Data Hiding in MIDI File

Mohammed abed al-Hadi Daikh

Thi-Qar university-Education College –Computer Science Department

Abstract

Data hiding is the transfer of data or information between two points to prevent any not authorized person to access about them .This paper concerned with hiding the data in the file of the type of MIDI file (Musical Instrument Digital Interface) to prevent hackers from obtaining confidential data .To increasing the the data security , the data encrypted before hide them using RC5 algorithm . In this paper ,we use the last significant bit algorithm to hide the data and we depending on the value of velocity for each tone in order to hide largest possible amount of information , so the listener to a file cannot recognize if the file contains additional or factual information for velocity . from the experiences, we have seen that is we cannot hide the data or information in all the tone values that representing the MIDI file because we can notice the change on the file , but when we use the velocity values to hide, that there is no great difference between the file contain the hidden data and the original file where we made matching using the meldistance function .

Key word:- data hiding, MIDI file,Velocity,LSB

إخفاء البيانات في ملف الميديا

محمد عبد الهادي دايخ

جامعة ذي قار –كلية التربية للعلوم الصرفة قسم الحاسبات

الملخص

أخفاء البيانات هو نقل بيانات أو معلومات بين نقطتين **وعدم السماح لاي** شخص غير مخول من الاطلاع عليها . يهتم هذا البحث بإخفاء البيانات في الملفات من نوع Midi File لمنع المتطفلين من الحصول على البيانات السرية ولغرض زيادة السرية تم تشفير البيانات قبل إخفائها **باستخدام أحدى** خوارزميات التشفير . تم استخدام خوارزمية الثنائيات الأقل أهمية LSB في إخفاء البيانات وتم الاعتماد في الإخفاء على قيم (velocity)**من كل نغمة** وذلك لإخفاء اكبر كمية ممكنة من المعلومات **وكذلك عدم** إمكانية الشخص المستمع للملف على التمييز بان الملف يحتوي على معلومات أضافية غير المعلومات الحقيقية الخاصة ب(velocity) **،لاحظنا من خلال التجارب** انه لا يمكن إخفاء البيانات او المعلومات في كل القيم الممثلة لكل نغمة **في** ملف الميديا حيث سوف يمكن ملاحظة التغيير على الملف. **وبعد إجراء الإخفاء باستخدام** قيم (Velocity) **من كل نغمة** لوحظ لايوجد اختلاف كبير بين الملف الذي يحتوي على البيانات المخفية والملف الأصلي حيث تم عمل مطابقة بين **الملفين** باستخدام دالة meldistance .

الكلمات المفتاحية :إخفاء البيانات ، ملف الميديا , السرعة (velocity**) ، الثنائيات الأقل أهمية** LSB

1.Introduction

The process of embedding information into digital content without causing perceptual degradation is called data hiding. A special case of data hiding is steganography. Steganography is a useful technique for secure communication for hiding existence of communication itself.

It is achieved by insertion of secret messages into public communication or information media. Recently, steganography has been researched as an application of information hiding with prevalence of digitized multi-media contents [1], [2].

Information hiding, accomplished by exploiting a computer's file system and various other operating system characteristics can take on many forms. In many cases, information hiding is a intentional activity that an individual employs to store away sensitive information in an attempt to make it invisible to everyone else. However, there are some exceptions, such as digital watermarking, that are used for appropriate purposes [1].

This paper about data hiding in MIDI files. These files are popular musical files; however the main difference between this file and other popular files such as .wav and .mp3 is that the MIDI files are command files. They contain of commands for the music performance. This fact makes them much smaller than other types of files where the musical wave should be recorded.

The early method of embedding information in MIDI file would be to insert additional events (for example may be Meta-events)it would be easy to add events but this method is very problematic because the original file size would increase significantly[4]

The aim of this paper is to develop another MIDI data hiding method with high Capability to hide data. In this paper, we propose a MIDI hiding method technique which embeds information in Velocity. To get addition security the encryption RC5algorithm is used before data hiding by manipulating the last significant bit of velocity usually does not affect the user's perception of the object.

