行政院國家科學委員會專題研究計畫　成果報告

數位典藏資訊之版權保護與驗證技術之研究(三)

計畫類別：個別型計畫
計畫編號：NSC93-2422-H-009-001-
執行期間：93 年 03 月 01 日至 94 年 02 月 28 日
執行單位：國立交通大學資訊科學學系(所)

計畫主持人：蔡文祥
共同主持人：吳大鈞
計畫參與人員：翁連奕、莊岳城、洪世結

報告類型：完整報告

處理方式：本計畫可公開查詢

中　華　民　國 94 年 5 月 31 日

行政院國家科學委員會補助專題研究計畫
期末執行報告

※※※※※※※※※※※※※※※※※※※※※※※※※※

※ 數位典藏資訊之版權保護與驗證技術之研究（三） ※

※ A Study on Copyright Protection & Authentication Techniques for Digital Information Archiving（3） ※

※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：個別型計畫

計畫編號：NSC92－2422－H－009－010

執行期間：93 年 3 月 1 日至 94 年 2 月 28 日

計畫主持人：臺中健康暨管理學院校長兼交通大學講座教授蔡文祥教授

共同主持人：高雄第一科技大學吳大鈞副教授

執行單位：國立交通大學資訊科學系

中 華 民 國 94 年 05 月 25 日

目錄

第一部份 期末執行成果略述

本計畫「數位典藏資訊之版權保護與驗證技術之研究」執行時間為 93 年 3 月 1 日至 94 年 2 月 28 日，須完成之工作項目包括：

1. 黑白影像之資訊隱藏技術與應用之開發
2. 公文影像之資訊隱藏技術與應用之開發
3. 影像認證中心工作的細化與運作

本計畫團隊經過一年積極的研究，已經完成上述全部工作。

此外，在去年 8 月 31 日，本計畫團隊與推廣辦公室合辦「數位典藏專業培訓課程系列四—數位典藏版權資訊研討會」，邀集數位典藏單位以及相關學術單位共二十多個單位參加。除了講解版權保護與驗證技術之發展歷程及現況之外，同時也發給參加人員「資訊保護家 3.0 版」應用軟體，讓各單位能在現場實際操作及發現問題。本計畫團隊也針對此次蒐集的問題詳加研究並予以改進。

而在去年 8 月 5 日、6 日由數位典藏技術分項所舉辦的第三屆數位典藏技術研討會中，本計畫發表了二篇論文:「Copyright Protection by Watermarking for Color Images against Rotation and Scaling Attacks Using Peak Detection and Synchronization in Discrete Fourier Transform Domain」與「Using a Human Visual Model and Boundary Lines for Embedding Robust Watermarks in Large Images」，其詳細內容如附件一。

此外依所研究的影像認證中心之運作，本計畫在二年前成立了交通大學影像認證中心，而在本年度內，本中心設計了註冊證明書（詳見附件二），作為典藏單位到認證中心來註冊的憑據。

由於目前本計畫正積極的發展「資訊保護家 4.0 版」軟體，希望能將這一年所發展的技術整合至軟體中，並在今年再次舉辦軟體訓練課程，讓本計畫團隊所發展的資訊保護技術能對數位典藏單位有實質上的幫助。

第二部分：計畫技術內容概述

在技術部分，本計畫所需完成之工作項目有黑白影像之資訊隱藏技術與應用之開發、公文影像之資訊隱藏技術與應用之開發，以及影像認證中心之工作的細化與運作等四項，在這個部分將針對這些技術作概略的說明。

一、 黑白影像之資訊隱藏技術與應用之開發

黑白影像的每個像素只能代表黑白兩色，故又稱二元（bi-level）影像。一般灰階及全彩影像也可以藉由臨界值法或網孔法（dithering）處理轉換成黑白影像。一般來說，二值化的目的是要將影像中的物體和背景分離，常應用在文字影像辨識上。浮水印相關的應用必須在影像中修改像素值才能達到植入資料的目的。黑白影像像素值代表兩極端對比顏色，若隨意改變其像素值，很容易觀察出影像的變化。所以在黑白影像中隱藏資訊的困難度很高。研究如何在不容易被人眼察覺的地方改變黑白像素值，進而達到植入資料的目的，是我們此一研究項目的目標。另外，為了植入更多的資料，就必須改變更多的黑白像素值，但是改變越多黑白像素值，影像的品質就會變的更糟、失真更多；相反的，如果要維持影像的品質，植入的資料就會變的較少。這是個兩難的問題，也是在計畫研究中要克服的地方，希望能做到植入的資料量既多，影像的失真亦少。

黑白影像之資訊隱藏技術與應用之開發主要進行下列兩項工作：

A. 以資料隱藏技術將註解性資料植入黑白影像中

在隱藏過程中，我們先將黑白影像切割成3×3區塊。每一區塊最多藏入兩個位元。接下來我們會判斷此區塊是否適合藏入資訊，若此區塊經判斷的結果不適合藏入資訊則不藏入任何的資料；若此區塊屬於可藏的區塊，則先判斷區塊中黑色像素個數，再視要隱藏的資料值為何，依據一先訂好的索引表格，調整區塊中黑色像素個數，以增加一個黑色像素、減少一個黑色像素或不改變，來對應其要隱藏的資料值。而在增加或減少黑色像素時，需要找不易被視覺察覺出來的地方才藏入，如黑色區塊或白色區塊的角點，以達到嵌入浮水印或註解資料而不被察覺之目的。對已隱藏資料之黑白影像，若要將所隱藏的資料抽取出來，我們先將黑白影像切成3×3二元區塊，依據先前植入資料的編碼規則，一一對每一3×3區塊進行處理，判斷該區塊中黑色像素的個數，便可分別得到其隱藏的資料。如此做到所有區塊都判斷完畢，抽取隱藏資料工作便告結束，而可得到完整的註解和典藏機構標記。

B. 以資料隱藏技術將可供防竄檢驗的易碎浮水印植入黑白影像中

一般來說，黑白影像中可隱藏的資料量不多，只能藏小型典藏機構標記或少量註解資料，而在驗證資料時也必須以較大區塊為判斷單位。本計畫也將依此性質提出對黑白影像植入認證用資訊的技術，利用這些認證用資訊，來驗證此黑白影像有無遭到竄改。

首先我們擬將將黑白影像切成不重疊的 9×9 區塊，再將各 9×9 區塊細分成九個 3×3 之二元區塊，接著從中先選出一個“可重新排列內容像素”的 3×3 區塊，再計算其餘 8 個區塊的標準差值，以此標準差值為相關的認證資料，重新排列可重排的區塊的像素位置，以不同的排列方式對應植入資料的認證資訊，來達到植入認證用資訊的目的。而所謂“選擇可

重新排列內容像素”的 3×3 區塊，是為了控制植入認證資訊後的影像品質。其方法是先用一個所謂“簡化半色調灰階函數”（reduced halftone gray function），給每一 3×3 區塊一個對應的灰階值，算出來的灰階值越大代表該 3×3 區塊黑色像素個數越少；基於此，我們將選擇黑色像素越多的區塊來做白色像素位置的重新排列，以及選擇白色像素越多的區塊來做黑色像素位置的重新排列，以便讓處理過的影像失真較少。當一張已植入認證資料的黑白影像要做影像驗證時，我們首先將黑白影像切成 9×9 區塊，再將此 9×9 區塊細分 3×3 二元區塊，之後計算 3×3 區塊的黑像素個數，再利用“簡化半色調灰階函數”算出對應的灰階值，選出“可重新排列內容像素”的一個 3×3 區塊，以其排列方式計算得到認證的資料，最後比對由其他 8 個 3×3 區塊計算標準差得到的認證資料，即可知道該 9×9 區塊有無遭到竄改：兩者認證資料相同代表未遭到竄改。

二、 公文影像之資訊隱藏技術與應用之開發

公文影像是指印刷或打字所成之文書稿件，再將其掃描或數位化後所得之影像。本計畫將探討以TIFF檔案方式存放之黑白公文影像之資料隱藏及影像認證技術。此外公文影像的最大特色是有大面積的白色區塊，因此，若在背景單調之區域隨意改變其像素值，則更容易觀察出影像的變化，故如何避開此種區塊而行藏入動作為其重點。不像一般的影像是針對圖片做處理，公文影像另一特色是針對字，只要隨意改變字的小小區塊的像素值，就會使原本清晰可見的字元產生模糊不清的現象，甚至可能會有錯誤情況發生。

公文影像之資訊隱藏技術與應用之開發主要進行下列兩項工作：

A. 以資料隱藏技術將註解性資料植入公文影像中

為了減少公文影像失真，我們提出一種稱為“環繞邊緣數”（surrounding edge count 簡寫為SEC）的測量，來估量在一影像區塊中結構上的亂度；並提出一所謂“像素可嵌性”(pixel embed ability)的測度，來選擇可藏資料的影像點，以減少影像的失真。除此之外，為了增加防止嵌入資訊被攻擊或不合法使用的安全性，我們用秘密金鑰(secret key)及亂數產生器(random number generator)來隨機化選擇藏入秘密資料的影像點位置。為了減少影像失真，我們將前述“像素可嵌性”定義如下。在隱藏過程中，先將原始影像切為3×3二元區塊，接著算出欲藏之點P由黑變白或由白變黑後其SEC值的變化$\triangle SEC_P$。若$\triangle SEC_P$小於某個門檻值(如3)，便可認定此點適合藏入資訊而不易被發覺。此外我們並將已處理過的點P及其周圍鄰居8點“標記”(label)起來；被標記的點就不能再被藏入資訊。一個像素若適合藏資訊及未被標記，我們便稱之為“可藏”(embeddable)，並稱該像素具有“可嵌性”。我們把典藏機構標記當作浮水印，並將其轉成二元串流(Binary stream)再選擇具可嵌性之像素，進行藏入動作。如上所提，為了增加嵌入資料的安全性，我們利用一把秘密金鑰和一個亂數產生器使所欲藏入的位置隨機化，即欲藏之資訊(0或1)應該被藏入那一個區塊的那一個位置點會隨機分布，進而達到安全嵌入浮水印或註解資料之目的。

B. 以資料隱藏技術將可供防竄檢驗的易碎浮水印植入公文影像中

本計畫針對公文影像認證提出解決方法。在黑白影像藏入驗證資訊亦會導致影像內容的破壞，所以好的解決方法不只在於減少被竄改機率的安全考量，也在於有效的減少因藏入驗證資訊而產生的影像失真。本計畫採納的驗證方式是要將隨機產生的驗

證碼藏於影像區塊中；竄改影像區塊將破壞此區塊的驗證碼而造成錯誤，並被驗證出來。講得更詳細點，在藏入驗證資訊方面，若輸入為一有L個區塊的原始影像I，我們將利用兩組金鑰K_1及K_2和兩組亂數產生器f_1及f_2來製作藏入驗證資訊後的影像I'。我們將先利用f_1和K_1產生一組L 亂數$c_1, c_2, \ldots, c_L$ 作為驗證碼，每一個亂數有m位元，再將c_i 藏入相對應的區塊B_i來產生I'。另一方面，在確認驗證資訊程序上，其輸入包括有L個區塊的影像I'、前述兩組金鑰K_1、K_2及兩組亂數產生器f_1、f_2，其輸出則是藏入驗證資訊後的影像I'的驗證報告。我們將利用f_1和K_1重新產生L個m位元驗證碼$c_1, c_2, \ldots, c_L$，再確認每個c_i在I'中相對應的區塊B_i。假若存在任一被竄改區塊，則記錄之並在最後輸出所有被竄改的區塊;否則即代表整張影像I'未被竄改。此處的難題一樣是如何藏入驗證碼而不被查覺到，而且如何與藏入的浮水印及其他資訊共存而不造成影像大幅失真。為此我們將利用公文的特性，考慮在公文影像上分區儲存不同資訊的方式。此法可行是因為公文中有很多區塊內涵為不重要、不具機密的標準欄位文字(如"姓名"或填表說明文字等)；該處本身並不須要驗證，而讓我們可用來藏入其他欄位的驗證資訊。

三、 影像認證中心工作的細化與運作

本計畫所發展的認證機制基本上為一由「主認證中心」及多個「子認證中心」所組成的雙層式架構，主認證中心即前述本計畫建立的「數位資訊認證中心」，各子認證中心也就是各典藏單位。由於各典藏單位都擁有「資訊保護家」應用軟體，就如同前面所述，各單位可藉由該軟體自行抽取出浮水印或註解資訊，以證明該影像、視訊檔案的版權。上述雙層式認證架構的運作方式為：各典藏機構所典藏之珍貴影像、視訊檔案可送至「交通大學數位資訊認證中心」註冊，除了在影像或視訊中藏入浮水印、註解等資訊之外，這個影像或視訊也會在本中心產生記錄。此種機制將對影像或視訊提供更公正的證明。此外，各典藏機構若對某一影像或視訊有侵權懷疑時，可將其攜至本中心，送入本中心浮水印讀取軟體，讀出是否有各典藏機構的各種版權資訊(包括各典藏機構的標記或識別碼，以及各種註解性資料)，再與事先在本中心註冊的資訊比對，即可判斷是否被侵權。此外，各典藏單位可能將典藏品授權給其他單位使用，若被授權的其中一個單位試圖在影像或視訊檔案中重新藏入該單位的標記，並宣稱該影像或視訊檔案為他們所有，那麼典藏單位將會遭受嚴重的損失，因此，除了黑白影像與公文影像的認證機制外，發展授權單位的認證機制將是本計畫的重點。

針對上述授權單位的認證機制，本計畫單位發展下列一套相關技術：

（1） 在各子認證中心將典藏品與軟體授權其他單位時，當被授權的單位要重新植入浮水印或註解時，軟體會將原本藏在影像中的識別碼抽出，在進行植入動作的同時，軟體會植入原本典藏單位與被授權單位的識別碼。

（2）往後若發生版權糾紛時，就可藉由本認證中心的軟體來抽取識別碼，從識別碼的判斷，就可知道該典藏品的原始擁有者及授權單位為何，因此可證明版權。

當然盜拷者仍有可能對本中心不服，但以本中心是國科會計畫所建立的機構，且為國立大學，相信必能獲得社會及法院之認可，於法律案件中扮演適當的證明角色。

第三部分 已開發技術詳述

Chapter1

Development of Data Hiding Technique and Application for Binary Images

1.1 Data Hiding Technique for Binary Images

In this chapter, the proposed method for embedding data in binary images is described. The idea is based on counting the number of the black pixels of a block to decide what kinds of combinations of bits can be utilized for data hiding.

The remainder of this chapter is organized as follows. In the first section, an introduction is given first. In the second section, the proposed data hiding method is presented. Some experimental results are shown in the third section. And finally, in the final section some discussions and a summary are made.

