Steganography with AVI type Uncompressed DIB

فراس عبد الحميد العبيدي
مدرس مساعد/قسم الحاسبات–كلية بغداد للعلوم الاقتصادية

Abstract

In this system we explain the structure of the AVI file “specail case of RIFF file “ in some details and focusing on the frame type uncompress DIB “Device Independent Bitmap” and then explain the catgorayes of stganography system and how can make steganographic system using the AVI as a cover for the proposed system.

1. AVI overview

AVI Files are a special case of RIFF files. RIFF is the Resource Interchange File Format. This is a general-purpose format for Microsoft and IBM defined exchanging multimedia data types that during their long forgotten alliance.[1]

1.1 AVI and Windows Bitmaps (DDB, DIB)

Microsoft Windows represents bitmapped images internally and in files as Device Dependent Bitmaps (DDB), Device Independent Bitmaps (DIB), and DIB Sections. Uncompressed 'DIB ' AVI files represent video frames as DIB's. Various multimedia API's that work with AVI use Windows bitmapped images.[2]
Prior to Windows 3.0, Windows relied on Device Dependent Bitmaps for bitmapped images. A DDB is stored in a format understood by the device driver for a particular video card.

1.2 AVI File Format[3]

```
RIFF ('AVI '
       LIST ('hdrl'
             'avih'(<Main AVI Header>)
             LIST ('strl'
                   'strh'(<Stream header>)
                   'strf'(<Stream format>)
                   'strd'(additional header data)
                       .
                       .
                       .
             )

             .
             .
             .
   )
```

```
LIST ('movi'
            {SubChunk | LIST('rec '
                              SubChunk1
                              SubChunk2
                                 .
                                 .
                                 .
                        )

                .
                .
                .
            }

        .
        .
        .
   )

   ['idx1'<AVIIndex>]
)
```

1..2.1 The main AVI header

The file begins with the main header. In the AVI file, this header is identified with “avih” four-character code. The header contains general information about the file, such as the number of streams within the file and the width and height of the AVI sequence. Some information in the main header.

The **dwMicroSecPerFrame** field specifies the period between video frames.
The **AVIF_HASINDEX** Indicates the AVI file has an “idx1” chunk.
The **AVIF_COPYRIGHTED** Indicates the AVI file contains copyrighted data.
The **dwTotalFrames** field of the main header specifies the total number of frames of data in file.
The **dwWidth** and **dwHeight** fields specify the width and height of the AVI file in pixels.

1.2.2 The Stream Header (“strl”) Chunks

One or more “strl” chunks follow the main header. (A “strl” chunk is required for each data stream.) These chunks contain information about the streams in the file. Each “strl” chunk must contain a stream header and stream format chunk. Stream header chunks are identified by the four-character code “strh” and stream format chunks are identified with the four-character code “strf”. In addition to the stream header and stream format chunks, the “strl” chunk might also contain a stream data chunk. Stream data chunks are identified with the four-character code “strd”.
Some information in stream hedear
The stream header specifies the type of data the stream contains, such as audio or video, by means of a four-character code. The fccType field is set to “vids” if the stream it specifies contains video data. It is set to “auds” if it contains audio data**.**

The **fccHandler** field contains a four-character code describing the installable compressor or decompressor used with the data.

1.2.3 The stream format (”strf”) chunks

A stream format (“strf”) chunk must follow a stream header (“strh”) chunk. The stream format chunk describes the format of the data in the stream. For video streams, the information in this chunk is a BITMAPINFO structure (including palette information if appropriate). For audio

streams, the information in this chunk is a WAVEFORMATEX or PCMWAVEFORMAT structure. (The WAVEFORMATEX structure is an extended version of the WAVEFORMAT structure.).

1.2.4 The stream data (“strd”) chunks

The “strl” chunk might also contain a stream data (“strd”) chunk. If used, this chunk follows the stream format chunk. The format and content of this chunk is defined by installable compression or decompression drivers. Typically, drivers use this information for configuration. Applications that read and write RIFF files do not need to decode this information. They transfer this data to and from a driver as a memory block.

