A Steganographic Method on MP3

林裕欽
逢甲大學資訊工程研究
e-mail：m9430122@fcu.edu.tw

劉振緒
逢甲大學資訊工程研究所
e-mail：liuj@fcu.edu.tw

林士正
逢甲大學資訊工程研究所
e-mail：m9489593@fcu.edu.tw

摘要

在這篇論文中提出了一個應用於 MP3 檔案上資訊隱藏的方法。資料將會在數位音訊壓縮成為 MP3 時，嵌入經過壓縮的 MP3 檔案中。我們提出的方式是修改 MP3 中量化過程的部份來嵌入秘密資訊，他的代價是犧牲些許的失真性來嵌入資料，但這樣些微的失真性並不會造成聽覺上明顯的異常，並且也不容易由隱藏分析技術(Steganalysis)所發現。

關鍵詞：MP3、隱藏學、資訊隱藏

Abstract

We present a steganographic method for MP3 encoding. It embeds the secret data into MP3 files during the digital audio compression phase. Our proposed method modifies the quantization process in order to hide secret information. Our method introduces very little distortion that is undistinguishable in auditory sense and hard to be detected by steganalysis.

Keywords: MP3, Steganography, data hiding

1. 前言

隱藏學(Steganography)是一個希臘字，其意義是指“隱蔽寫入(covered writing)”，是資訊隱藏領域中的一種技術，將秘密訊息嵌入隱藏的目標，像是影像檔、音訊檔、文件檔或者是電腦的程式碼。

基本上這樣的嵌入資料的動作由特殊的技術完成，在無法得知隱藏技術所使用的方法的情況下，第三方將會難以從隱藏的目標中偵測或者移除嵌入於其中的資料。

一般而言隱藏學可以依據嵌入時的修改動作分類為六個類別。這六種類別分別為[16]：置換系統(Substitution systems)、轉換領域技巧(Transform domain techniques)、展頻技巧(Spread spectrum techniques)、統計法(Statistical methods)、失真技巧(Distortion techniques)及產生遮蓋物件法(Cover generation methods) [11]。此外，依據隱藏資訊的目標也可以大略的分為四種類別，分別是圖像(picture)、音訊(audio)、影像(video)及文件(document)。

隱藏學近幾年快速的成長因此導致許多的隱藏術工具的出現。有極大的部份都是在圖像方面的應用，例如：JPHide and JPSeek [15]，BlindSide Cryptographic Tool [7] 及 GIFShuffle [12]。而在音訊方面，MP3Stego [17]可以被使用在 MP3 的檔案上。在影像上， 則有 StegoVideo [18] 可以使用。在文件方面，SNOW [13] 則可以被用來這樣的檔案型態上而在 [16] 中則提到了“就目前所知的知識中，目前沒有出現任何方式可以偵測在文件檔案中的隱藏學”。此外，還有一些工具不只可以應用在特定的格式上，例如 wbStego 可以同時用在影像、文字文件、HTML 文件及 PDF 文件上 [4]。

MPEG/AUDIO Layer III，也就是所謂的 MP3，是目前在全球最熱門廣泛使用的數位音訊格式。尤其在網際網路上 MP3 是人們最常用來傳送音樂的格式，而當人們想要在電腦上聽些音樂時也總是使用 MP3，因為 MP3 是一種比原始音樂格式檔案要小的多的檔案格式同時擁有良好的音質，並且因為檔案較小，所以在網際網路上傳送格外方便。而且現今有許多的電子產品也都支援 MP3 檔案的播放，因此 MP3 目前遍佈了全球各地。

因為 MP3 的如此熱門，所以相當適合被利用來傳送秘密資訊到世上的每個地方給特定的人士。而當第三方接到這樣的 MP3 資料時因為沒有出現與其他 MP3 檔案有明顯的差異之處，因此難以產生懷疑，而且同時也難以將其中嵌入的資訊加以移除在不造成 MP3 品質損壞的前提下。

在目前，MP3Stego 是最為人所知 MP 隱藏術的工具。他將秘密資訊隱藏在 MP3 壓縮時的過程中，利用 Secure Hash Algorithm 1 (SHA-1 [8])當作位元亂數產生器，其所產生的亂數將決定資訊隱藏的位置。任何人都可以將 MP3 反解壓縮之後再重新壓縮，而這些嵌入的位元就會因此消失，這是目前眾所皆知的攻擊方式，但是這樣的代價將是造成嚴重的品質損失。

在這篇論文中提出了一個新的方法去完成 MP3 上的隱藏學。這個方法是利用修改編碼時用來對資料進行編碼的霍夫曼表的選擇。當 MP3 檔案解碼時，將會對被霍夫曼編碼的資料由對應的霍夫曼表進行解碼之後取得正確的音訊資訊。藉由霍夫曼表的編號，在編碼時將可嵌入秘密訊息，隱藏的位置則由 SHA-1 當作位元亂數產生器所得的資料做選擇，解碼時一樣使用 SHA-1 得出嵌入資訊的所在位置藉此取出嵌入的秘密訊息。