1.1.1 Introduction

Data hiding technique has been proposed for a variety of applications in digital images. Most works of data hiding in images were proposed for color or grayscale images because pixels in such images take a wide range of values and so are more proper for data hiding. One simple approach is to use the LSB replacement technique to hide data or authentication signals. However, data hiding in a binary image is a more challenging work. A reason is that changing a pixel in a binary image can often be detected visually because of the binary nature of the image.

1.1.1.1. Properties of Binary Images

In a binary image, there are only two pixel values, 0 and 255, and the corresponding pixels may be called *black* and *white* ones, respectively. If data are embedded in a binary image, the values of the image pixels will be altered. If the values of the image pixels are flipped arbitrarily, the resulting image will be quite noticeable. That is, it will cause visible artifacts in binary images to flip white or black pixels.

1.1.1.2. Problem Definitions

In order to embed more data in a binary image, more pixels need to be changed. The quality of the image will then get worse. On the contrary, in order to control the quality of the binary image, the number of hidden data should be smaller. The proposed method for data hiding in binary images is designed under the condition of making a compromise between the goal of embedding more data in the binary image and that of controlling the quality of the resulting image. Our method has the merit of concealing up to two bits of data in a 3×3 block by changing at most just one bit in a block. Another merit is that the hidden data can be extracted without referencing the original image.

1.1.2 Proposed Data Hiding Method

In this section, the method proposed to hide data in binary images and to extract the hidden data from stego-images is described. In order to embed up to two bits in a 3×3 block, a table about how to embed input data is constructed. And in order to control the quality of the resulting image, a principle about how to choose *flippable pixels* is proposed. They are both presented subsequently.

1.1.2.1 Data Embedding Process

In the proposed embedding data process, an input data stream D with L characters is converted in advance into a binary form, which we denote by $d_1d_2d_3\ldots\ldots d_{8\times L}$. On the other hand, an input binary image is divided into non-overlapping 3×3 blocks before being used for hiding data. Before describing the proposed hiding process, the definitions of some terms used later are given first as follows.

1. *Black contour*: a set of all black pixels whose next or previous pixel in the raster scanning order is a white pixel.
2. *White contour*: a set of all white pixels whose next or previous pixel in the raster scanning order is a black pixel.
3. *Starting pixel*: the pixel whose value is different from that of its previous pixel in the raster scanning order.
4. *Ending pixel*: the pixel whose value is different from that of its next pixel in the raster scanning order.
5. *Critical pixel*: a black or white pixel that belongs to one of the black or white contours and satisfies one of the following four conditions:
   I. The pixel is both a starting pixel from left to right and a starting pixel from top to bottom.
   II. The pixel is both a starting pixel from left to right and an ending pixel from top to bottom.
   III. The pixel is both an ending pixel from top to bottom and a starting pixel from left to right.
   IV. The pixel is both an ending pixel from top to bottom and an ending pixel from left to right.

Figure 1.1.1 shows an example illustrating some of these terms.

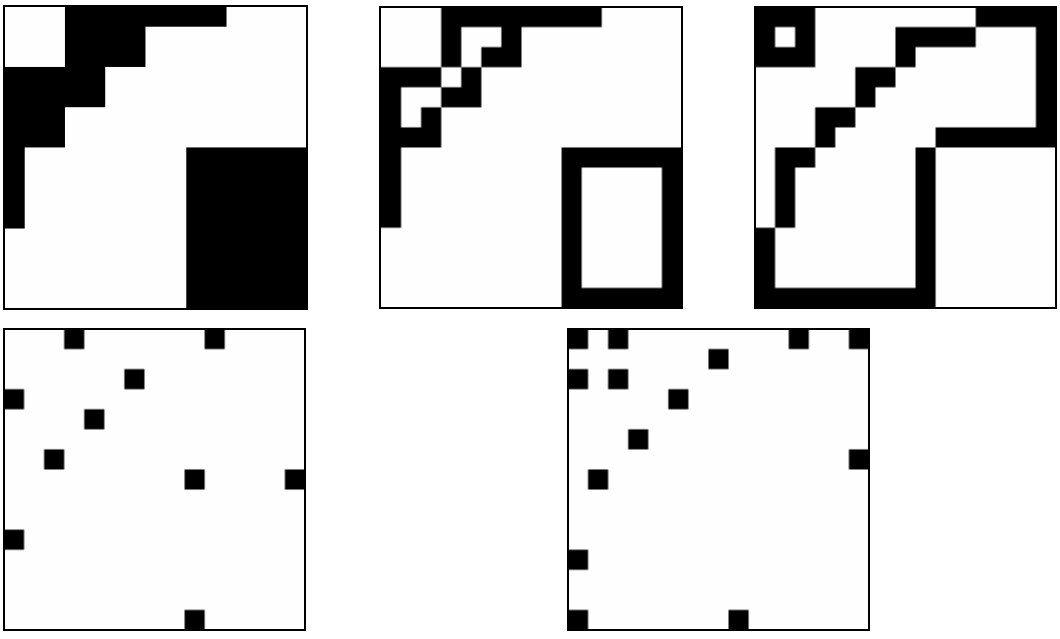


Figure 1.1.1 An example of terms. (a) A binary image. (b) A black contour of an image. (c) A white contour of an image. (d) Critical black pixels of an image. (e) Critical white pixels of an image.

The ideas involved in the proposed embedding process are described as follows. And a detailed algorithm for the process will be given later.

A. Finding Flippable Pixels

In order to keep the quality of binary images, it is important to choose *flippable pixels* cautiously. By a flappable pixel, we mean that the change of its binary value will not cause a noticeable artifact to a casual inspector. It seems to be a better choice to consider pixels on the region boundary. Therefore, for the binary image, all pixels on the black contour and white contour are taken as flippable pixels in this study, and so are critical pixels.

B. Creation of Embedding Table

In order to conceal the input data D in a binary image, every 3×3 block of the binary image is regarded as *a kind of combination of bits*. More specifically, by computing the number of black pixels in the 3×3 block, the block will be assigned a *bit-combination type*. The main idea is based on hiding at most two bits of data in a 3×3 block by changing at most one bit in the block. Note that most existing methods for data hiding in binary images can embed only one bit of data in a 3×3 image block.

In a 3×3 block, the possible number of black pixels is 0 through 9. "0" means the block is entirely white and "9" means the block is entirely black. For these two situations, to control the quality of the image, no bit should be hidden in the block; data bits can be hidden only in the other circumstances. An *embedding table* shown in Table 3.1 is designed in this study to accomplish the desired purpose of efficient data hiding. The ideas behind the design are described subsequently. Refer to the first column of the table about the various cases of data hiding in the following discussions.

a. Case A and Case J:

Because Case A means that the block is entirely black and Case J means that the block is entirely white, and so for either case, no bit is hidden in the block.

b. Case B:

Case B means that the block contains 8 black pixels. Under the aforementioned condition to hide at most two bits by changing at most one pixel, the possible number of black pixels of the block is 7 or 8 after hiding the input data. 7 black pixels mean one of the black pixels of the block should be flipped to

be white. And 8 black pixels mean the block is unchanged. Because the input data start with either “0” or “1”, we must handle both cases of the input data values.

(a). If the input data start with “0”, a block with 8 black pixels is used to represent bit “0.” That is, when a block has 8 black pixels and the input data start with “0”, then the bit “0” of the input data is regarded to be hidden in the block already without changing the block content.

(b). If the input data start with “1”, a block with 7 black pixels is used to represent bit “1.” That is, when a block has 8 black pixels, if the input data start with “1”, in order to hide the bit “1”, one of the 8 black pixels in the block will be flipped to be white so that the block becomes one with 7 black pixels.

Table 1.1.1 Proposed embedding table.

Case	Number(s) of black pixels before hiding	Represented bit(s)	Input data bit(s) to be embedded	Number(s) of black pixels after hiding
A	9	--	--	9
B	8	0	0	8
			1	7
C	7	1	0	8
			1	7
			01	6
D	6	01	1	7
			01	6
			00	5
E	5	00	01	6
			00	5
			1	4
F	4	1	00	5
			1	4
			01	3
G	3	01	1	4

			01	3
			0	2
H	2	0	01	3
			0	2
			1	1
I	1	1	0	2
			1	1
J	0	--	--	0

c. Case C:

Case C represents a block that has 7 black pixels. The possible number of black pixels of the block is 6, 7, or 8 after hiding the input data. For this, we consider the following cases.

(a). Because the block with 7 black pixels has already been used to represent the bit "1", if the input data start with "1", the block is kept unchanged.

(b). When the input data start with "0", we consider two situations as follows.

(c). If the next input data bit is "0", we take the input data bit as "0" and flip one of 2 originally white pixels to be black so that the block becomes one with 8 black pixels. The reason is that the block with 8 black pixels has already been used to represent the bit "0."

(d). If the next input data bit is "1", we take the input data bit as "01" (two bits together instead of just one) and change one of 7 black pixels be white so that the block becomes one with 6 black pixels. That is, the block with 6 black pixels is used to represent the bits "01."

d. Case D:

Case D represents that 6 black pixels are in a block. The possible number of black pixels of the block is 5, 6, or 7 after hiding the input data. We consider three situations as follows.

(a). When the input data start with two bits "01", the block will be kept unchanged with 6 black pixels.

(b). When the input data start with "1", by flipping one of 3 originally white pixels to be a black one, the block will become one with 7 black pixels.

(c). In order to handle all possible variations of the input data, the block with 5 black pixels is used to represent the bits "00." That is, when the input data start with "00", one of the 6

black pixels will be flipped to be a white one. The block will become one with 5 black pixels.

e. Case E:

Case E represents that 5 black pixels are in a block. The possible number of black pixels of the block is 4, 5, or 6 after hiding the input data. Three situations are considered as follows.

(a). Because a block with 5 black pixels is used to represent the bits "00", when the input data start with "00", the block will be kept unchanged

(b). If the input data start with "01", because a block with 6 black pixels is used to represent the bits "01", one of the white pixels in this block will be flipped to be a black one so that the block becomes one with 6 black pixels.

(c). In order to handle all possible variations of the input data, a block with 4 black pixels must be used to represent bits "1." That is, when the input data that start with "1", a block with 5 black pixels will be modified to become one with 4 black pixels. It is unreasonable to assign "10" to 4 black pixels because it will cause an input data starting with "11" to be out of control, or vice versa.

f. Case F:

For a block with 4 black pixels, consider the following situations after hiding input data.

(a). Because a block with 4 black pixels is used to represent the bit "1", if the input data start with "1", the block will be kept unchanged.

(b). If the input data start with "00", one of the white pixels in this block will be flipped to be black one so that the block becomes one with 5 black pixels.

(c). In order to handle all possible variations of the input data, a block with 3 black pixels is used to represent the bits "01." When the input data start with "01", one of the black pixels in this block will be flipped to be a white one so that the block becomes one with 3 black pixels.

g. Case G:

For a block with 3 black pixels, consider the following situations after hiding input data.

(a). If the input data start with "01", the block will be kept unchanged.

(b). If the input data start with "1", the block will be modified to become one with 4 black

pixels.

(c). Because a block with one black pixel may become one with 1 or 2 black pixels after hiding input data, a block with 2 black pixels is used to represent bit "0" and a block with 1 black pixel is used to represent bit "1." Therefore, if the input data start with "0", one of the black pixels in this block with 3 black pixels will be flipped to be a white one so that the block becomes one with 2 black pixels.

h. Case H and Case I:

The rest may be deduced in similar ways. A block with 2 black pixels in Case H will become one of the following situations after hiding input data.

(a). The block is kept unchanged if the input data start with "0."

(b). A black pixel in the block is flipped to be a white one so that the block becomes one with 3 block pixels if the input data start with "01."

(c). A white pixel in the block is flipped to be a black one so that the block becomes one with 1 block pixel if the input data start with "1."

And in Case I, for a block with one black pixel, if the input data start with "1", the block will be kept unchanged. If the input data start with "0", the block will be changed to become one with 2 block pixels.

C. Ways for flipping pixels

In order to satisfy the number of black pixels of a 3×3 block according to the embedding table, a black pixel of the 3×3 block may be flipped to white or a white pixel of the 3×3 block may be flipped to black. The way proposed in this study to flip pixels is as follows.

a. If the number of black pixels needs to be decreased, it means that a black pixel has to be flipped to white in this block. The way proposed in this study for this is as follows:

(1) if critical black pixels exist in this block, one of the critical black pixels is selected and flipped to white;

(2) if critical black pixels do not exist in this block, one of black pixels in the black contour is selected and flipped to white.

b. If the number of black pixels needs to be increased, it means a white pixel has to be flipped to black in this block. The way proposed in this study for this is as follows:

(1) if critical white pixels exist in this block, one of the critical white pixels is selected and flipped to white;

(2) if critical white pixels do not exist in this block, one of the white pixels in the white contour is selected and flipped to black.

A reason for assigning higher priority to critical pixels than to contour pixels for use as flippable pixels is that critical pixels are located at the corners of contours and flipping of them will be less perceptive for the human visual system than flipping of contour pixels.

D. Detailed Algorithm

The inputs to the proposed data embedding process include a binary image C and certain secret data D with L characters. The output is a stego-image S. The process can be briefly expressed as an algorithm as follows. Figure 1.1.2 illustrates a flowchart of the embedding process.

Step .1 Find all the black contour, the white contour, and the critical pixels of C.

Step .2 Convert D into a binary form ($d_1d_2d_3\ldots\ldots d_{8\times L}$) and divide C into non-overlapping 3×3 blocks.

Step .3 For each 3×3 image block, perform the following operations.

3.1 Count the number of the black pixels in the block.

3.2 Select flappable pixels and flip them according to the embedding table and input data in a way as described previously.

Step .4 Take the final result as the desired stego-image S.