1.2.5 The list (“movi”) chunks

Following the header information is a LIST “movi” chunk that contains chunks of the actual data in the streams; that is, the pictures and sounds themselves. The data chunks can reside directly in the LIST “movi” chunk or they might be grouped into “rec ” chunks. The “rec ” grouping implies that the grouped chunks should be read from disk all at once. This is used only for files specifically interleaved to play from CD-ROM.[2]
Like any RIFF chunk, the data chunks contain a four-character code to identify the chunk type. The four-character code that identifies each chunk consists of the stream number and a two-character code that defines the type of information encapsulated in the chunk. For example, a waveform chunk is identified by a two-character code of “wb”. If a waveform chunk corresponded to the second LIST “hdrl” stream description, it would have a four-character code of “01wb”.

A data chunk for an uncompressed DIB contains RGB video data. These chunks are identified with a two-character code of “db” (db is an abbreviation for DIB bits). Data chunks for a compressed DIB are identified with a two-character code of “dc” (dc is an abbreviation for DIB compressed). Neither data chunk will contain any header information about the DIBs. The data chunk for an uncompressed DIB has the following form: [5]

```
DIB  Bits    '##db'
     BYTE    abBits[];
```

The data chunk for a compressed DIB has the following form:

```
Compressed DIB    '##dc'
     BYTE          abBits[];
```

1.2.6 The (“idx1”) chunks

AVI files can have an index chunk after the LIST “movi” chunk. The index chunk essentially contains a list of the data chunks and their location in the file. This provides efficient random access to the data within the file, because an application can locate a particular sound sequence or video image in a large AVI file without having to scan it.

1.2.7 Other data chunks

If you need to align data in your AVI file you can add a “JUNK” chunk. (This chunk is a standard RIFF type.) Applications reading these chunks ignore their contents. Files played from CD-ROM use these chunks to align data so they can be read more efficiently. You might want to use this chunk to align your data for the 2 kilobyte CD-ROM boundaries. The “JUNK” chunk has the following form:

```
AVI Padding    'JUNK'
   Byte        data[]
```

For more details about the AVI structure see appendix A

2.Steganography

many different steganographic methods haven proposed during the last few years, these methods depends on different way how to hide information in proper cover .

There are several approaches in classifying steganographic systems, one could categorize them according to the type of covers used for secret communication, a classification according to cover modifications applied in the embedding process is another possibility, we want to follow the second approach and group and group steganogrphic method in five category. [6]

1. Substitution systems substitute redundant part of cover with a secret message.
2. Transform domain techniques embed secret information in a transform space of the signal " frequency domain"
3. Spread spectrum techniques adopt idea from spread spectrum communication.

4. Distortion techniques store information by signal distortion and measure the deviation from the original cover in the decoding step.
5. Cover generation methods encoded information in the way a cover for secret communication is create .

3. Propose algorithm

Name : steganographic with AVI.
Input : AVI and the message.
Output : message hiding in AVI.

3.1 Encoding algorithm

1. get the number of frames that want to hide information (F) "must less than number of frames in movie that take from main AVI header".
2. get the message or any digital data files (M)
3. count number of character in message (N)
4. if M is digital data file goto 6

else get the encode/decode table that contents of two column one for character and the other for the corresponding code, six bit for each character, not take in consideration in English that letter capital or small.