針對隱藏學的偵測技術被稱為隱藏偵測與分析(Steganalysis)，這是在 1998 年首先由 Johnson 及 Jajodia 所提出 [10]。這樣的技術可以被分成五種類別：特定隱藏學的攻擊法(Attack for specific Steganography)、視覺攻擊法(Visual Attack)、統計攻擊法(Statistical Attack)、位元選擇攻擊法(Bit Selection)及進階偵知技術(Advanced Steganalysis)。目前大多數存在的技術都使利用統計特性去找尋當秘密資訊嵌入過程中所留下的一些特性。

我們所提出的方法可以在 MP3 壓縮過程中嵌入秘密資訊在 MP3 檔中，其資訊隱藏效率比現有的 MP3Stego 要好，同時藏入資訊所造成的失真影響也同樣不會輕易的被發現。所以使用這樣的方法將可以在 MP3 檔案中藏入不少的資訊達到資訊隱藏的目的，同時被隱藏 MP3 檔案中也不容易被察覺藏有秘密資訊。

2. MP3 - Overview

2.1 MP3 編碼

MP3 是一種專用於音訊方面的壓縮格式，MPEG/audio 壓縮演算法是第一個高度傳真性(high-fidelity)的數位音訊壓縮國際標準 [1]。可以減少用來對 pulse code modulation 也就是所謂的 PCM 這種數位音訊表現的使用空間，藉由使用聲響心理模型(psychoacoustic model)除去難以被人類所聽見的音訊成分，並且以有效率的方式重新組織剩餘的部分。

Fig. 1 及 Fig. 2 是 MP3 編碼的流程圖，主要包含四個部分 [9] [5]：

1. **濾波器排(Filter bank)**：一個混和式的濾波器排用來分解輸入的訊號，在經過這個濾波器頻帶之後，將資料分解成為頻帶用以做其後的計算。
2. **感知模型(Perceptual model)**：使用聲響心理模型中已知的規則去評估遮罩門檻，藉由使用遮罩門檻(masking threshold)決定音訊中的資訊哪些是需要去除與或者保留。
3. **量化及編碼(Quantization and coding)**：對頻帶進行量化及編碼，需確保量化後的值低於遮罩門檻值。
4. **位元流的編碼(Encoding of the bitstream)**：產生位元流的格式資料，其中包含量化及編碼後的頻線(frequency lines)及 Side_Information。

在下面的部份將會進一步的對整個 MP3 的壓縮作更進一步詳細的描述。

2.2 壓縮概觀

在 MP3 壓縮中首先取得輸入的 PCM 數位音訊訊號進入濾波器排，藉由濾波器排將輸入的訊號分成 32 個等寬子頻帶(subbands)，之後將每個在時域(time domain)子頻帶中的取樣由改良式離散餘弦轉換(Modified Discrete Cosine Transform, MDCT)的方式轉換對應到頻域(frequency domain)中。同時將相同的輸入資料也輸入到快速傅立葉轉換(Fast Fourier Transform, FFT)中經過轉換後的資料給予聲響心理模型去決定每一個子頻帶的訊號遮罩比(signal-to-mask ratio, SMR)。在失真控制的部分，利用聲響心理模型所計算出來的訊號遮罩比去決定如何給定在子頻帶量化時可用的編碼位元，使得可聽見的量化雜訊最小。而這些量化過後的子頻帶取樣會再用無失真的霍夫曼編碼進行編碼，最後這些編碼後的子頻帶取樣及 Side_Information 將被包裝成為符合 MPEG/Audio standard [1] 的位元流如 Fig. 3 所示，其中的編碼框(frame)是位元流中的最基本

單位，其架構如 Fig. 4 所示，而每個編碼框中依據音訊類型而包含有兩個或者四個 granules，其架構如 Fig. 5 所示，一個 granule 是編碼過程中最基本的單位，相關這部份的細節將在編碼過程介紹過後再做細部的介紹。

以下將以 MP3 壓縮的順序做進一步的說明在壓縮過程中的步驟及其內容。

2.2.1 分析率波器排(Analysis Filter Bank)

這是編碼過程中的第一個步驟，他將音訊資料切割成為 32 個等寬子頻帶，一個編碼框將會包含 1152 個 PCM 的取樣，所以其中每一個子頻帶含有 36 個子頻帶取樣，之後這些資料再送往改良式離散餘弦轉換進行處理。

2.2.2 聲響心理模型(Psychoacoustic Model)