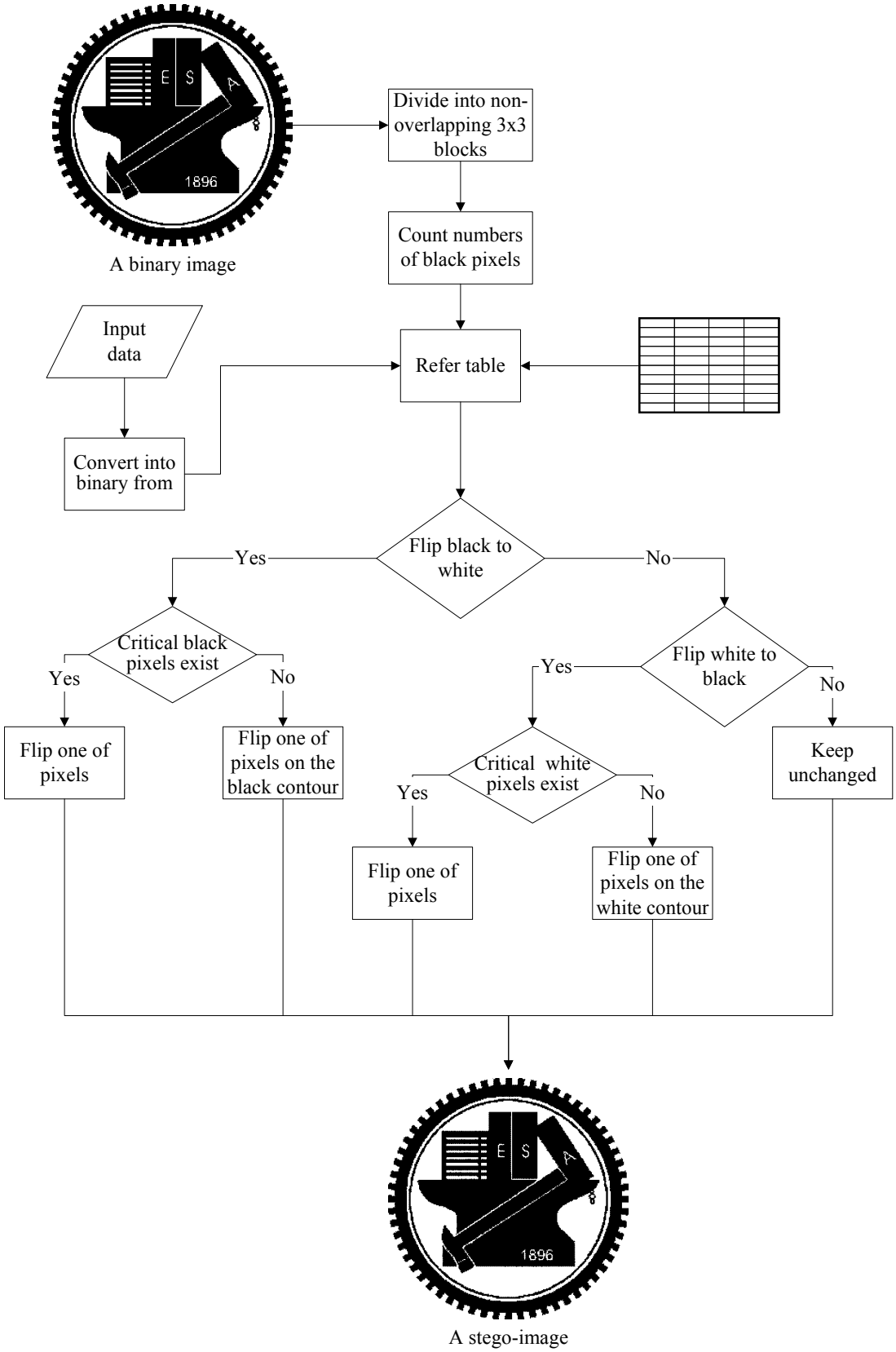


Figure 1.1.2 Flowchart of the proposed embedding process.

1.1.2.2 Data Extraction Process

In the proposed data extraction process, Table 1.1.1 is first simplified as an extraction table as shown in Table 1.1.2. It is easy to finish the extraction process. The stego-image is first divided into non-overlapping 3×3 blocks. For each 3×3 block, the number of black pixels in it is computed. And by table lookup, the embedded data bit(s) in the block can be determined. After extracting all embedded data bits, they are converted to obtain the original data.

Table 1.1.2 Extraction table.

Number(s) of the black pixels	Hidden bit(s)	Number(s) of the black pixels	Hidden bit(s)
9	--	4	1
8	0	3	01
7	1	2	0
6	01	1	1
5	00	0	--

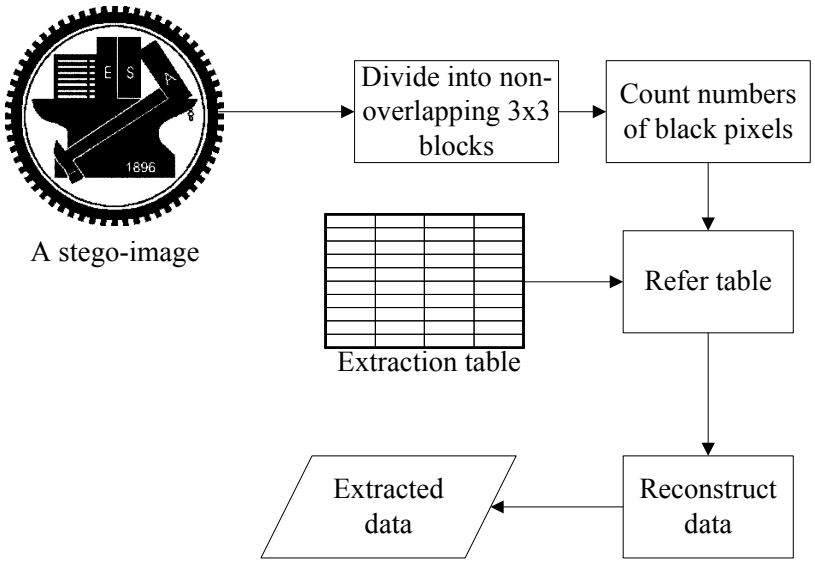


Figure 1.1.3 Flowchart of the proposed extraction process.

1.1.3 Experimental Results

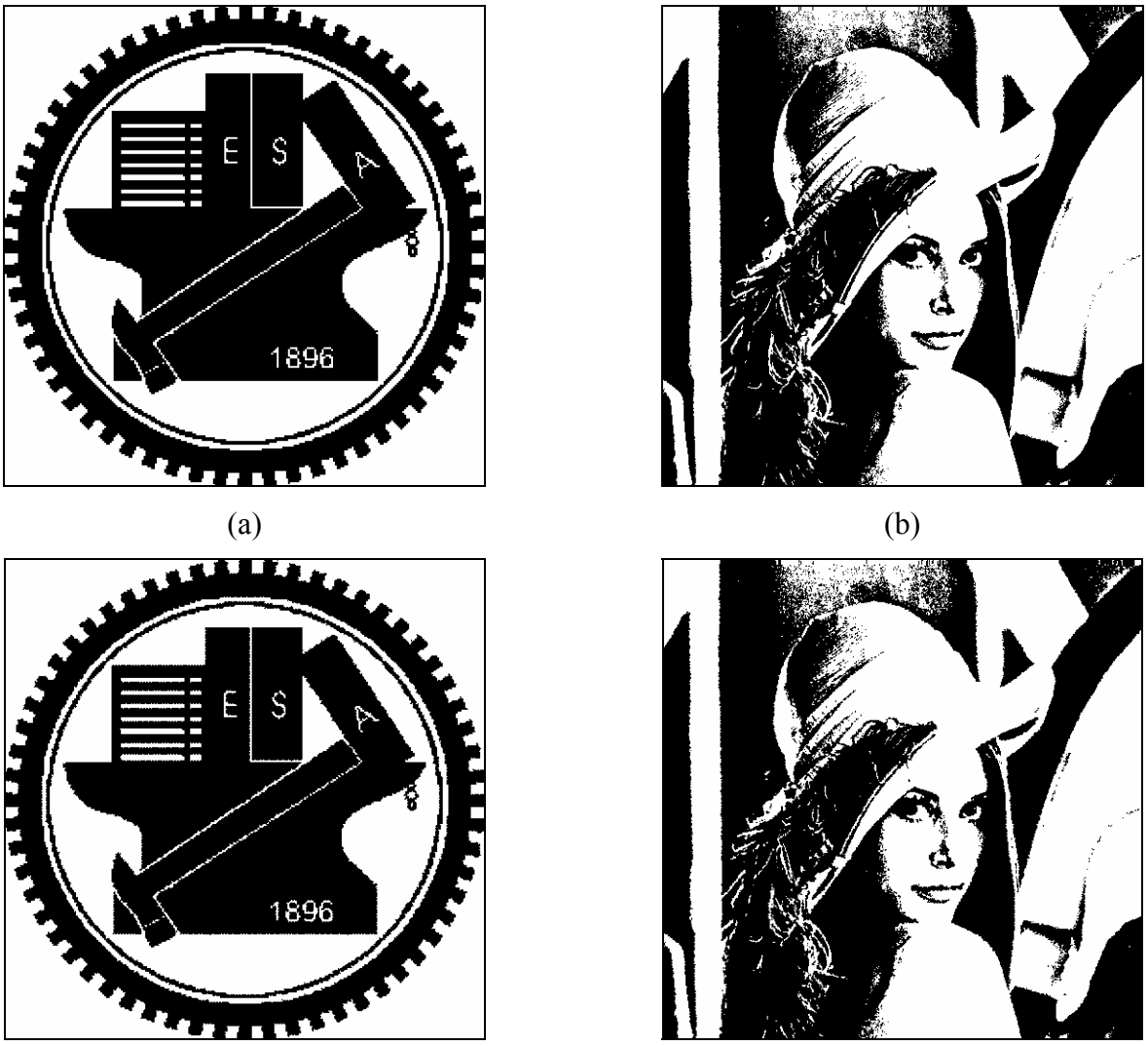
Some experimental results of applying the proposed method are shown here. The secret data α are designed to be *K* times duplications of the string "1100," i.e., α = 110011001100…1100. Figures 1.1.4(a)

and (b) show two binary images both with size 512×512. And the stego-images after embedding the secret data are shown in Figures 1.1.4(c) and (d), respectively. And Figures 1.1.4(e) and (f) show their differences in black pixels after embedding the secret data, respectively.

Figures 1.1.5(a) and (b) show two binary document images both with size 512×512. And the stego-images after embedding the secret data are shown in Figures 1.1.5(c) and (d), respectively. And Figures 1.1.5 (e) and (f) show their differences in black pixels after embedding the secret data, respectively. And Table 1.1.3 shows the numbers of the used blocks, the amount of the embedded bits, and the numbers of the difference pixels for the four binary images. An average amount of embedded bits in a block is 1.25~1.30 bits.

1.1.4 Discussions and Summary

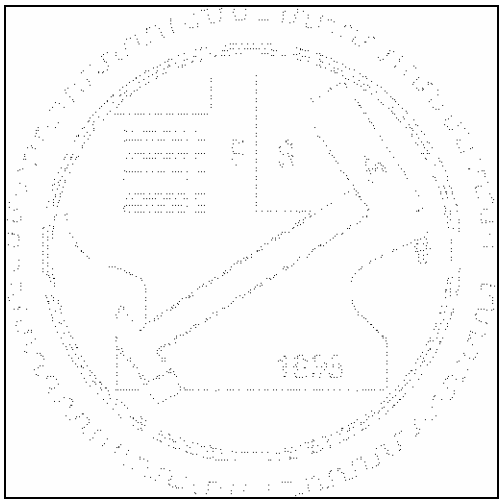
In this chapter, we propose a novel data hiding technique for binary images. An embedding table is created in our proposed method. With the help of the table, we can understand what kinds of combinations of bits should be embedded in a block. Our method can embed up to two bits in a 3×3 image block, by changing at most just one bit in a block. Besides, the image quality will be also considered in our method. In order to keep the image quality, each pixel that needs to be flipped must be located in the image boundary. The reason is that it is imperceptible for the human visual system to flip pixels in the image boundary to be black or white one. Therefore, by our method, not only more data can be embedded in a binary image, but also the quality of the stego-image will be not bad.



(a) (b)

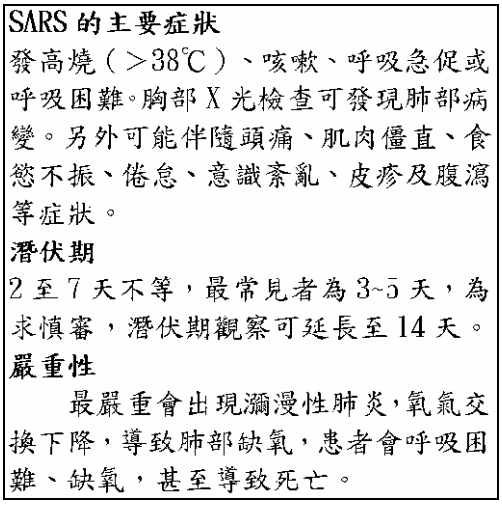
(c) (d)

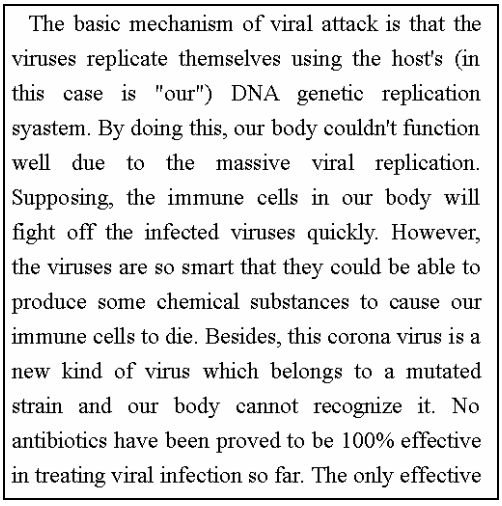
Figure 1.1.4 Input binary images, output stego-images with secret data, and the differences. (a) Binary image "NCTU". (b) Binary image "Lena". (c) and (d) Stego-images after embedding secret data, respectively. (e) and (f) The difference pixels after embedding secret data, respectively.



(e) (f)

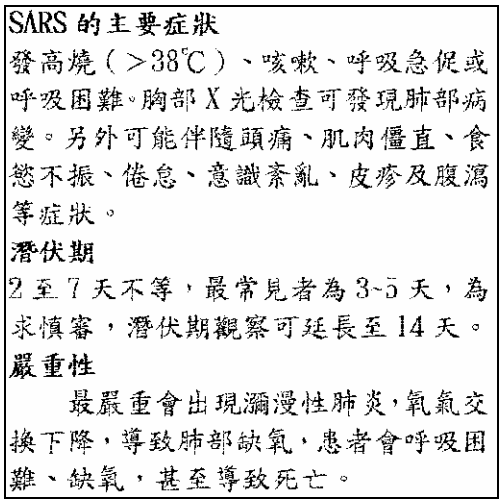
Figure 1.1.4 Input binary images, output stego-images with secret data, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding secret data, respectively. (e) and (f) The difference pixels after embedding secret data, respectively (continued).