2. MIDI File Structure

The standard MIDI file(SMF) is a file format specifically designed to store the data that a sequencer records and plays (whether that sequencer be software or hardware based). This format stores the standard MIDI messages (i.e., status bytes with appropriate data bytes) plus a time-stamp for each message (i.e., a series of bytes that represent how many clock pulses to wait before playing the event). The format allows saving information about tempo, pulses per quarter note resolution (or resolution expressed in divisions per second, i.e, SMPTE setting), time and key signatures, and names of tracks and patterns. It can store multiple patterns and tracks so that any application can preserve these structures when loading the file. The format was designed to be generic so that any sequencer could read or write such a file without losing the most important data, and exible enough for a particular application to store its own proprietary, extra data in such a way that another application won't be confused when loading the file and can safely ignore this extra stuff that it doesn't need. Think of the MIDI file format as a musical version of an ASCII text file (except that the MIDI file contains binary data too), and the various sequencer programs as text editors all capable of reading that file. But, unlike ASCII, MIDI file format saves

data in chunks (i.e., groups of bytes preceded by an ID and size) which can be parsed, loaded, skipped, etc. Therefore, it can be easily extended to include a program's proprietary info. For example, maybe a program wants to save a flag byte that indicates whether the user has turned on an audible metronome click. The program can put this flag byte into a MIDI file in such a way that another application can skip this byte without having to understand what that byte is for. In the future, the MIDI file format can also be extended to include new official chunks that all sequencer programs may elect to load and use. This can be done without making old data files obsolete (i.e., the format is designed to be extensible in a backwardly compatible way). In conclusion, any software that saves or loads MIDI data should use SMF format for its data files. Standard MIDI files provide a common file format used by most musical software and hardware devices to store song information including the title, track names, and most importantly what instruments to use and the sequence of musical events, such as notes and instrument control information needed to play back the song[3].

A MIDI file always starts with a header chunk, and is followed by one or more track chunks. Some numbers in track chunks are represented in a form called a variable-length quantity. These numbers are represented 7 bits per byte, most significant bits first. All bytes except the last have bit 7 set, and the last byte has bit 7 clear. If the number is between 0 and 127, it is thus represented exactly as one byte. Here is the syntax of an MTrk (track) chunk:

<track data> = <MTrk event>+

<MTrk event> = <delta-time> <event>

<delta-time> is stored as a variable-length quantity. It represents the amount of time before the following event. If the first event in a track occurs at the very beginning of a track, or if two events occur simultaneously, a delta-time of zero is used. Delta-times are always present.

<event> = <MIDI event> | <sysex event> | <meta-event>

<MIDI event> is any MIDI channel message. Running status is used: status bytes may be omitted after the first byte. The first event in a file must specify status. Delta-time is not considered an event itself: it is an integral part of the specification. Notice that running status occurs across delta-times.

<meta-event> specifies non-MIDI information useful to this format or to sequencers, with this syntax:

FF <type> <length> <bytes>

All meta-events begin with FF, then have an event type byte (which is always less than 128), and then have the length of the data stored as a variable-length quantity, and then the data itself. If there is no data, the length is 0. As with sysex events, running status is not allowed.

3. Data Hiding Model

Data hiding is the technology to embed the secret information into a cover data, and to make the secret information

invisible. Figure (1) shows the general model of the information hiding [1]. The model consists of three processes, embedding, transmitting and extracting.

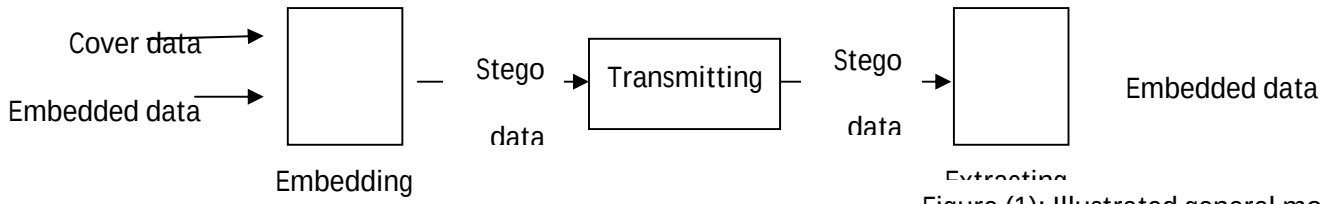


Figure (1): Illustrated general model of th

In this example, 3 bits are hiding. The last there bits from the velocity parameter were set to 011.