輸入的訊號資料不僅僅只送往分析率波器排中，同時也必須輸入聲響心理模型中。MP3 之所以可以同時擁有高壓縮比及高音訊品質都是藉由聲響心理模型來達到這樣的特性。在資料進入聲響心理模型之前，需先經過快速傅立葉轉換而後再進入聲響心理模型進行人類聽覺的模擬計算。藉由利用人耳的聽覺特性，聲響心理模型將會遮罩人耳無法聽見的非必要的音訊資料，因此部分的音訊資料將會被移除，因為如此並不會導致人耳聽覺上的失真，同時可以節省音訊資料所需的空間大小。此外，在聲響心理模型中，也計算了改良式離散餘弦轉換中的窗框選擇(windowing)，以及在量化過程中的失真控制參數。

2.2.3 改良式離散餘弦轉換(MDCT)

改良式離散餘弦轉換是在原本的離散餘弦轉換之前再加上窗框處理，並且在最後再加上假象處理(anti-aliasing)，窗框的選擇會依據不同的取樣音訊資料而定，找出適合的窗框型態對該取樣進行處理，選擇的方式則由聲響心理模型中所計算的參數作窗框的選擇決定。在訊號經過改良式離散餘弦轉換後會將取樣資料由時域轉換成為頻域以得到頻線資料，之後再進行量化及編碼。

2.2.4 量化及編碼(Quantization and Coding)

在經過改良式離散餘弦轉換之後，將會得到頻線資料作為輸出，這些頻線將經過三層迴圈進行量化及編碼。這三層迴圈分別是疊代迴圈(iteration loop)、外部迴圈(outer loop)及內部迴圈(inner loop)。在疊代迴圈中，將會計算可用的位元數並且設定每一個外部迴圈的初始值。在每一次外部迴圈執行過後，計算剩餘的位元數將他存在儲藏處中。在外部迴圈中，最重要的工作是失真的控制，當每一個內部迴圈完成量化之後，外部迴圈將會計算失真大小，控制其在可容忍的範圍內，否則則要求重新執行內部迴圈。而內部迴圈中，頻線將會被量化並且需確保沒有溢位的出現，同時不能讓編碼後的長度超過原先預期的編碼可用位元數。

最後，每一個內部迴圈的資料組成一個對應到 granule 的 Main_Data 區塊，如 Fig. 5 所示，並且依據低頻、中頻及高頻分成三個部分，依序對應到 big_value、count_1 及 zero_region 中。

1. **big_value**：這個部分可以被 region1、region2 及 region3 所組成，每個區域都會使用霍夫曼表進行編碼，每一個區域所用霍夫曼表都可以不同。在這個區塊中以兩個量化後的頻線為一組，其霍夫曼表可編碼的最大值為 8206。
2. **count_1**：在這邊有兩個霍夫曼表可供選擇，這邊每一個量化後的頻線的最大值是 1。而在這邊每四個量化後的頻線一組作一個編碼，其霍夫曼表可編碼的最大值為 15。
3. **zero_region**：而這個部份所有的量化後的頻線皆為 0，因此這個部份不需要編碼，因為這個部份的長度可以由另外兩個部份的長度求出。

總共有 34 個霍夫曼表可以用來進行編碼，如 Tab. 1 及 Tab. 2 所示，其中編號 0 到 31 用在 big_value 中，編號 32 及 33 則被用在 count_1 這個部份。而這些霍夫曼表之間的不同處在於他的最大值及統計特性不同，選擇最適合的霍夫曼表用來將編碼作最佳化。

2.2.5 位元流的格式化

最後，將先前經過計算的資料格式化之後將可以得到位元流，也就是一般所用的 MP3

檔案，其為數個編碼框所串接而成，如 Fig. 3，而每個編碼框在單聲道(channel)的狀況下將含有兩個 granules；雙聲道狀況下則為四個 granules，每個聲道各有兩個。

為了更清楚的說明 MP3 的檔案格式，以下將 MP3 的檔案格式由上到下分成位元流、編碼框及 granule 三個部份說明：

位元流

如 Fig. 3 所示，其由一個一個的編碼框所依序串接而成，其所包含的編碼框數量主要由音訊資料的時間長度決定，當音訊資料時間較長則會包含較多的編碼框，其中的每一個編碼框代表的皆是相同的時間長度的音訊資料。所以一個位元流中所包含的編碼框數量可由全部的音訊時間長度除以一個編碼框所代表的時間長度算出該時間長度的位元流需要由多少個編碼框所組成。

