMPULAN

DOS text file memang jarang digunakan oleh orang biasa. Tetapi, para DOS programmer sering sekali menggunakannya. Oleh karena itu, jika seorang DOS programmer membutuhkan sebuah cara untuk melakukan steganografi pada DOS text file, ia dapat menggunakan metode end of file seperti yang sudah dijelaskan pada bagian sebelumnya. Tekniknya cukup sederhana, tetapi jika kita gunakan secara efektif dan efisien, kita akan mendapatkan hasil yang optimal.

Begitu juga dengan rich text format file, kita dapat memanfaatkan keunggulan representasi internalnya yang berupa file text berformat. Jadi, kita tidak perlu melakukan hal rumit seperti bermain dengan file binary. Kita hanya tinggal mempermainkan beberapa atribut pada file tersebut lalu mengubah nilainya. Aturan pembentukan nilai dilakukan sedemikian sehingga blok huruf yang mempunyai nilai atribut berbeda tidak dapat dibedakan oleh mata manusia ketika dibuka oleh aplikasi word processor.

REFERENSI

[1] Munir, Rinaldi. *Diktat Kuliah Kriptografi.*. Bandung 2006.