5. Encode the message with the encode/decode table (ST)
6. count number of byte in each frames (B)

a .get the horizontal and vertical number of pixel (H,V)

b. get the depth of color" 4,8,24 "bit (BI)

- Bb=V*H*BI in bit
- B = Bb / 8 in byte

7. a. C= N div F

b. C1 = N mod F

8. If M is digital data file Z=8 else z=6

9. if ((C_{+1} * Z) * 6) <= B then T1= True

Else must increase F

If ((C_{+1} * Z) * 6) * 2 <= B then T2= True

10. get the first frame and make it index frame

a. get the bit(7) from first byte and make it

0 for sequence hidden

1 for hiding only in odd block

b. Encode F as a 20 bits binary code and put it in next 20 blocks.

c. Encode C as a 10 bit binary and put in next 10 blocks.

d. Encode C1 as a 10 bit binary and put in next 10 blocks.

e. Encode the type of hidden data

0 for hiding mesasage

1 for hiding digital data file.

11. get the ST or the digital data file

for I = 2 to C1 do

each frame hide C+1 character either in sequence byte or in odd block IFF T2=True,

for I = C1 +2 to F+1 do

each frame hide C character either in sequence byte or in odd block IFF T2=True.

12. How hide each bit

a. in sequential byte

1. get the 7^{th} bit from the six bytes"block"and add to other

   for j = 1 to 6 do

   Q = Q + b(7)j

2. if Q equal the secret bit that want to hide then go to next bit from character

   else randomly choose one 7^{th} bit from six bytes and change it.

3. go to next six bytes.

b .In the odd block

1. get the 7^{th} bit from the six bytes"block" and add to other

   for j = 1 to 6 do

   Q = Q + b(7)j

2. if Q equal the secret bit that want to hide then go to next bit from character

   else randomly choose one 7^{th} bit from four byte and change it.

3. Skip the next six bytes “block” and go to step 1.

End of algorithm

This algorithm is symmetric in other station “Decode state”.

Conclusion

Some application need use movie each frame is index frame and high resolution whithotu any image process like in compression process and without and degradation in both humain vision system and statistical computation as in medical dignostic, We use the AVI movie uncompress

DIB frame as a cover to hide messages or any digital data file by using the proposed method and take in consideration the steganography parameter.

We can see the same size of the original AVI file and play it with out any sensative of the human vision

References

1.John McGowan's AVI Overview: Video and Image File Format Conversion
@ 2000 by John F. McGowan, Ph.D.

2. DV Data in the AVI File Format Specification, Version 1.01
Microsoft Corporation
June 25, 1997

3. John McGowan's "AVI Overview: Programming and Other Technical "
@ 2000 by John F. McGowan, Ph.D.

4. OpenDML AVI File Format Extensions
Last revision: February 28, 1996
Reformatting: September 1997

5. Video for Windows Programmer's Guide //microsoft site

6. stefan katzenbisser&fabien A.P.petircolas "information hiding techniques for steganography and digital watermarking " @2000 ARTECH HOUSE,INC

Appendix A

AVI is a specialization or "form" of RIFF, described below:

'RIFF' (4 byte file length) 'AVI '   // file header (a RIFF form)

'LIST' (4 byte list length) 'hdrl'   // list of headers for AVI file

The 'hdrl' list contains:

'avih' (4 byte chunk size) // the AVI header

```
                                                Offset/byte no.
    typedef struct {
        DWORD  dwMicroSecPerFrame;          32-35
        DWORD  dwMaxBytesPerSec;            36-39
        DWORD  dwReserved1;                 40-43
        DWORD  dwFlags;                     44-47
        DWORD  dwTotalFrames;               48-51
        DWORD  dwInitialFrames;             52-55
        DWORD  dwStreams;                   56-59
        DWORD  dwSuggestedBufferSize;       60-63
        DWORD  dwWidth;                     64-67
        DWORD  dwHeight;                    68-71
        DWORD  dwScale;                     72-75
        DWORD  dwRate;                      76-79
        DWORD  dwStart;                     80-83
        DWORD  dwLength;                    83-87
             } MainAVIHeader;
     End in 87 begin new 88
```

'strl' lists of stream headers for each stream (audio, video.) in the AVI file. An AVI file can contain zero or one video stream and zero, one, or many audio streams. For an AVI file with one video and one audio stream:

'LIST' (4 byte list length) 'strl'   // video stream list (a list)

The video 'strl' list contains as a chunk:

'strh' (4 byte chunk size)   // video stream header (a chunk)

```
                                                    Offset/byte no.
typedef struct {
    FOURCC  fccType;                                    108-111
    FOURCC  fccHandler;                                 112-115
    DWORD   dwFlags;                                    116-119
    DWORD   dwReserved1;                                120=123
    DWORD   dwInitialFrames;                            124-127
    DWORD   dwScale;                                    128-131
    DWORD   dwRate;                                     132-135
    DWORD   dwStart;                                    136-139
    DWORD   dwLength;                                   140-143
    DWORD   dwSuggestedBufferSize;                      144-147
    DWORD   dwQuality;     {from 0-10000}               148-151
    DWORD   dwSampleSize;                               152-155
} AVIStreamHeader;
```

'strf' (4 byte chunk size)   // video stream format (a chunk)

```
                                                    Offset/byte no.
typedef struct tagBITMAPINFOHEADER {
   DWORD  biSize;                                     172-175
   LONG   biWidth;                                    176-179
   LONG   biHeight;                                   180-183
   WORD   biPlanes;                                   184-185
   WORD   biBitCount;                                 186-187
   DWORD  biCompression;                              188-191
   DWORD  biSizeImage;                                192-195
   LONG   biXPelsPerMeter;                            196-199
   LONG   biYPelsPerMeter;                            200-203
   DWORD  biClrUsed;                                  204-207
   DWORD  biClrImportant;                             208-211
} BITMAPINFOHEADER;
```

'LIST' (4 byte list length) 'strl'   // audio stream list (a list)

The audio 'strl' list contains:

'strh' (4 byte chunk size) (data)   // audio stream header (a chunk)
'strf' (4 byte chunk size) (data)   // audio stream format (a chunk)

'JUNK' (4 byte chunk size) (data - usually all zeros) // an OPTIONAL junk chunk to align on 2K byte boundary

'LIST' (4 byte list length) 'movi'   // list of movie data (a list)

The 'movi' list contains the actual audio and video data.

This 'movi' list contains one or more ...

 'LIST' (4 byte list length) 'rec ' // list of movie records (a list)
 '##wb' (4 byte chunk size) (data)   // sound data (a chunk)
 '##dc' (4 byte chunk size) (data)   // video data (a chunk)
 '##db' (4 byte chunk size) (data)   // video data (a chunk)

A 'rec ' list (a record) contains the audio and video data for a single frame.
 '##wb' (4 byte chunk size) (data) // sound data (a chunk)
 '##dc' (4 byte chunk size) (data) // video data (a chunk)
 '##db' (4 byte chunk size) (data) // video data (a chunk)

The 'rec ' list is an optional list and may not be used for AVI files with only audio or only video data.

Steganography with AVI type Uncompressed DIB

فراس عبد الحميد العبيدي
مدرس مساعد/قسم الحاسبات–كلية بغداد للعلوم الاقتصادية

الخلاصة

في هذا النظام اوضحنا الهيكل العام الى الفايل نوع AVI والذي هو نوع خاص من الفايلات نوع RIFF وقد تم التركيز على اطر الهياكل من نوع DIB الغير مشفرة "الاجهزة الغير معتمدة على الصور من نوع Bitmap " ومن ثم تم توضيح انواع الاخفاء "steganography" وكيفية عمل اخفاء في AVI كغطاء في النظام المقترح

بعض التطبيقات بحاجة الى افلام تكون كل صورة فيها مرجع وكذلك الى دقة عالية وبدون أي معاجة صورية كتلكة التي تحدث اثناء ضغط الصور وبدون أي تقليل الى كفاءة الصورة من ناحية رؤية الانسان او الحسابات الاحصائية مثال على هذة التطبيقات هي التشخيصات المرضية