編碼框

如 Fig. 4 所示，基本上而言，每個編碼框有一個標頭(Header)、錯誤偵測碼(CRC)、Side_Information、Main_Data 這四個部份。每一個標頭含有 32 個位元，在標頭中的前 11 個位元是連續的 11 個位元 1 用以表示一個編碼框的開始，其後則是對應於每個編碼框的相關資訊，包含位元率(bit rate)、保護位元(protection bit)、取樣率(sample rate)等等。當標頭中的保護位元設為 0 則在編碼框中才會在標頭之後緊跟著 16 位元的錯誤檢查碼，反之則無。Side_Information 如果是單聲道的情況下則有 136 個位元長度，雙聲道則為 256 個位元長度，其中所包含的是關於 Main_Data 的解碼資訊，如：不同編碼區塊對應其霍夫曼表以及其所選擇的改良式離散餘弦轉換的窗框選擇型態的資訊等。在 Main_Data 中包含比例因子(scalefactor)及量化和位元分配後被霍夫曼表編碼後的頻線資料。而每個編碼框代表固定時間長度的音訊資料，由 1152 個取樣所表示，編碼框的時間長度可以由下算式算出：

$$time(\text{sec}onds) = 1152 \times Samplerate$$

其中 1152 代表的就是一個編碼框中固定含有的取樣數量，之後再乘上取樣頻率，也就是一個取樣所代表的時間長度，即可算出一個編碼框所代表的時間長度。一般來說常見的取樣頻率是 44100 Hz，換句話說，一個取樣表示 1/44100 秒，所以在取樣頻率為 44100 Hz 的 MP3 檔案中一個編碼框所代表的時間長度約為：

$$1152 / 44100 = 26ms$$

因此，一個編碼框所包含的時間長度是由取樣頻率所決定。 當取樣頻率越高，則一個編碼框所代表的時間長度相對較少，所以相同時間長度的音訊資料再取樣頻率較高時，會產生較多的編碼框。而在編碼框中，當為單聲道中的音訊資料時將會在 Main_Data 中含有兩個 granules 的資料；當為雙聲道時則左右聲道各有兩個 granules，所以共包含四個 granules。

granule

如 Fig. 5 所示，其所在位置是在編碼框中的 Main_Data 區塊。一個 granule 是編碼過程中編碼的最小單位，其一個 granule 所包含的是該編碼框中一半的取樣個數，也就是說在一個 granule 中將會含有 576 個取樣，這些資料被轉換為頻線，再經過量化及編碼後存放在 Main_Data 中對應於 granule 的部份。所以每個 granule 表示的是一個編碼框中的一個聲道，其中經過轉換、量化及編碼過後的二分之一數量的取樣資料。所以對單聲道的音訊檔案而言，在編碼框的 Main_Data 中，有兩個 granules 各自包含 576 個取樣，因而組成在一個編碼框中 1152 個取樣資料；而在雙聲道中，每一個聲道各自有兩個 granules 同樣各自包含 576 個取樣，所以共有四個 granules，組成編碼框中的兩個聲道各自包含 1152 個取樣資料。

2.3 MP3 解碼

解碼部份相較於編碼而言較為容易，基本流程如 Fig. 6 所示。在解碼過程中位元流中的 Main_Data 將會利用紀錄在 Side_Information 的霍夫曼表的編號進行解碼，然後對解碼後的資料進行反量化，將反量化過後的資料在經過反向的改良式離散餘弦轉換(inverse MDCT, IMDCT)，最後將資料進行合成之後就可以取得 PCM 音訊訊號。

3. MP3Stego 簡介

MP3Stego 是現今所知最聞名的 MP3 隱藏術的工具。他是一種在 [3] 中所提到的 parity of bit 的實做例子。他將秘密資訊隱藏在 MP3 壓縮時的過程中，隱藏的過程發生在 MPEG Layer III 編碼過程中的核心部份，稱為內部迴圈。在內部迴圈中，將對輸入的資料進行量化，且不斷的增加量化的 stepsize 直到量化後的資料所用的位元數低於可用的位元數的情況下才可以完成編碼。而在編碼後的 Side_Information 中含有一個 part2_3_length 變數，其表示包含在一個編碼框中的 Main_Data 的位元長度，其中包含有該編碼框中的比例因子及霍夫曼編碼過後的量化後的頻線資訊。MP3Stego 將嵌入的位元視為 part2_3_length 的同位碼，藉由改變內部迴圈中的迴圈結束條件將資料嵌入。而選擇嵌入的位置則是用 SHA-1 當作位元亂數產生器所產生的亂數來決定。雖然將 MP3 反解壓縮之後再重新壓縮，可將這些嵌入的位元移除，但這樣的代價將是造成嚴重的品質損失，因此嵌入其中的秘密資訊因此得到保障無法輕易的被移除。