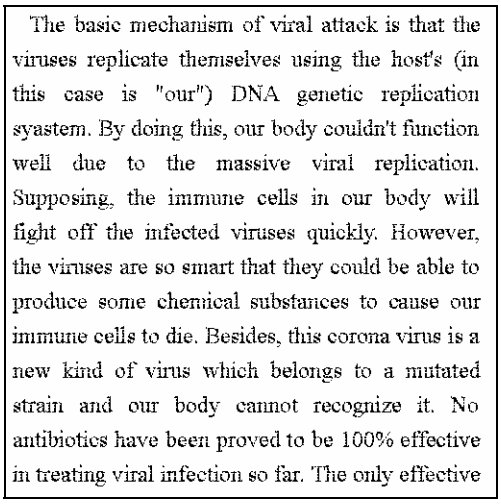


(a) (b)

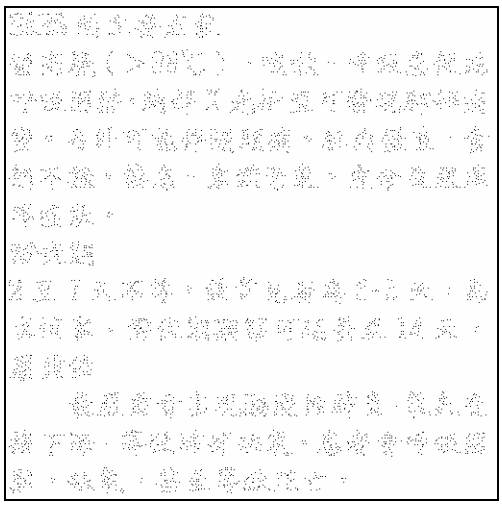
Figure 1.1.5 Input binary document images, output stego-images with secret data, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding secret data, respectively. (e) and (f) The difference pixels after embedding secret data, respectively.



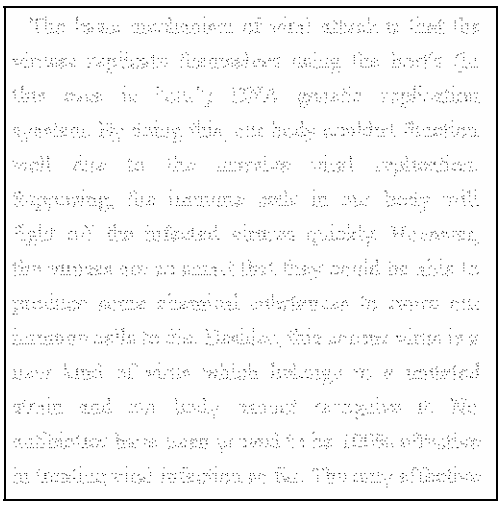
(c)



(d)



(e)



(f)

Figure 1.1.5 Input binary document images, output stego-images with secret data, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding secret data, respectively. (e) and (f) The difference pixels after embedding secret data, respectively (continued).

Table 1.1.3 The statistics about the numbers of used blocks, embedded bits, and difference pixels for stego-images of Figs. 1.1.4 and 1.1.5 after embedding the secret data.

	NCTU	Lena	Chinese	English
Used blocks	3243	4489	6755	6472
Embedded bits	4181	5572	8842	8364
Difference pixels	2129	2724	4133	4453

1.2 A New Image Authentication Technique for Binary Images

In this chapter, the proposed new method for binary image authentication is presented. By rearranging all the pixels of each block in a binary image, authentication signals represented by the pixels' locations can be embedded in the image. Image authentication can be achieved by checking the rearranged locations of all the pixels in each block of a given image in suspicion.

The remainder of this section is organized as follows. In Section 1.2.1, an introduction is given first. In Section 1.2.2, the proposed authentication method is described. In Section 1.2.3, some experimental results are given to show the feasibility of the proposed approach. Finally, in Section 1.2.4 some discussions and a summary are made.

1.2.1 Introduction

Because image transmission is a major activity in today's communication and digital images can be modified easily, it is necessary to design an effective algorithm for image authentication. However, in a binary image there are only two types of pixels, black and white, with values 0 and 255, respectively, and so if the values of the image pixels are flipped arbitrarily, visible artifacts in the image will be created. Therefore, authentication of binary images is a more challenging work than that of other types of images.

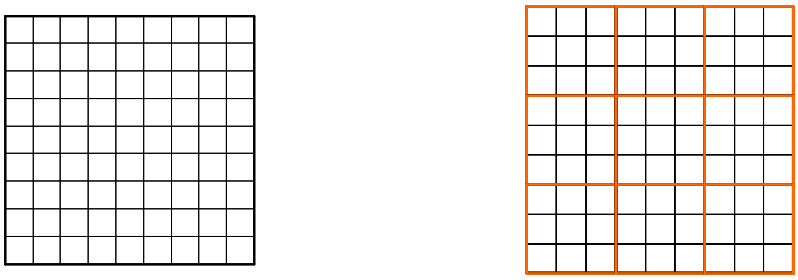
In order to verify the fidelity of a binary image, authentication signals need to be embedded in the image. It is hoped that such signals may also be used to check the integrity of each image block. The proposed method meets these two purposes. It takes all the pixels of a properly-selected 3×3 block in each 9×9 block to compute an authentication signal. Such authentication signals can be used to conduct the authentication work for a given image without using other information.

1.2.2 Proposed Authentication Method

In this section, the proposed method to embed authentication signals into a binary image and to authenticate the resulting image is introduced. The idea of *inverse halftoning* is employed in our method. The halftone technique was proposed to convert grayscale images into binary ones and the inverse halftoning process aims to recover grayscale images from binary halftone images. In the *modified* inverse halftoning technique proposed in this study for use in authentication signal generation, each 3×3 block of a given binary image is assigned a gray value.

1.2.2.1 Authentication Signal Generation and Embedding

To generate authentication signals for a binary image, the image is first divided into non-overlapping 9×9 blocks. Then, each 9×9 block is divided further into nine non-overlapping 3×3 blocks. Figure 1.2.1 shows an example of a 9×9 block and its nine 3×3 blocks.



(a) (b)

Figure 1.2.1 An example of 9×9 image blocks. (a) A 9×9 block. (b) Each 3×3 block in the 9×9 block.

A. Assigning Gray Values to 3×3 Blocks

In the proposed modified inverse halftoning technique, each 3×3 binary image block B is assigned a gray value G by the following the *reduced halftone gray function* proposed in this study:

$$G = \lfloor (9 - N) \times 255 / 9 \rfloor, \qquad (1.2.1)$$

where N is the number of black pixels in B, and $\lfloor \bullet \rfloor$ means the integer floor function. The reduced halftone gray function maps the range of gray values [0 255] into 9 discrete gray levels. That is, for an input N, B is assigned a gray value G which is one of the nine values 0, 28, 57, …, 255. For example, if N is 0, the assigned gray value is 255. If N is 6, then the assigned value is 85. And if N is 9, the assigned value is 0. Therefore, in each 9×9 block with nine 3×3 blocks, nine gray values will be assigned. Such

assigned gray values will be called *reduced halftone gray values*, and abbreviated as *RHG values*, where the word *reduced* is used to indicate that only nine discrete gray values instead of the usual 256 ones are generated here from the gray scale.

B. Choice of Rearrangeable Block

In order to control the quality of the image resulting from authentication signal embedding, we should select 3×3 image blocks carefully to embed the signals. Each block selected for this purpose is called a *rearrangeable block* in this study. The reason for using this term will be obvious later. For each 9×9 block, if it is neither entirely black nor entirely white, two candidate 3×3 blocks for signal embedding are picked out within it. One block is that with its RHG value G_s being the *smallest* but not 0, and the other that with its RHG value G_l being the *largest* but not 255. The reason to select them is explained subsequently. First, a larger RHG value means that the black pixels in the block are fewer, and a smaller RHG value means the reverse. Taking the latter case as an example, it means that the white pixels in the block are fewer. So it will cause less distortion to rearrange the positions of these white pixels than to rearrange those of the black ones. Similar reasoning applies to the former case. The desired rearrangeable block is chosen from the two candidate blocks in this study. Let w_s be the number of white pixels in the block B_s whose RHG value is G_s and b_l be the number of black pixels in the block B_l whose RHG value is G_l. If w_s is larger than or equal to b_l, then according to the previous discussion B_s is taken as the rearrangeable block in the proposed method; otherwise, B_l is taken as the rearrangeable block. We call this way of selecting a rearrangeable block in a 9×9 block *a rearrangeable block selection process* in the sequel.

C. Calculation of Standard Deviation

For each 9×9 block, a standard deviation σ of the RHG values of the eight 3×3 blocks other than the rearrangeable block is calculated. And a *standard deviation level L* is computed according to the following rule:

$$L = n, \qquad \text{if } (n \times 128 / 9) \geq \sigma \geq [(n - 1) \times 128 / 9]. \qquad (1.2.2)$$

Because in the gray value range [0 255], the largest integer value of the standard deviation σ is 128, the possible value of σ will fall within the range $R = [0, 127]$. Therefore, we normalize R into 9 levels in designing the rule of (1.2.2) above. For example, when $\sigma = 0$, then $L = n = 1$; when $\sigma = 127$, $L = n = 9$;

and when $\sigma = 80$, $L = n = 6$.

D. Rearrangement of Pixels

For the rearrangeable 3×3 block B in each 9×9 block, let N be the number of black pixels in this block. We consider two cases in the following.

(a). Case 1:

If $N \le 4$, it means the number of black pixels in B is fewer than the number of white ones. So, it is faster and also causes less distortion to rearrange the locations of the N black pixels in B than to do so for the $(9 - N)$ white ones. Therefore, we will assign N new locations to the N black pixels, respectively. We employ the value N and the standard deviation level L to produce a rearrangement rule for this purpose as follows:

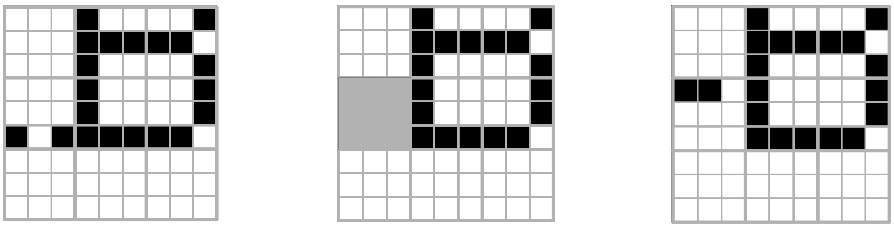
$$P_i^b = \left(L \times N^i \bmod C\right) \bmod 9 \qquad \text{for all } i \le N \text{ and } N \le 4, \qquad (1.2.3)$$

where P_i^b denotes the index of the new location of the ith black pixel in B and C is a constant pre-selected in such a way that each value of P_i^b is distinct. The indexes of the original locations of a 3×3 block is shown in Figure 1.2.2. The value of C is chosen to be 11 according to our experimental experience in this study. As a summary, the essence of the proposed binary image authentication method is that all the indexes specified by P_i^b are regarded as *authentication signals*.

0	1	2
3	4	5
6	7	8

Figure 1.2.2 Indices of a 3×3 block.

For example, Figure 1.2.3(a) is a 9×9 block and Figure 1.2.3(b) shows the rearrangeable 3×3 block, which is the one in shadow, of a 9×9 block. According to the aforementioned steps, N is 2 and L is 5. By (1.2.3), we can compute P_1^b to be 1 and P_2^b 0. That is, the two pixel positions with indexes, 0 and 1, are filled with black pixels according to the proposed rearrangement rule. The other positions will be filled with white pixels. Figure 1.2.3(c) is the resulting 9×9 block after embedding the authentication signals in this way.



(a) (b) (c)

Figure 1.2.3 An example. (a) A 9×9 block. (b) The rearrangeable 3×3 block (in shadow) of the 9×9 block. (c) The 9×9 block resulting from embedding authentication signals.

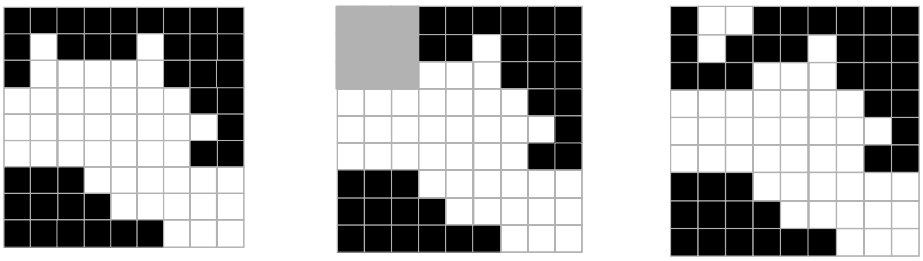
(b). Case 2:

If $N \geq 5$, it means that the number of black pixels in B is larger than the number of white ones, which is $(9 - N)$. So, it is faster and causes less distortion to rearrange the locations of the $(9 - N)$ white pixels in B than to rearrange those of the N black ones. That is, we will assign $(9 - N)$ new locations to the $(9 - N)$ white pixels, respectively. We again employ the values of N and L to produce another rearrangement rule as follows:

$$P_i^w = \left(L \times N^i \bmod C\right) \bmod 9 \quad \text{for all } i \leq (9 - N) \text{ and } N \geq 5, \qquad (1.2.4)$$

where P_i^w denotes the index of the new location of the ith white pixel in B and C is a constant pre-selected to be 11 in a way as mentioned previously. Similarly all P_i^w are regarded as authentication signals in this study.

For example, Figure 1.2.4(a) is a 9×9 block and Figure 1.2.4(b) shows its rearrangeable 3×3 block in shadow. According to the aforementioned steps, N is 6 and L is 8. By (1.2.4), we can compute P_1^w, P_2^w, and P_3^w to be 2, 2, and 1, respectively. That is, the three pixels at positions, 1, 2, and 4, are filled with white pixels according to the rearrangement rule. The other positions are filled with black pixels. Figure 1.2.4(c) shows the resulting 9×9 block after embedding the authentication signals in this way.



(a) (b) (c)

Figure 1.2.4 Another example. (a) A 9×9 block. (b) The rearrangeable 3×3 block (in shadow) of the 9×9 block. (c) The 9×9 block after embedding authentication signals.

Based on the idea of the inverse halftoning, it is noted the RHG value of the rearrangeable 3×3 block, after the above pixel rearrangement, will still be the same as before because the change is just a permutation of the pixels' positions and not the number of the black pixels or white pixels.

E. Detailed Algorithm

The input to the proposed authentication signal embedding process is a binary image *I*. The output is a stego-image *S*. The algorithm for the process is as follows. Figure 1.2.5 shows a flowchart for the algorithm.

Step .1 Divide *I* into non-overlapping 9×9 blocks.

Step .2 Divide each 9×9 image block further into non-overlapping 3×3 blocks.

Step .3 For each 3×3 image block, count the number *N* of the black pixels in it and assign it an RHG value G_i by the reduced halftone gray function described by (1.2.1).

Step .4 For each 9×9 image block *D*, perform the following operations.

4.1. Find the rearrangeable block *B* in *D* using the RNG values of all the blocks in *D* according to the aforementioned rearrangeable block selection process.