We divide the system to two layers for data hiding in MIDI file they are:

- **Information embedding**
- **Information extraction**

5.2.Powerful Encryption

We used RC5 algorithm to encrypt the message before hiding for further security purpose [6]. RC5 algorithm consist of three phases one for key expansion, encryption and decryption.These algorithm use the following three preemptive operations(and their inverses)

a. Two's complement addition of words denoted by "+",this is modulo-2^w addition.

b. Bit-wise exclusive-OR of words denoted by ⊕

c. A Left rotation (or 'left-spin') of words. The rotation of the word X left by Y bits is denoted X<<<Y.Only the log(w) low-order bit of Y are used to determine the rotation amount ,so that y is interpreted modulo w.

5.3. Information Embedding

Each tone of MIDI file is classified one or more notes ,in embedding procedure by the proposed system, information embedding is achieved with replacement the hiding message in velocity parameters .Take an MIDI files which plays for different time . It has more than 4KB . a matrix representation of note events in a MIDI file contains the information (onset (beats), duration(beats), MIDI channel and pitch ,Velocity, onset(sec) duration(sec)) we use LSB of the velocity parameters to embedding message Figure (2) illustrates note message structure. Information embedded to the file by changing velocity parameters.

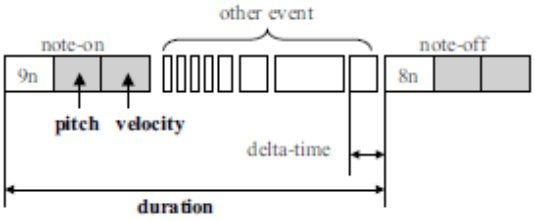


Figure (2): Illustrate note message structure

5.3.1.Embedding Algorithm :

Input: Encrypted message, cover MIDI fil

Step1.Read message

Step2:Convert the message to binary

Step3. Extract the velocity from each tone in MIDI file

Step4.Extract the LSB from velocity

Step5. While not end of the (secret message) do

4.1 Get next note

4.2 Convert velocity value to binary

4.2.2 Replace LSB velocity with bit from message

End {While}

End

5.4. Information Extraction

The procedure of information extraction is similar to the information embedding The proposed method uses extract LSB bit from each tone (from velocity value) for determination of embedding bit, based on size of message after extract hiding message perform Decryption The LSB bits to get the original character .

5.4.1.Extracting Algorithm:

Input: MIDI file

Step1. Extract velocity value from first tone from MIDI file

Step2. Extract the LSB from velocity.

Step3. while secret message not completed do

3.1 Get next note

3.2 Extract velocity value from tone

3.3 Extract LSB

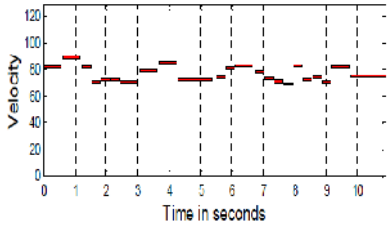
end{while}

Step4. Decode secret message bits

Output: Secret message

5. 5Velocity CHART

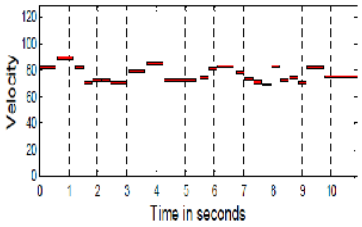
We use velocity chart to comparing the cover file(MIDI file) after and before hiding operations .To explain dissimilarity between them



File after hiding operation

Figure (3) illustrate Velocity CHART

Original file



6. I

Ir ıach

aga a hiding in MIDI file , We hiding data in multi Midi file using our method discussion in paper (using velocity value) and using all parameter of note value in midi file .We observe that the time to hiding data and extracting are minimal from MIDI file in our approach and when we hiding data in all parameter of tone, the file is changed and the listener can observe that .The Table(1) below show the statistics of the data hiding in value of velocity in midi file, the table (2) show the statistics of the

data hiding in all tone parameters value in midi file . Two comparing between two file(Source file and secret file) we used the function (meldistance.[10]) this function Calculates the dissimilarity between two Midi file in a particular representation.

Through the blow tables we note that difference ratio between two files is increase in case of hiding in all the values that represent each tone7. Conclusion

This proposed system is to provide a and efficient method for hiding the good data from hackers and sent to the destination in a safe manner. MIDI files are very popular for data hiding this make them good choice for this purposes .The embedding and extraction procedures are easy and fast. The hiding information with MIDI file in this manner cannot recognize

by listener. Also to increase security, we used the encryption system.

8. References

[1] B. Pfitzmann, “Information Hiding Terminology”, First International on Information Hiding, Workshop May 30 – June 1, 1996, Cambridge, UK, pp. 347-350.