4. 隱藏的方法

在這篇論文中提出了一個利用改變對頻線編碼時所選的霍夫曼表來嵌入秘密資訊，原先這些霍夫曼表的選擇原則是依據其霍夫曼表所能表示的最大值及統計特性做出適當的選擇以得到最佳化的編碼。選擇最合適的霍夫曼表將使得 MP3 檔案在有限空間下提供最佳的聲音品質，改變這些霍夫曼編碼則會對整體檔案造成改變也可能造成些微的失真相較於原先未嵌入資訊的編碼方式。因此我們提出藉由修改選擇的霍夫曼表用以在 MP3 中嵌入秘密訊息。

因為在 MP3 壓縮中使用了 34 個霍夫曼編碼，其中前 32 個編號由 0 到 31 的霍夫曼表用在 big_value 而編號 32 及 33 則用在 count_1 中對量化後的頻線作編碼。不同的霍夫曼表之間有不同的可用最大編碼值及不同的統計特性。在編碼時，當需要編碼的大小可以被該霍夫曼表所表示時，選擇其中可用較少位元數完成編碼的霍夫曼表進行編碼；如果選擇的霍夫曼表並不是其中使用最少位元完成編碼者，則可能會造成些許失真，但是並不會明顯的被人類的聽覺所發現。因此在編碼的過程中，改變選擇的霍夫曼表的編號，用來在 MP3 中嵌入秘密位元，而在解碼過程中，藉由讀取霍夫曼表的編號，將可以取出對應於該霍夫曼表編號所表達的訊息，因此取得隱藏於其中的嵌入訊息。

雖然 MP3Stego 提出一種好用的 MP3 的資訊隱藏術，但是在 MP3Stego 程式實際的使用上，他只能使用在 16 位元的 PCM、單聲道且取樣頻率為 44100 Hz 的 wav 檔案上，在其他格式的檔案使用上則無法確保可以正確順利的完成工作，也無法確定完成之後可以將檔案取出。然而在這篇論文中提出的隱藏術的方法則可以被使用在一般性的音訊檔案上，在將其壓縮成為 MP3 時嵌入秘密資訊，同時也將會比 MP3Stego 擁有更好的隱藏效率並且也同樣不容易被第三方所偵測或者移除。

4.1 霍夫曼表

用於 big_value 的霍夫曼表的編號為 0 到 31，其中的資料如 Tab. 1 所示。 從其中可以看到，編號 0 到 15 的最大值為 15，編號 16 到 31 在不使用 linbits 的情況下最大值也是 15，當編碼的值超過 15 則使用 linbits 進行編碼，其表示的值如下所示：

$$\text{最大值} = 15 + 2^{\text{linbits}} - 1$$

顯示在 Tab. 1 中編號 16 到 31 的最大值是使用 linbits 之後的值，編號 0 到 23 中的最大值逐漸遞增，編號 24 到 31 的最大值也是逐漸遞增。

而用於 count_1 中的霍夫曼編號為 32 及 33，其中的資料如 Tab. 2 所示，這兩個表的最大值皆為 15。

4.2 隱藏術的方法

為了避免輕易的被偵測出異常的狀況，使用 SHA-1 當作位元亂數產生器，使用輸入的密碼當成 SHA-1 的種子產生一個亂數的位元序列，利用這個序列中的位元資料去決定每個霍夫曼表是否要嵌入資料，同時為了加強隱匿性，在產生的亂數序列中，當遇到的 0 是在這個序列中的第三個 0、第六個 0、第九個 0 這類屬於三的倍數的 0 時，略過這個 0 當作這個 0 不存在，不將他視為決定隱藏位置的的判斷

位元，直接在往下取出下一個位元取代這個跳過的位元值。

當得到的該位元為 1 時表示該處嵌入資訊，反之為 0 則不藏。另外還有一個例外的狀況就是當遇到的霍夫曼表的編號為 0 時，表示該處沒有使用霍夫曼表做編碼，則在這個地方也不嵌入任何資訊避免造成明顯的異常現象使人容易察覺。在這邊每個位元對應到一個 granule 中的一個霍夫曼表。在一個 granule 中可能包含三個或四個霍夫曼表，這樣的差異取決於該 granule 所對應的音訊訊號的取樣。此外，這些要嵌入的資訊將會先進行壓縮然後再做三重資料加密標準(Triple Data Encryption Standard, Triple DES)的加密，藉此加強在隱藏術的方法上的能力。

4.2.1 嵌入位元

以下針對 big_value 用霍夫曼表的編號嵌入秘密資訊的方法進行解說。基本的規則都相同，當嵌入的位元為 0 則霍夫曼表的編號為偶數；當嵌入的位元為 1，則霍夫曼表的編號為奇數。每一個表都個別唯一表示位元 0 或者位元 1 的資訊，以下分成嵌入位元 0、位元 1 及例外三種狀況做討論：