4.2. For each of the remaining eight 3×3 blocks in *D*, calculate its standard deviation σ and get the standard deviation level *L* according to (1.2.2).

4.3. For the 3×3 rearrangeable block *B*, rearrange all pixels of *B* according to (1.2.3) and (1.2.4).

Step .5 Take the final result as the desired stego-image *S*.

1.2.2.2 Image Authentication Process

In the authentication signal embedding process, the locations of all pixels of the rearrangeable 3×3 block are taken as authentication signals. So, we can judge an image in suspicion as being tampered with or not by checking the locations of all pixels of the 3×3 rearrangeable block according to the rearrangement rule described previously.

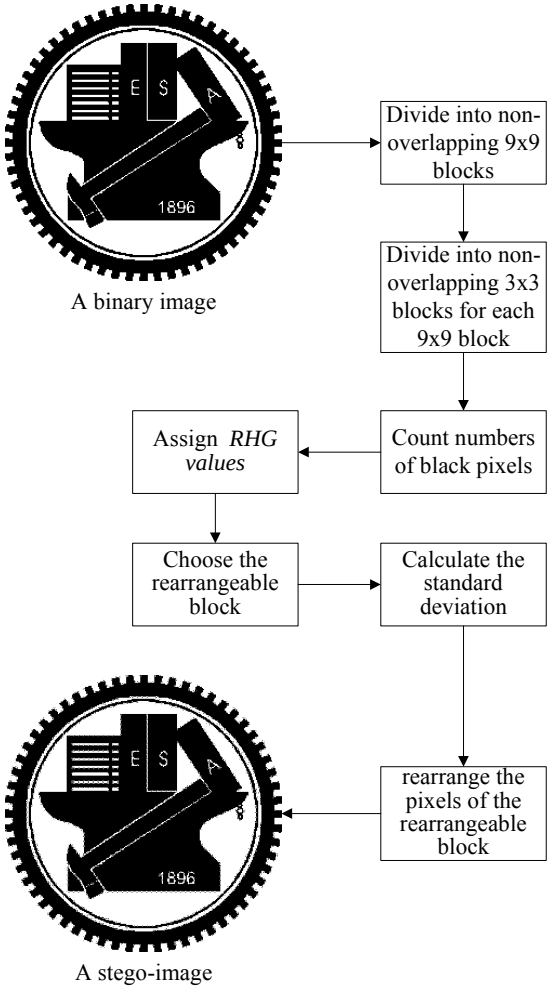


Figure 1.2.5 Flowchart of authentication signal embedding process.

The proposed image authentication process is essentially similar to the proposed authentication signal embedding process but in a reverse order. A suspicious image is first divided into non-overlapping 9×9 blocks. Then, each 9×9 block is divided further into nine non-overlapping 3×3 blocks. For each 3×3 image block *B*, the number *N* of black pixels in it is counted. And *B* is assigned an RHG value *G* by the reduced halftone gray function. And then, for each 9×9 block, the rearrangeable

3×3 block in it is selected. The standard deviation σ and the level L of σ of each of the remaining eight 3×3 blocks are then computed. By the rearrangement rules (1.2.3) and (1.2.4) with N and L as inputs, authentication signals P_i^b or P_i^w can be obtained. By checking the permutation of all the pixels of the rearrangeable block, if the computed indexes P_i^b (or P_i^w) are the same as the locations of all the black (or white) pixels of the rearrangeable block, the 9×9 block is judged as not being altered; otherwise, tampered with. In the output images of our experiments, blocks judged as being tampered with are marked with red color.

A. Detailed Algorithm

The proposed image authentication algorithm can be expressed as an algorithm as follows. The input is a stego-image S. The output is an *authentication image A*. Figure 1.2.6 illustrates the process.

Step .1 Divide S into non-overlapping 9×9 blocks.

Step .2 For each 9×9 image block, divide it further into non-overlapping 3×3 blocks.

Step .3 For each 3×3 image block, counter the number N_i of black pixels in it and assign it an RHG value G_i by reduced halftone gray function specified by (1.2.1).

Step .4 For each 9×9 image block D, perform the following operations.

4.1 Find the 3×3 rearrangeable block B of D according to the aforementioned rearrangeable block selection process.

4.2 For each of the remaining eight 3×3 blocks in D, calculate its standard deviation σ and get the corresponding standard deviation level L according to (1.2.2).

4.3 For the 3×3 rearrangeable block B with N black pixels, perform the following operations.

4.3.a If $N \le 4$, calculate authentication signals P_i^b by (1.2.3) for i =1, 2, …, N. Let the index of ith black pixel be denoted as p_i^b for i =1, 2, …, N. For all i, if $P_i^b \neq p_i^b$, regard the 9×9 image block as being tampered with and mark it red.

4.3.b If $N \ge 5$, calculate authentication signals P_i^w by (1.2.4) for i =1, 2, …, (9 – N). Let the index of ith white pixel be denoted as p_i^w for i =1, 2, …, (9 – N). For all i, if $P_i^w \neq p_i^w$, regard the 9×9 image block as being tampered with and mark it red.

4.4 Take the final result as the desired stego-image *S*.

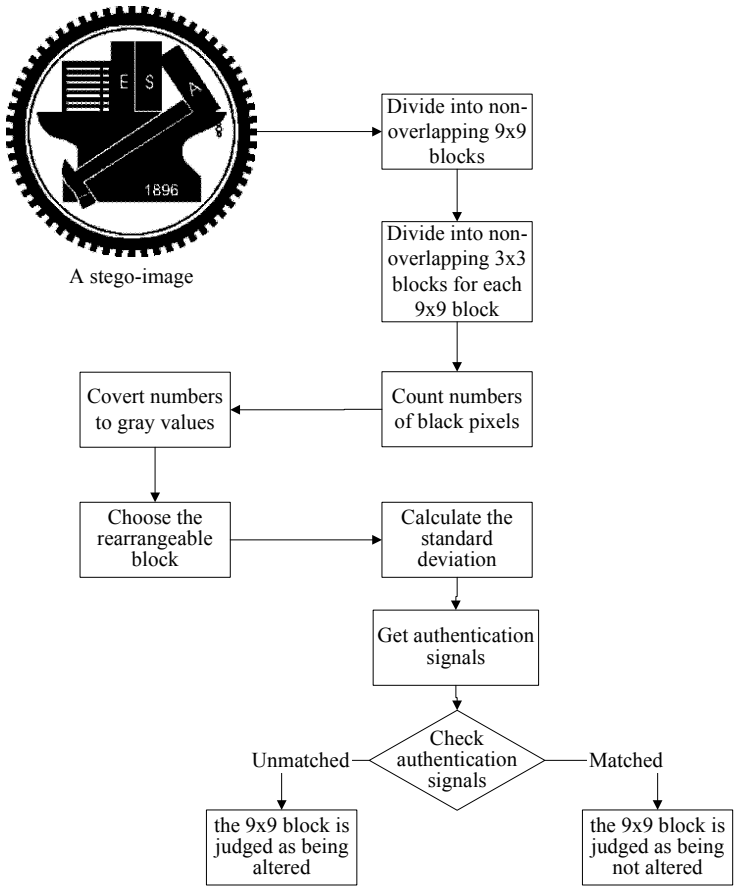


Figure 1.2.6 Flowchart of proposed image authentication process.

1.2.2.3 Applications

For binary images, we can apply the proposed image authentication method to check whether a binary image is tampered with or not. But this is just one application of the proposed method described previously. Another application of the method is described in this section.

In Section 1.1, we have introduced a data hiding method for binary images. In this data hiding method, we take the number of black pixels in each 3×3 binary image block as a kind of combination of bits. Because the proposed method for embedding authentication signals described in the last section changes just the permutation of the pixels' positions and not the number of the black pixels or white pixels, this image authentication method can be applied in the proposed data hiding method to

authenticate the hidden data. This application will not destroy the hidden data. That is, after implementing image authentication method, we can extract correctly the hidden data.

More specifically, if a person, say A, wants to send some secret data to another person, say B. First, A can implement the data hiding method proposed previously to embed the secret data in a binary image, yielding a stego-image *T*. Next, A can apply the image authentication method proposed in this chapter to embed authentication signals in *T* and produce another stego-image *F*. Finally, A sends *F* to B.

Before extracting data from *F*, B can employ the image authentication process to verify the integrity of *F*. If the result of image authentication says that the image *F* is not tampered with, then B can proceed to extract the hidden data correctly. Otherwise, B, knowing that the received image *F* has been altered, can abandon it and ask A to send the secret data hidden in a similar way again. Figure 1.2.7 shows a flowchart of this application.

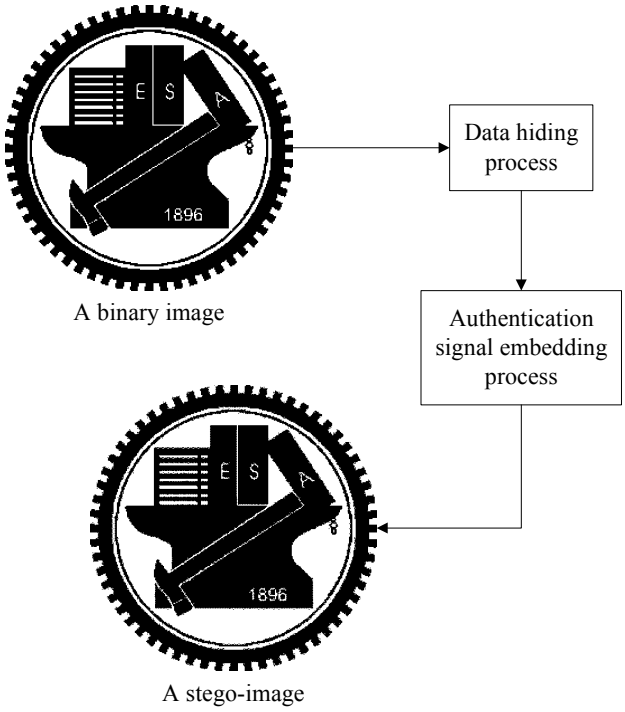


Figure 1.2.7 Flowchart of application.

1.2.3 Experimental Results

Some experimental results of applying the proposed method are shown here. Figures 1.2.8 (a) and (b) show two binary images both with size 512×512. And the stego-images resulting from embedding the authentication signals are shown in Figures 1.2.8 (c) and (d), respectively. And Figures 1.2.8 (e) and

(f) show their differences in black pixels after embedding authentication signals, respectively.

Figures 1.2.9 (a) and (b) show two binary document images both with size 512×512. And the stego-images resulting from embedding the authentication signals are shown in Figures 1.2.9 (c) and (d), respectively. And Figures 1.2.9 (e) and (f) show their differences in black pixels after embedding authentication signals, respectively.

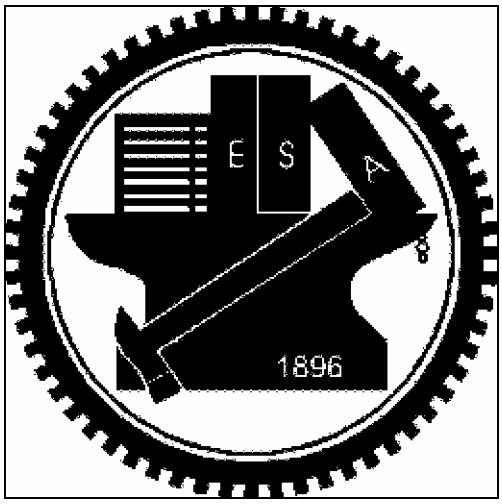
Four tampered and cropped images are shown in Figures 1.2.10 (a) through (d). And Figures 1.2.10 (e), (f), (g), and (h) show the respective authentication results. The red parts indicated the detected tampered areas.

Figures 1.2.11 (a) and (b) show two binary images both with size 512×512. And the stego-images resulting from embedding the secret data and the authentication signals are shown in Figures 1.2.11 (c) and (d), respectively. And Figures 1.2.11 (e) and (f) show their differences in black pixels after embedding the secret data and the authentication signals, respectively.

Figures1..2.12 (a) and (b) show two binary document images both with size 512×512. And the stego-images after embedding secret data and authentication signals are shown in Figures 1.2.12 (c) and (d), respectively. And Figures 1.2.12(e) and (f) show their difference in black pixels after embedding secret data and authentication signals, respectively.

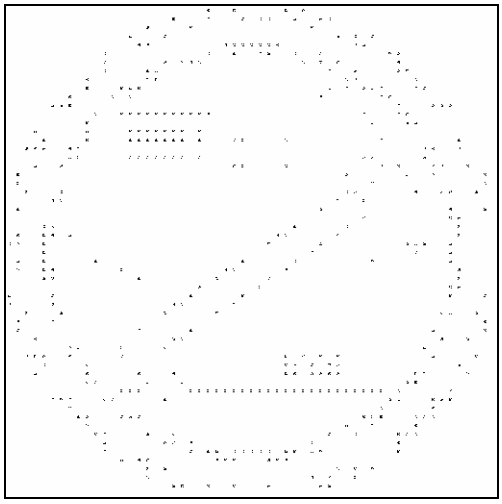
(a) (b)

Figure 1.2.8 Input binary images, output stego-images with authentication signals, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding authentication signals, respectively. (e) and (f) The difference pixels after embedding authentication signals, respectively.