[2] Rade Petrovi, Kanaan Jemili, Joseph M. Winograd, Ilija Stojanovi and Eric Metois, “DATA HIDING WITHIN AUDIO SIGNALS”, Lab, Series: June 15, 1999, MIT Media Electronics and Energetics vol. 12, No.2, pp.103-122.

[3] M. A. Raju, B. Sundaram, and P. Rao, TANSEN: “A Query-By- Humming Music Retrieval System”, In Proc. based National Conference on (NCC), 2003 Communications

[4] Matsumoto, Inoue, Kitabayashi, “An Information Hiding Method for Standard MIDI File”, Symposium on Cryptography and Information Security, SCIS2000-C03, Jan..

[5] Kotaro Yamamoto, Munetoshi Iwakiri , “An Information Hiding and Performance Rendering to Tempo " by Delta-time Control”, Proceeding of CSS2006, pp.549–554, 2006

[6]Biham.E and A.Shamir ,”A differential cryptanalysis of the data encryption standard”. Berlin:springer-verlay,1993

[7]Adli, A., Nakao, Z.: “Three Steganography Algorithms for MIDI Proceedings of 2005 Files”. In: International Conference on Machine Cybernetics, Learning and Guangzhou, China, August 2005, vol. 4, pp. 2404–2407 (2005)

[8] MIDI Manufacturers Association, http://www.midi.org

[9] Kotaro Yamamoto, Munetoshi Iwakiri : “A Steganography to Music Code With Adaptation in Musical Expression”, Transactions of IPSJ, Vol.47, No.8, pp.2724-2732, 2006.

[10] Tuomas Eerola & Petri Toiviainen,”MATLAB Tools for Music University of Research” Jyväskylä: Kopijyvä, Jyväskylä, Finland.

Electronic version available from: http://www.jyu.fi/musica/miditoolbox/

Table (1): below show the statistics of dissimilarity between two file (data hiding in value of velocity)

Source file	Secret file	Place data hiding	Time	Comparing result (original file with secret file)
Sample1	Sample11	Velocity value	25Sec.	0.220
Sample2	Sample22	Velocity value	25Sec.	0.11
Sample3	Sample33	Velocity value	30Sec.	0.234
Sample4	Sample44	Velocity value	30Sec.	0.220
Sample5	Sample55	Velocity value	25Sec.	0.61
Sample6	Sample66	Velocity value	25Sec.	0.323
Sample7	Sample77	Velocity value	30Sec.	0.33
Sample8	Sample88	Velocity value	30Sec.	0.12
Sample9	Sample99	Velocity value	25Sec.	0.44
Sample10	Sample1010	Velocity value	25Sec.	0.23
Sample11	Sample1111	Velocity value	30Sec.	0.44
Sample12	Sample1212	Velocity value	30Sec.	0.55
Sample13	Sample1313	Velocity value	25Sec.	0.55
Sample14	Sample1414	Velocity value	25Sec.	0.88
Sample15	Sample1515	Velocity value	30Sec.	0.45

Table (2) :Show the statistics of dissimilarity between two file (data hiding in all note parameters value in MIDI file)

Source file	Secret file	Place data hiding	Time(sec	Comparing result (original file with secret file)
Sample1	Sample111	All tone parameter	25Sec.	2.3
Sample2	Sample222	All tone parameter	27Sec.	4

Sample3	**Sample333**	**All tone parameter**	**30Sec.**	**1.9**
Sample4	**Sample444**	**All tone parameter**	**34Sec.**	**2.4**
Sample5	**Sample555**	**All tone parameter**	**25Sec.**	**3.4**
Sample6	**Sample666**	**All tone parameter**	**29Sec.**	**3.7**
Sample7	**Sample777**	**All tone parameter**	**30Sec.**	**3.560**
Sample8	**Sample888**	**All tone parameter**	**35Sec.**	**1.5666**
Sample9	**Sample999**	**All tone parameter**	**25Sec.**	**1.899**
Sample10	**Sample10101**	**All tone parameter**	**27Sec.**	**1.988**
Sample11	**Sample11111**	**All tone parameter**	**30Sec.**	**2.777**
Sample12	**Sample12122**	**All tone parameter**	**34Sec.**	**2.47**
Sample13	**Sample13133**	**All tone parameter**	**25Sec.**	**3.89**
Sample14	**Sample14144**	**All tone parameter**	**25Sec.**	**3.747**
Sample15	**Sample15155**	**All tone parameter**	**30Sec.**	**3.848**