狀況 1－嵌入位元 0

當要嵌入的位元為 0 時表示需要使該被選擇的霍夫曼表的編號為偶數。如果該處被選擇的霍夫曼表的編號恰好為偶數，則不需要做任何的改變即可完成嵌入；如果為奇數則強制將其改變為偶數，且為了不使得強制改變後的霍夫曼表因為其所能表示的最大值小於要編碼的值，因此強制選擇的霍夫曼表編號為原先所選的編號的下一個。

狀況 2－嵌入位元 1

規則與嵌入位元 0 相同，改變的部份在於當要嵌入位元 1 時選擇的霍夫曼表的編號為奇數，當遇到需要強制改變霍夫曼表的編號的狀況時，其使用的原則與嵌入位元 0 相同。

狀況 3－例外狀況

在強制改變霍夫曼表的編號時，將會有五個表的改變方式有例外的狀況，分別為編號 3、13、23、30、31。

由於編號 4 及 14 的表並不被使用，所以當原先被選擇的霍夫曼表的編號為 3 時，需要強制改變時(嵌入位元為 0)，因為遇上的下一個表的編號為 4 不被使用，因此將編號 3 的表改變為編號 6；當被選擇的表編號為 13 時因為編號 14 的表一樣不被使用，所以改變為編號 16 的表。

當選擇的表為 23 或 31 時，由於這兩個表所能表示的最大值是這全部 32 個霍夫曼表中最大的兩個，所以當要強制改變時這兩個表只能彼此互換成為對方才不會造成編碼上的錯誤。

因此，當嵌入位元 0 時，如果被選擇的表為編號 23，則將其強制改變為表 31；當嵌入位元 1 時，如果被選擇的表為編號 31，則強制將其改為編號 23。簡單的來說將 23 及 31 視為一組，當被選到的表編號是 23 或者 31 時，如要表示位元 1 則選擇編號 23，要表示位元 0 則選擇編號 31。在這邊可以發現一個重要的例外，就是所有的表的編號中，編號 31 是唯一一個表示嵌入位元 0 的奇數。在其他的編號中，當編號為奇數則表示嵌入位元 1，當編號為偶數則表示嵌入位元 0。

最後，編號 30 的霍夫曼表，當其遇到嵌入位元 1 時，則將會強制改變編號為 23 的霍夫曼表，因為無法改變為原先預期的編號 31 的霍夫曼表，其所代表的為嵌入位元 0，因此改以編號 23 的霍夫曼表取代編號 31。

所有的霍夫曼表的編號對應其所嵌入的位元資訊顯示在 Tab. 3 及 Tab. 4 中，從表中的第二行及第五行可以發現，當嵌入位元 1 時，霍夫曼表的編號必定為奇數；從第三行及第六行可以發現，當嵌入位元 0 時，霍夫曼表的編號必定為偶數。這之中唯一的例外是霍夫曼表編號 31，編號 31 是唯一一個表示嵌入位元 0 的奇數編號霍夫曼表。因此可以很簡單的發現，除了編號 31 的霍夫曼表之外，只要表的編號為奇數則表示嵌入位元 1，為偶數則表示嵌入位元 0。

4.2.2 嵌入演算法

```
get first table number and random bit
while there are bits need to be embeded
  if table number != 0 and random bit is 1
    if Exception
```

```
        deal with Exception situation
      else if embed bit 1/0
        make sure table number is odd/even
      get next random bit
   get next table number
end while
```

4.2.3 取出位元

當在位元序列中該位元為 1 且依然有嵌入於其中的資料在 MP3 中，同時目前的這個霍夫曼表的編號不為 0 時，則檢查這個霍夫曼表的編號。除了編號 31 的霍夫曼表外，如果這個霍夫曼表的編號為奇數則取出的位元為 1，反之為偶數則取出的位元為 0。當取出的霍夫曼表編號為 31 時則取出的位元為 0。

在 Tab. 3 及 Tab. 4 中顯示霍夫曼表的編號所對應的嵌入位元。

4.2.4 取出演算法

```
get first table number and random bit
while there are bits need to be extracted
   if table number != 0 and random bit is 1
      if table number is 31
        extracted bit is 0
      else if table number is odd/even
        extracted bit is 1/0
      get next random bit
   get next table number
end while
```

5. 實驗

實驗所作的實做程式主要以參考 MP3Stego 為主，而實際上 MP3Stego 的程式則是根基於 8Hz-MP3 [2] 對 MP3 編碼及解碼的實做。

實驗的處理器為 Intel(R) Pentium(R) M processor 1.70GHz，編碼的實驗過程如 Fig. 7，而解碼的實驗過程如 Fig. 8。實驗數據則是取自三個 wav 格式的音訊檔，而嵌入的檔案皆為 14 個位元組的文字檔。其實驗音訊檔案相關資料紀錄在 Tab. 5，而其所對應的檔案實驗結果紀錄在 Tab. 6，從表中可以明顯發現在雙聲道的音訊資料上編碼及解碼的時間花費有顯著的差異。