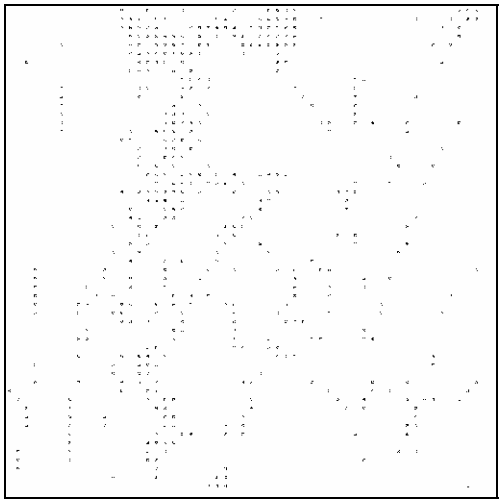


(c)

(d)

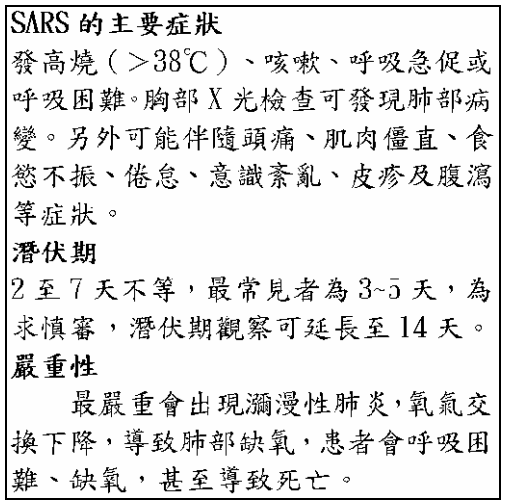


(e)

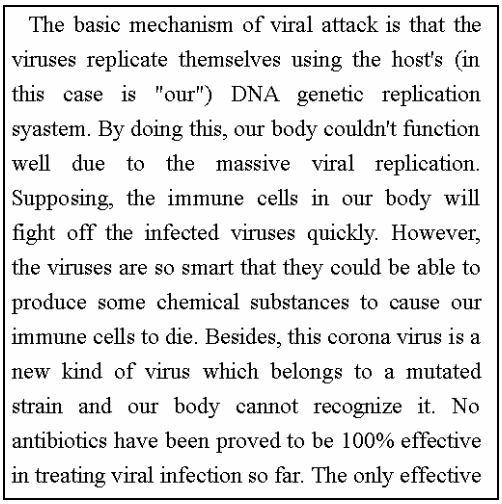


(f)

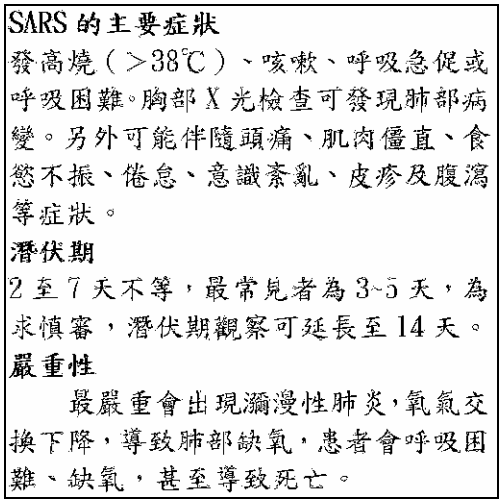
Figure 1.2.8 Input binary images, output stego-images with authentication signals, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding authentication signals, respectively. (e) and (f) The difference pixels after embedding authentication signals, respectively (continued).



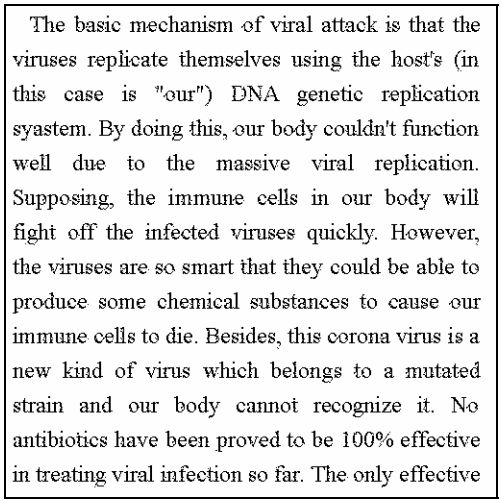
(a)



(b)

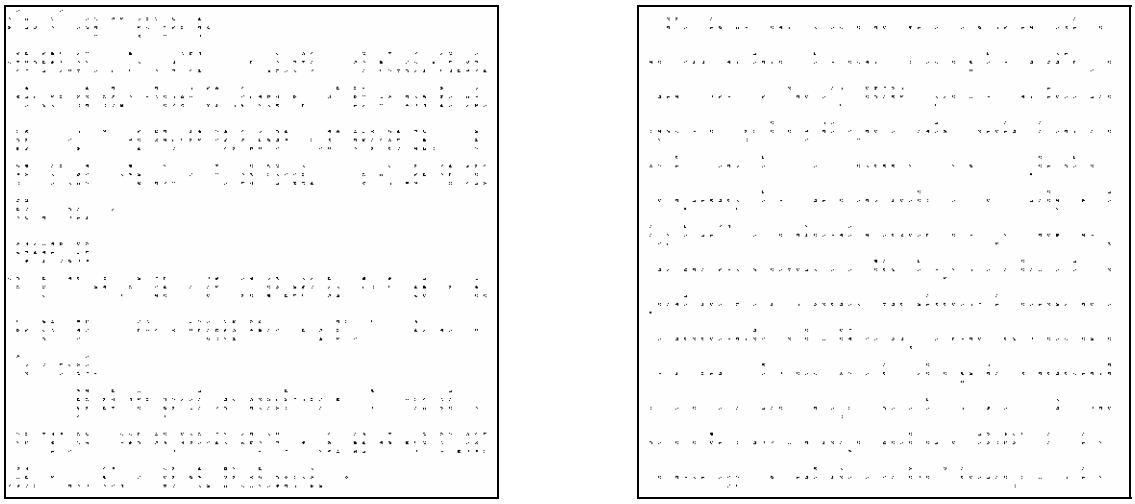


(c)



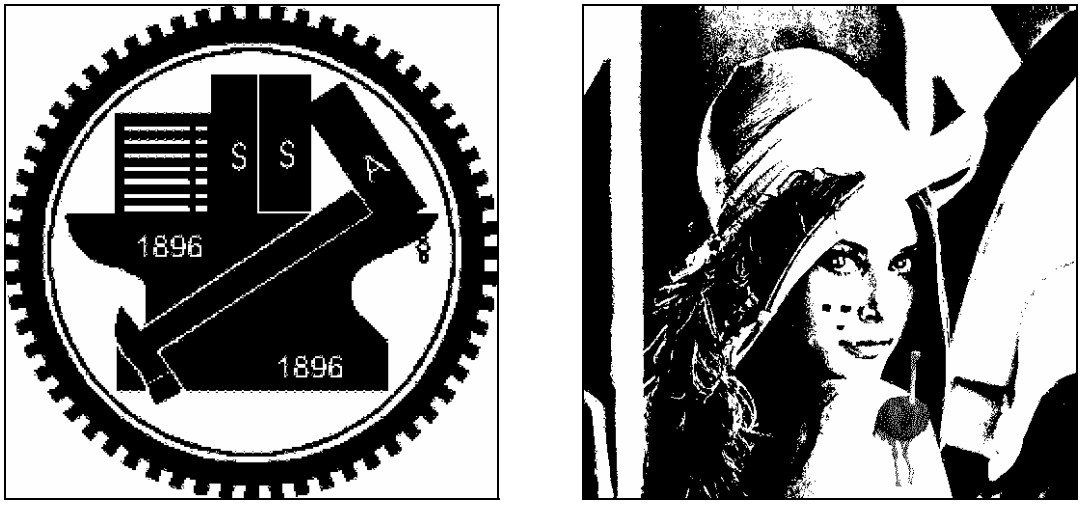
(d)

Figure 1.2.9 Input binary document images, output stego-images with authentication signals, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding authentication signals, respectively. (e) and (f) The difference pixels after embedding authentication signals, respectively.



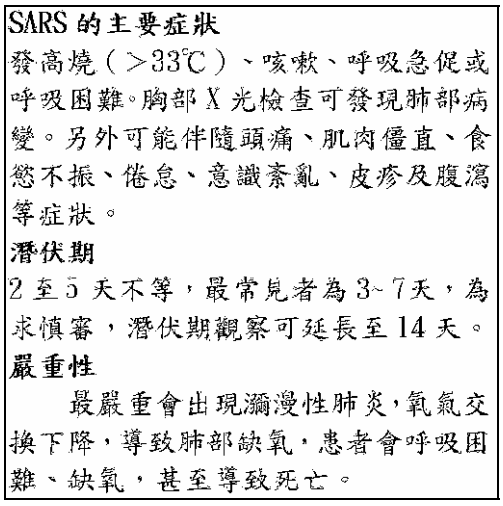
(e) (f)

Figure 1.2.9 Input binary document images, output stego-images with authentication signals, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding authentication signals, respectively. (e) and (f) The difference pixels after embedding authentication signals, respectively (continued).

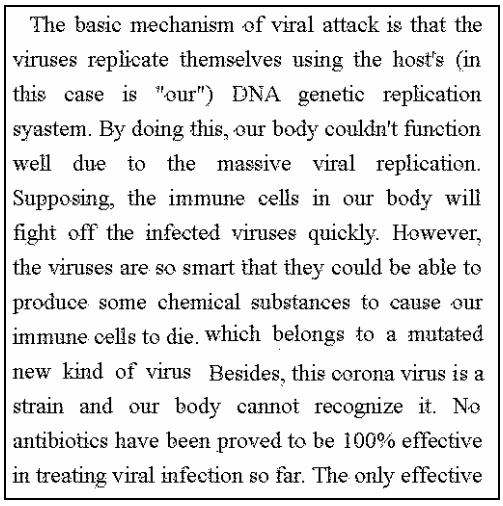


(a) (b)

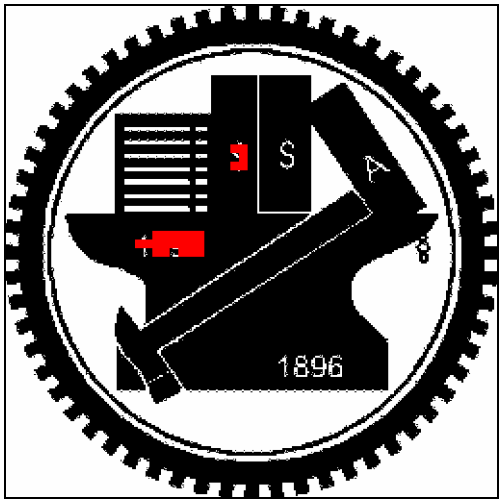
Figure 1.2.10 Some tampered images and authentication results. (a) – (c) and (d) Tampered images. (e) – (g) and (h) authentication results, respectively.



(c)



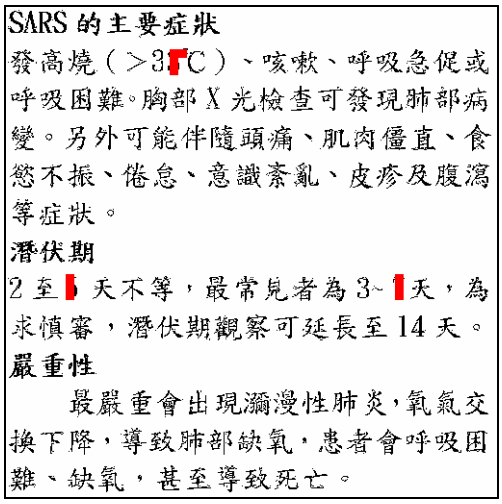
(d)



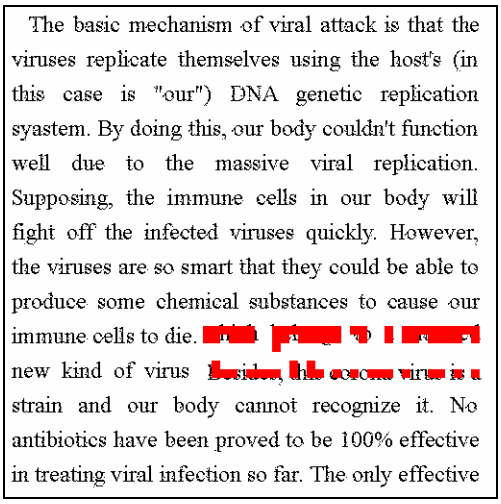
(e)

(f)

Figure 1.2.10 Some tampered images and authentication results. (a) – (c) and (d) Tampered images. (e) – (g) and (h) authentication results, respectively (continued).

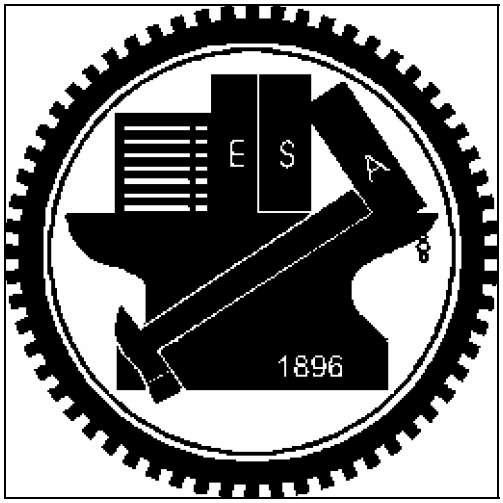


(g)



(h)

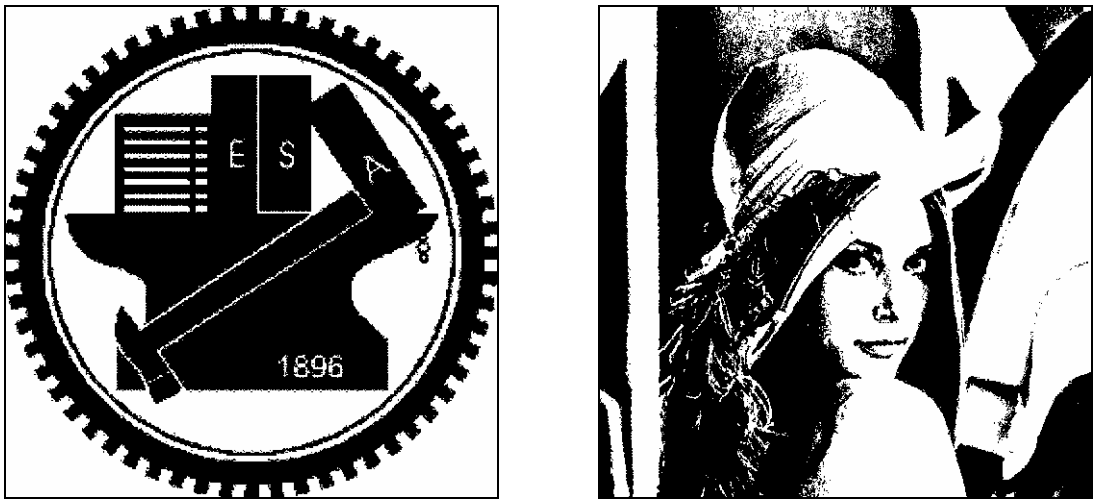
Figure 1.2.10 Some tampered images and authentication results. (a) – (c) and (d) Tampered images. (e) – (g) and (h) authentication results, respectively (continued).



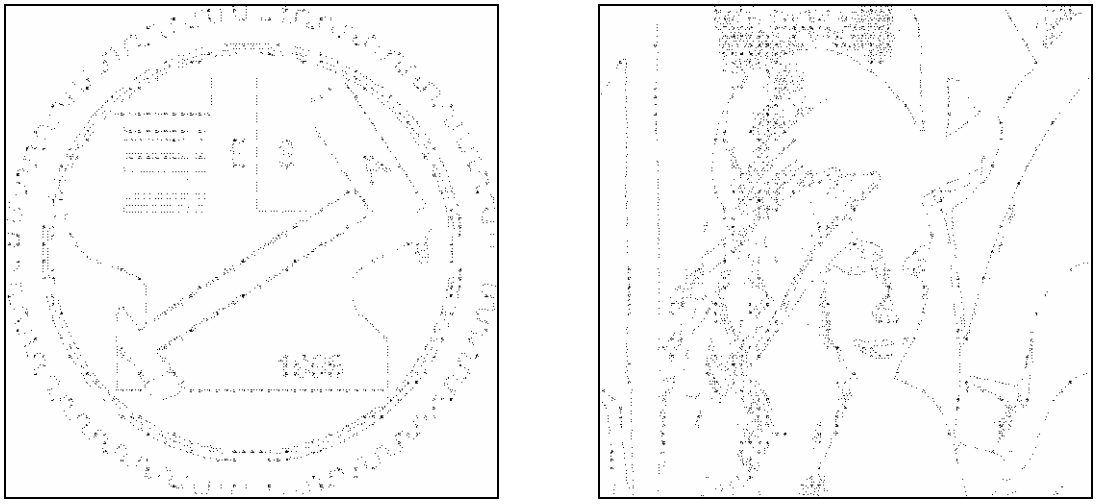
(a)

(b)

Figure 1.2.11 Input binary images, output stego-images with secret data and authentication signals, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding secret data and authentication signals, respectively. (e) and (f) The difference pixels after embedding secret data and authentication signals, respectively.



(c) (d)



(e) (f)

Figure 1.2.11 Input binary images, output stego-images with secret data and authentication signals, and the differences. (a) Binary image “NCTU”. (b) Binary image “Lena”. (c) and (d) Stego-images after embedding secret data and authentication signals, respectively. (e) and (f) The difference pixels after embedding secret data and authentication signals, respectively (continued).

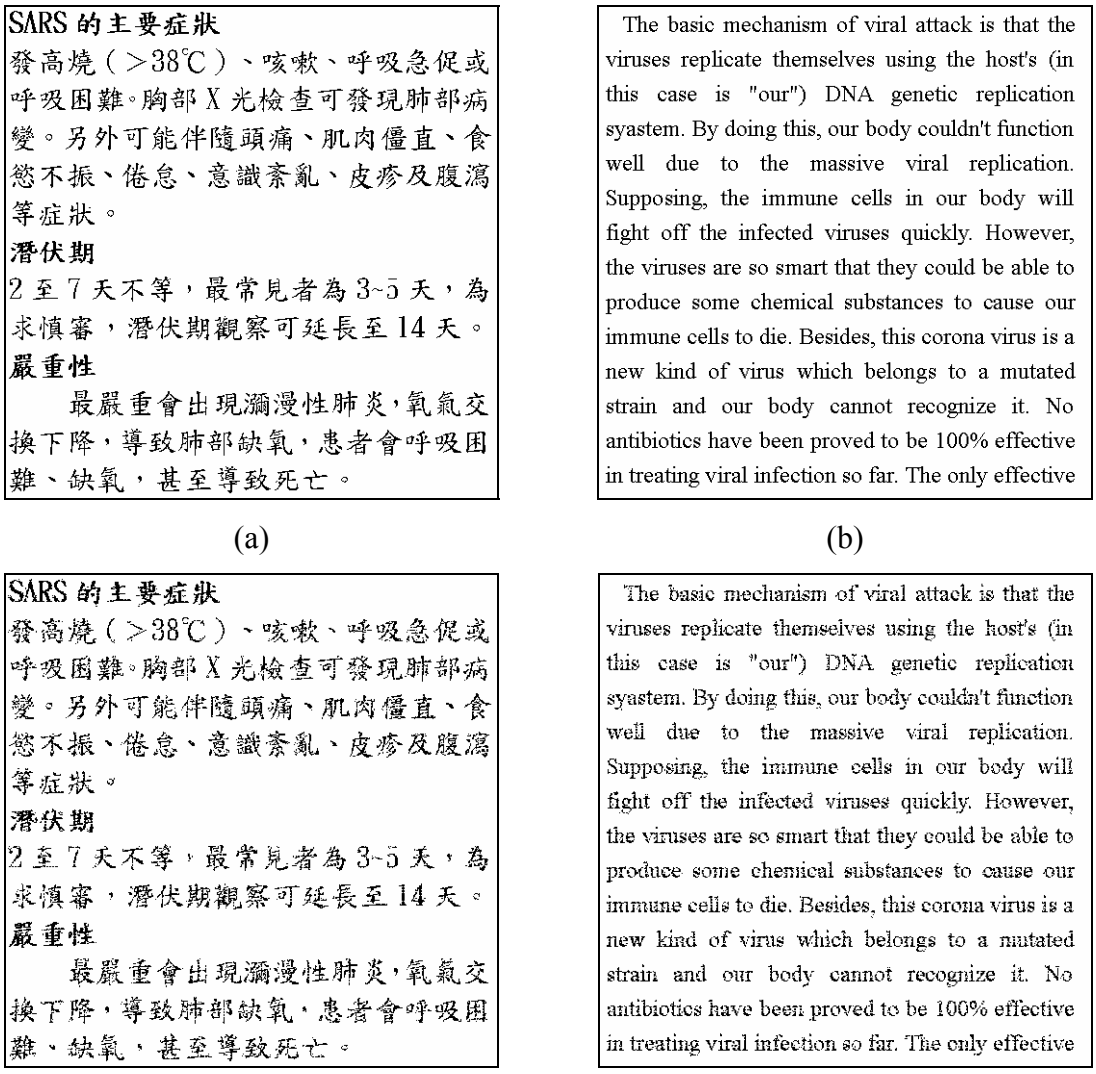
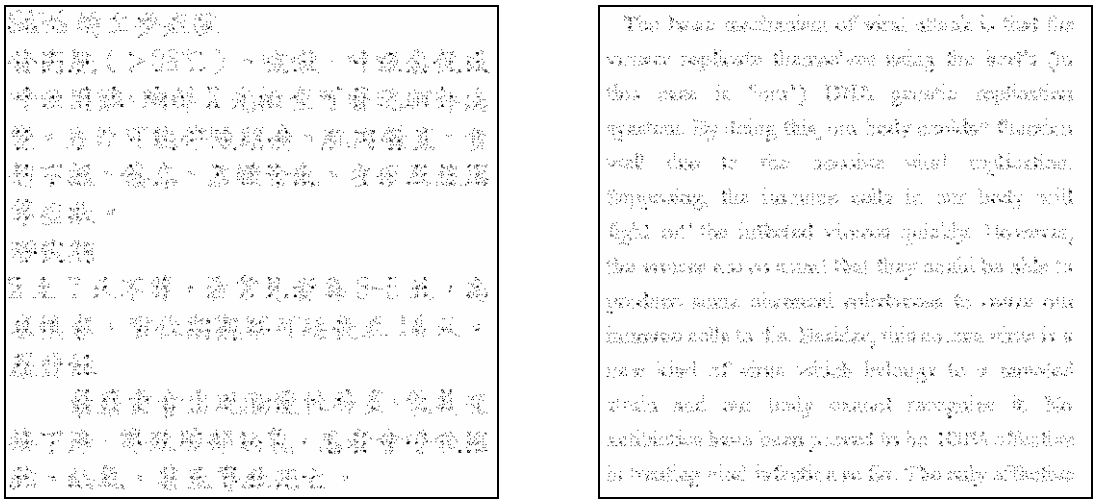


Figure 1.2.12 Input binary document images, output stego-images with secret data and authentication signals, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding secret data and authentication signals, respectively. (d) and (e) The different pixels after embedding secret data and authentication signals, respectively.



(e) (f)

Figure 1.2.12 Input binary document images, output stego-images with secret data and authentication signals, and the differences. (a) Chinese binary document images. (b) English binary document images. (c) and (d) Stego-images after embedding secret data and authentication signals, respectively. (d) and (e) The different pixels after embedding secret data and authentication signals, respectively (continued).

1.2.4 Discussions and Summary

In this chapter, we have presented a novel authentication scheme to embed authentication signals in binary images. We change the positions of white or black pixels in so-called rearrangeable blocks to obtain and embed authentication signals. That is, authentication signals are taken as the permutation of all the pixels of the rearrangeable block in each 9×9 image block. Because the authentication signal of each 9×9 image block contains certain relationship contributed by the standard deviation of the RHG values of other eight 3×3 blocks in the 9×9 block, if somebody wants to tamper with the stego-image, we can get a different standard deviation value and different authentication signals in each 9×9 block. The result is that the permutation of the rearrangeable 3×3 block in the tampered image may be not the same as the calculated authentication signals. Therefore, by checking the permutation, the tampered areas can be detected and located.

We can apply this method in the proposed data hiding method to yield a *combined* data hiding and authentication method. We can authenticate whether the image with the embedded data is tampered with or not and then decide whether it is all right to extract the hidden data.

However, in the proposed authentication signal embedding methods, we do not deal with entirely black or entirely white blocks. Therefore, if somebody replaces part of a stego-image with an entirely black or entirely white region, the tampered region cannot be located. In future works, it may be tried to solve this problem.

Chapter 2

Development of Data Hiding Techniques and Applications for Binary Document Images

2.1 Hiding Authenticable General Digital Information behind Binary Document Images with Reduced Distortion

2.1.1 Abstract

Binary document images are the images scanned or digitalized from printing or manuscripts. The significant feature of binary document images is that they have white blocks with large areas. A new approach to information hiding in binary document images with the capabilities of authenticating hidden digital data and reducing distortion effects in resulting stego-images due to data embedding is proposed. The information, which may be hidden, is general in type, and so may be of any form of secret bit streams. Based on a new feature called surrounding edge count for measuring the structural randomness in an image block, pixel embeddability is defined from the viewpoint of reducing the distortion caused by embedded pixel values. Accordingly, embeddable image pixels suitable for hiding secret data are selected. Furthermore, an error-correcting scheme is used both for extracted data authentication and embedding distortion reduction. Finally, to increase the security of embedded data, a secret key and a random number generator are also employed to randomize the locations of selected cover image pixels into which secret data are embedded. Experimental results show the feasibility of the approach for real applications.

Key words: secret hiding, secret recovery, secret authentication, binary images, error-correcting schemes, distortion reduction, surrounding edge count, pixel embeddability.

2.1.2 Introduction

Information hiding behind digital images has many applications, including covert communication, copyright protection, annotation association, etc. However, it is generally difficult to hide information behind binary document images. There are at least three reasons for this problem. First, embedding data in a binary cover image will cause obvious image content changes because of the binary (black and white) nature of the image. This indicates that reduction of image distortion due to data embedding (called embedding distortion in the sequel) should be taken as a major consideration in designing data hiding algorithms. Next, binary document images are more fragile to disturbances or attacks like channel noise or image operations. Such a characteristic makes authentication of recovered hidden information a required work. Finally, with the widespread use of color images, binary document images today are used mainly for conveying text or graphic based document images in which color is not important information, and so the semantics of binary image contents are very vulnerable to pixel value changes due to data hiding. This means that a more careful selection of image pixels for data hiding is required; pixel value changes leading to obvious destruction of image contents should be avoided. In this paper, we propose an information hiding method which takes the above three requirements into consideration. Moreover, the digital information that can be hidden by the method is general in type, which we assume to be secret bit streams in the sequel.

There were only a few studies in the past about information hiding behind binary document images, possibly due to the difficulty mentioned above. Wu, et al. [1] embedded bits in image blocks selected by pattern matching. The method can be used both for data hiding and for image authentication. Tseng, et al. [2] changed pixel values in image blocks and mapped block contents into the data to be hidden. In [3, 4], word or line spaces in textural document images are utilized to embed watermarks for copyright protection. In [5, 6], secret information is embedded into dithered images by manipulating dithering patterns. And Koch and Zhao [7] embedded a bit 0 or 1 in a block by enforcing the ratio of the number of black pixels in the block to that of white ones to be larger or smaller than the value 1, respectively. The method proposed in this study may be used for data hiding as well as for data authentication.

More specifically, in the proposed approach we define a measure of pixel embeddability by which we can select suitable cover image pixels for embedding secret data. This measure is defined in such a way that embedding distortion can be reduced, and that pixels selected for data embedding can be

identified correctly later for secret recovery. We also employ an error-correcting scheme to encode secret stream before secret embedding for the purposes of authenticating the extracted secret stream as well as reducing embedding distortion in a more global way. At last, we propose the use of a secret key as well as a random number generator to randomize the locations of the selected pixels for data embedding. This enhances the safety of the hidden data from being attacked or accessed illicitly. Based on these measures of distortion reduction and safety protection, processes for secret hiding and recovering are proposed. Some experimental results are also included to show the effectiveness the proposed method.

In the remainder of this section, we first describe the proposed processes of secret hiding and recovering in Section II, followed by the descriptions of the details of the involved measures for distortion reduction and safety protection in Section III. Some experimental results are given in Section IV, followed by a conclusion in Section V.

2.1.2.1 Proposed Secret Embedding and Recovering Processes

In the proposed approach, we hide a given secret bit stream behind a cover binary image in a random fashion controlled by a secret key and a random number generator. The proposed secret hiding and recovering processes are described in this section. Only basic ideas are included; the details of the involved terms and techniques will be explained in the next section. In the sequel, by embedding a value v into a pixel p, we mean to replace the value of p with v; and by extracting a value v from p, we mean to take v to be the value of p.

Algorithm 1. ***Secret hiding process***.

Input: a secret bit stream S, a cover image I, a secret key K, a random number generator g, and three pre-selected positive integer numbers m, n, and t.

Output: a stego-image I' in which S is embedded.

Steps:

1. Take sequentially m bits of S and encode the bits using a t-error-correcting scheme to form an n-bit substream s.
2. Create a set C of n-bit streams from s by changing at most t bits in s in all possible ways.
3. Select an ordered sequence E of n embeddable pixels in I randomly using g with K as the seed.
4. Select from C a substream sopt, which causes minimum distortion, when embedded into E (as described in the next section).
5. Embed the bits of sopt sequentially into the pixels of E.

6. Repeat Steps 1 through 5 until all bits in S are processed.

For convenience, in the sequel each pixel selected to be included in E in Step 3 above is said to have been visited. On the other hand, the proposed secret recovering process (including secret bit stream extraction and authentication) is described as follows.

Algorithm 2. **Secret recovering process**.

Input: a stego-image I' presumably including a secret bit stream S; and the secret key K, the random number generator g, as well as the positive integer numbers m, n, and t all used in Algorithm 1.

Output: the secret bit stream S or the report of failure to recover the secret. Steps:

1. Select an ordered sequence E of n embeddable pixels in I' using g with K as the seed.
2. Extract a bit b' from each pixel p in E with value v by setting b' = v, and compose all the n extracted bits sequentially to form a bit stream s'.
3. Decode s' by the t-error-correcting scheme used in Algorithm 1 to recover an m-bit secret stream s". If more than t errors are found in s' during the decoding process, decide the bits of s" to be unauthentic, yield a report of failure to recover the secret, and exit; otherwise, take s"as part of the desired secret bit stream S.
4. Repeat the above steps to extract other m-bit substreams to compose the remaining part of S until done.