在雙聲道的情況下編、解碼明顯緩慢許多，而在編碼上的差異又比解碼要來的明顯許多，主要造成如此差別的原因是因為雙聲道的音訊資料較為複雜，導致需要更多的處理時間。而在時間長度急遽增加的狀況下，編、解碼的時間長度也明顯的大幅度的提昇，顯示音訊檔案的時間長度的長短，對編、解碼的時間花費都有極大的影響力。

另外，將相同的音訊資料使用目前號稱最佳的 MP3 編碼器－LAME [14] 進行編碼時，可以發現與使用 8Hz-MP3 為基礎的編碼器在編、解碼時間上有著極大的落差，顯示兩個編碼器在本質上的效能上有著明顯的差異。

6. 結論

在研究 MP3Stego 的資訊隱藏方法中，我們發現了一個新的方式，利用改變壓縮編碼中所使用的霍夫曼表的編號來藏入隱藏資訊，而這樣的藏入方法所造成在人耳聽覺上的失真影響相當細微，以至於在實驗中經過資訊隱藏的檔案上，並沒有音訊資料會因為所作的這些改變而導致被人耳聽見其中有著不尋常的狀況。

而我們所提出的方法，相較於目前最廣為人知的 MP3Stego 而言，我們提高了資訊隱藏的效率，使得藏入的資訊平均而言是 MP3Stego 的二到三倍左右，並且同樣不容易被人耳聽覺所發現，而且我們實做的方法也比 MP3Stego 的使用性更加廣泛，將不再只局限於取樣頻率 44100 Hz、16 bit PCM 及單聲道的條件限制下，而能更全面性的對於各種音訊檔案加以進行資訊的隱藏，提昇了在 MP3 上進行資訊隱藏的實用性。

參考文獻

[1] Iso/iec international standard is 11172-3. Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s - Part 3: Audio, 1993.

[2] 8Hz. 8hz-mp3. http://www.8hz.com/mp3/, 2002.

[3] Anderson, R. J., and Petitcolas, F. A. P. On the

limits of steganography. *IEEE Journal of Selected Areas in Communications 16*, 4 (May 1998), 474–481. Special issue on copyright & privacy protection.

[4] Bailer, W. wbstego. http://wbstego.wbailer.com/, 2004.

[5] Brandenburg, K., and Popp, H. An introduction to mpeg layer, 2003.

[6] Chang, T. Y. Research and implementation of mp3 encoding algorithm. http://140.130.175.70/pdf2/90nctu0591009.pdf, 2002.

[7] Collomosse, J. Blindside cryptographic tool. http://www.mirrors.wiretapped.net/security/steganography/blindside/, 2001.

[8] D. Eastlake, r., and Jones, P. Us secure hash algorithm 1 (sha1), 2001.

[9] Fraunhofer-IIS. Audio and multimedia realtime systems mp3: Mpeg audio layer-3. http://www.iis.fraunhofer.de/amm/techinf/layer3/, 2006.

[10] Johnson, N. F., and Jajodia, S. Steganalysis of images created using current steganography software. *Lecture Notes in Computer Science 1525* (1998), 273–289.

[11] Katzenbeisser, S., and Petitcolas, F. A., Eds. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Inc., Norwood, MA, USA, 2000.

[12] Kwan, M. Gif colourmap steganography. http://www.darkside.com.au/gifshuffle/, 2003.

[13] Kwan, M. Snow. http://www.darkside.com.au/snow/, 2006.

[14] LAME. Lame ain't an mp3 encoder. http://lame.sourceforge.net/index.php, 2006.

[15] Latham, A. Jphide and jpseek. http://linux01.gwdg.de/alatham/stego.html, 1999.

[16] Mangarae, A. Steganography faq. http://www.zone-h.org/, 2006.

[17] Petitcolas, F. A. P. Mp3stego. http://www.petitcolas.net/fabien/steganography/mp3stego/, 2006.

[18] Vatolin, D., and Petrov, O. Msu stegovideo. http://compression.ru/video/stego ideo/index.html, 2006.

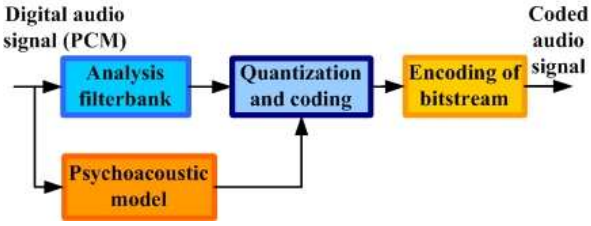

Fig. 1: Typical MPEG-1 Layer-3 Encoder Flowchart.

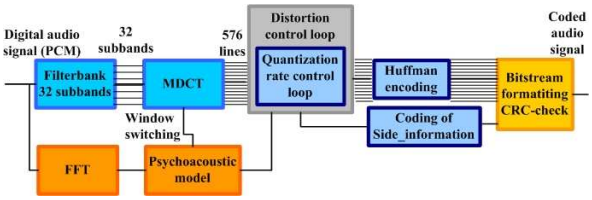

Fig. 2: An MPEG-1 Layer-3 Encoder.

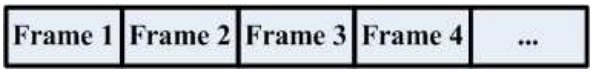

Fig. 3: Typical MPEG-1 Layer-3 Bitstream Structure.

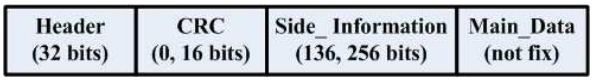

Fig. 4: Typical MPEG-1 Layer-3 Frame Data Structure.

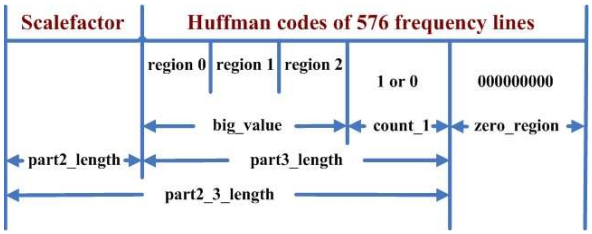

Fig. 5: Main_Data For A Granule.

Fig. 6: MPEG-1 Layer-3 Decoder Flowchart.

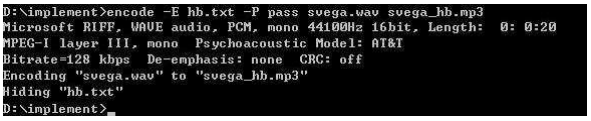

Fig. 7: Encoding example.

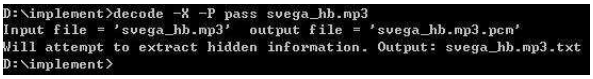

Fig. 8: Decoding example.

Tab. 1: 32 Huffman tables for big_value region [6]

Index	Maximum Value	Index	Maximum Value	Linbits
0	0	16	16	1
1	1	17	18	2
2	2	18	22	3
3	2	19	30	4
4	not used	20	78	6
5	3	21	270	8
6	3	22	1038	10
7	5	23	8206	13
8	5	24	30	4
9	5	25	40	5
10	7	26	78	6
11	7	27	142	7
12	7	28	270	8
13	15	29	526	9
14	not used	30	2062	11
15	15	31	8206	13

Tab. 2: 2 Huffman tables for count_1 region

Index	Maximum Value	Index	Maximum Value
32	15	33	15

Tab. 3: decode table for big_value

Table Index	Embed bit		Table Index	Embed bit	
	bit 1	bit 0		Bit 1	bit 0
0	-	-	16	**17**	16
1	1	**2**	17	17	**18**
2	**3**	2	18	**19**	18
3	3	**6**	19	19	**20**
4	-	-	20	**21**	20
5	5	**6**	21	21	**22**
6	**7**	6	22	**23**	22
7	7	**8**	23	23	**31**
8	**9**	8	24	**25**	24
9	9	**10**	25	25	**26**
10	**11**	10	26	**27**	26
11	11	**12**	27	27	**28**
12	**13**	12	28	**29**	28
13	13	**16**	29	29	**30**
14	-	-	30	**23**	30
15	15	**16**	31	**23**	31

*****粗體表示受到改變
*****底線表示例外狀況

Tab. 4: decode table for count_1

Table Index	Embed bit	
	bit 1	bit 0
32	**33**	32
33	33	**32**

*****粗體表示受到改變

Tab. 5: experiment audio files

File Index	Time Length (sec)	File Size (MB)	Stereo/ Mono	Data Type	Sample Rate (Hz)
1	20	1.73	Mono	wav	44100
2	24	4.18	Stereo	wav	44100
3	267	44.9	Stereo	wav	44100

Tab. 6: experiment result of audio files

Our Steganographic Method implementation				
File Index	STEGO		Non-STEGO	
	Encoding Time (sec)	Decoding Time (sec)	Encoding Time (sec)	Decoding Time (sec)
1	6	3	6	3
2	20	5	19	5
3	228	81	199	81
Other MP3 Encoder/Decoder				
File Index	MP3Stego		General MP3 Encoder/Decoder LAME 3.89	
	Encoding Time (sec)	Decoding Time (sec)	Encoding Time (sec)	Decoding Time (sec)
1	6	3	2	2
2	20	5	3	2
3	-	-	28	4