The ordered sequence E of pixels selected in Step 1 above presumably should be identical to that yielded in Step 3 of Algorithm 1 to ensure that the secret bit stream can be extracted correctly. For this to be true, in addition to requiring the use of the same random number generator g and the same secret key K in the two processes as already done, an extra condition is that the embeddability of the selected pixels must be preserved after the secret hiding process, and not be changed before the secret recovering process. We satisfy this condition by proposing a proper definition of pixel embeddability, as described in the next section.

2.1.2.2 Proposed Pixel Embeddability And Distortion Reduction Measures

A. Pixel Embeddability based on surrounding edge count

In the above secret hiding process, we select embeddable pixels from a cover image to embed secret bits. We define pixel embeddability in this section from the viewpoint of reducing embedding distortion. First, we propose a new type of feature, called surrounding edge count and abbreviated as SEC. Let B be a 3×3 block in a cover image I with pixel p being its center and pixels p1, p2, …, and p8 being the eight surrounding neighbors of p in B. The SEC of p, denoted as SECp, is defined as the number of existing edges between p and its eight neighbors in B. Since the cover image I is binary, the existence of an edge between p with value v and one of its neighbors, say pi with value vi, means that |vi − v| = 1, and the reverse situation means that |vi − v| = 0. This in turn means that SECp may be computed by

$$SEC_p = \sum_{i=1}^{8} |v_i - v| .$$

The SEC value of p is a measure of the structural randomness in the block from the centralized viewpoint of p. By definition, the SEC value of a fully black or white block is 0 (no edge exists), that of a block filled with a checker pattern is 4 (four edges exist), and that of a white (or black) pixel surrounded by eight black (or white) neighbors is 8 (eight edges exist).

Next, we define a measure of distortion resulting from complementing the value v of p by:

$$\Delta SECp = |SECp - SECp\ '|$$

where SECp and SECp ' denote the SEC values before and after the complementation operation, respectively. It is not difficult to figure out that the above measure of distortion is just the amount of the resulting change of the edge numbers in B. As an example, if the central pixel p of a fully black block is changed to be a white pixel, the above distortion value ΔSECp will have the largest possible value 8, which cannot be endured because the new white central pixel is too contrastive to its eight black neighbors.

Finally, we define a pixel p in a block to be embeddable (i.e., suitable for embedding a bit value) if the following two conditions are satisfied:

(a) ΔSECp < Td; and

(b) p and its eight neighbors in B have not been visited yet, where Td is a pre-selected threshold value. Condition (a) above restricts the distortion introduced by the complementation of p's value to be sufficiently small, so that the resulting image quality will not be affected too much. And Condition (b) requires embeddable pixels to be disconnected from one another (by at least one pixel in distance), so that pixel value changes due to secret embedding will not be clustered or propagated to cause obvious

larger-sized visual artifacts.

It is pointed out that pixel embeddability defined as above can be preserved after the secret hiding process. The existence of this embeddability preserving property is briefly explained as follows. First, since the pixel p and its eight neighbors are required by Condition (b) to be unvisited yet, this means that the neighbors' values will not be altered after p is visited and labeled to be embeddable. This in turn means that the value $\Delta SECp$ will be fixed, and so Condition (a) will hold, after the secret hiding process. As a result, after a secret bit is embedded into an embeddable pixel p in the secret hiding process, p will still be embeddable in the secret recovering process, guaranteeing that the embedded secret bit stream can be extracted correctly.

B. Authentication of extracted secret bit streams

In Step 3, we recover secret substreams and authenticate them simultaneously using a t-error-correcting scheme [8]. The authentication capability of the scheme is explained here. In the encoding stage, the scheme appends several extra bits, forming a redundant checking part, to a bit sequence, called the message part. Each extra bit in the redundant checking part is a parity bit computed from a certain number of bits at certain specific positions in the message part. Then, in the decoding stage, if some bits in the two parts are changed, the computed values of the parity bits will not match those in the redundant checking part, and errors can thus be found out. In this way, authentication of the message part, which, in our case here, is the secret stream, can be achieved.

C. Further reduction of embedding distortion

By using the error-correcting scheme, errors in extracted secret data not only can be detected, but also can be corrected. In this study, we adopt the BCH method [8] for such an error correction function in the scheme. And this error-correcting capability is utilized in Step 4 in the above secret hiding process to select a so-called optimal substream sopt for further reduction of embedding distortion from a global view. The details are described in the following.

After embedding an n-bit secret substream $s = b_1b_2\ldots bn$ into n pixels $p_1, p_2, \ldots, pn$ in a selected embeddable pixel sequence E, we compute a measure of the total embedding distortion, denoted as D(s), by

$$D(s) = \sum_{i=1}^{n} \Delta SEP_{p_i}$$

Also, as described in Step 2 in the secret hiding process, we create from s a set C of n-bit streams by changing at most t bits in s in all possible ways. For each stream si in C, we compute similarly another total embedding distortion value D(si). Then, we choose as sopt the stream si in C with the smallest D(si), and embed sopt into E, as done in Step 3 of the secret hiding process. Though there might exist at most t errors in sopt, but by the t-error-correcting scheme, s still can be correctly recovered from sopt as done in Step 3 of the secret recovering process. Because of the freedom of the choice of sopt from $\sum_{r=0}^{t} \binom{n}{r}$ totally candidate streams, it may be expected that the embedding distortion can be reduced further

2.1.3 Experimental Results

A large number of binary document images, including several typical images for binary image compression standards, were used in our experiments. An example of the experimental results is shown in Figure 2.1.1. Figure 2.1.1 (a) shows a cover image of size 256×256 with machine-printed as well as handwritten characters, and line drawings. And Figure 2.1.1 (b) shows the stego-image resulting from embedding 630 secret bits into Figure 2.1.1(a) using the proposed method with m, n, t, and Td being 4, 15, 5, 3, respectively. The difference between Figure 2.1.1 (a) and Figure 2.1.1 (b) is illustrated in Figure 2.1.1(c) as black pixels. The gray pixels are included just for the purpose of comparison and are not part of the difference. It can be seen that Figure 2.1.1 (a) and Figure 2.1.1 (b) are visually close to each other; no obvious distortion can be observed. Another example of the results is shown in Figure 2.1.2, which demonstrates the effectiveness of the proposed technique for reducing embedding distortion. Figure 2.1.2(a) shows a 64×192 cover image. Two stego-images resulting from embedding a 60-bit secret stream into Figure 2.1.2(a) without and with the use of the proposed distortion reduction technique using the t-error-correcting scheme are shown in Figure 2.1.2 (b) and Figure 2.1.2 (c), respectively. It can be noted that the visual quality of Figure 2.1.2 (c) is better than that of Figure 2.1.2 (b). Figure 2.1.3 shows the effect of distortion reduction using the error-correcting scheme with different values of t. One curve in the figure specifies the average DSEC yielded in the secret hiding process, and

the other curve specifies the average number of changed pixels, computed as the ratio of the number of changed pixels to that of embedded secret bits. Both curves show that the distortion is decreased with the increase of the error-correcting capability specified by t.

2.1.4 Conclusion

A new approach to data hiding in binary document images has been proposed, which may be employed to hide general secret data streams behind binary document images with the additional capability to verify the authenticity of extracted data. Reduction of embedding distortion is the major consideration in the approach. The first measure for this goal is the proposal of pixel embeddability based on the new feature of SEC, which makes data hidden in embeddable pixels less noticeable. Computation of SEC values does not require excessive works like pattern matching, and so is efficient. Another measure proposed for distortion reduction from a more global view is the use of the error-correcting scheme for creating a distortion-minimizing secret stream from the original one. This merit is not found in other approaches dealing with data hiding in binary document images. Our experimental results also reveal its effectiveness. In addition, the use of the error-correcting scheme also provides the ability to verify the authenticity of extracted data. Finally, because of the randomization policy employed for selecting embeddable pixels as well as the nature of the proposed pixel embeddability, embedded values are spread in the entire image and disconnected from one another, so that pixel changes will not be clustered and thus less hints for the embedded secret will be revealed.

References

[1] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary images," presented at the *IEEE International Conference on Multimedia and Expositions*, New York, 2000.

[2] Y.-C. Tseng, Y.-Y. Chen, and H.-K. Pan, "A secure data hiding scheme for binary images," *IEEE Transactions on Communications*, vol. 50, no. 8, pp. 1227-1231, August 2002.

[3] S. H. Low, N. F. Maxemchuk, and A. M. Lapone, "Document identification for copyright protection using centroid detection," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 372-383, March 1998.

[4] D. Huang and H. Yan, "Interword distance changes represented by sine waves for

watermarking text images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp1237-1245, December 2001.

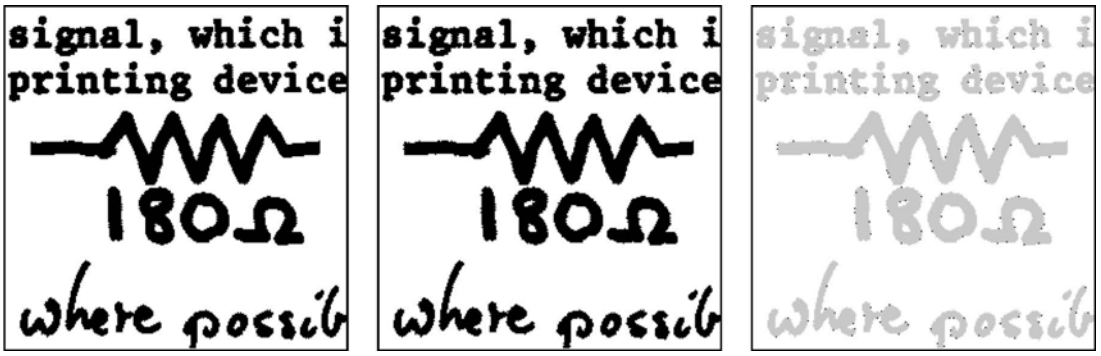
[5] K. Matsui and K. Tanaka, "Video-steganography: how to secretly embed a signature in a picture," Proceedings of IMA Intellectual Property Project, vol. 1, no. 1, 1994.

[6] H. C. Wang, "Data hiding techniques for printed binary images," Proceedings of the IEEE International Conference on Information Technology: Coding and Computing, Las Vegas, NV, USA, April 2001, pp. 55-59.

[7] E. Koch and J. Zhao, "Embedding robust labels into images for copyright protection,"

Proceedings of International Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Techniques, Munich, Germany, 1995, pp. 242-251.

[8] S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.



(a) (b) (c)

Figure 2.1.1 An experimental result of the proposed method: (a) a cover image; (b) the stego-image resulting from embedding 630 secret bits into (a); (c) the difference between the cover image (a) and the stego-image (b) shown as black pixels.

(a) (b) (c)

Figure 2.1.2 Embedding results yielded with and without proposed embedding distortion reduction: (a) the cover image; (b) the stego-image yielded without distortion reduction; (c) the stego-image yielded with distortion reduction

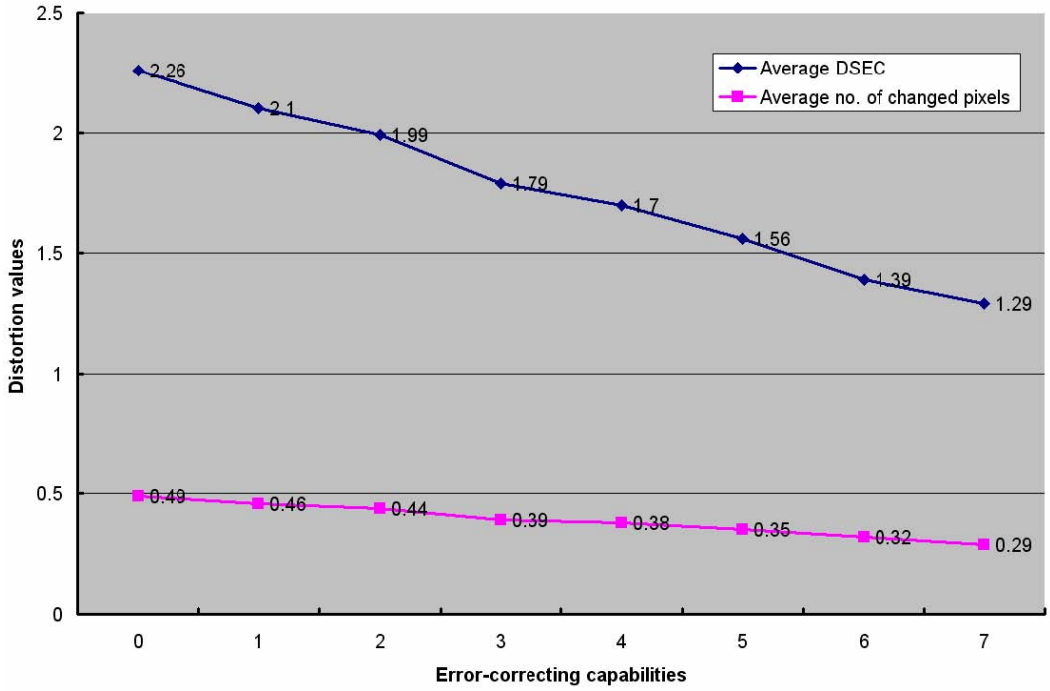


Figure 2.1.3 Curves showing the decrease of embedding distortion with the increase of theerror-correcting capability specified by the number *t* of corrected bits.

2.2 A New Approach to Authentication of Binary Document Images for Multimedia Communication with Distortion Reduction and Security Enhancement

2.2.1 Abstract

A new approach to binary image authentication in multimedia communication with distortion reduction and security enhancement is proposed. Special codes are embedded into the blocks of given images and verified to accomplish the authentication purpose. Enhancement of security in detecting tampered images is achieved by randomly generating the codes and embedding them into randomly selected locations in the image blocks. And reduction of image distortion coming from pixel value replacement in code embedding is carried out by allowing multiple locations for embedding the codes. Security analysis and experimental results are also included to show the effectiveness of the proposed approach.

Key words: image authentication, binary document images, authentication codes, code embedding and verification, distortion reduction, security protection.

2.2.2 Introduction

Image transmission is a major activity in today's communication. With the advance of digital technologies, it is now easy to modify digital images without causing noticeable changes, resulting possibly in illicit tampering of transmitted images. It is so desirable to design effective algorithms for *image authentication*, aiming at checking the fidelity and integrity of received images.

This authentication problem is difficult for binary document images because of their simple binary nature. Embedding of authentication signals into binary document images will cause destruction of image contents, and so arouses possible suspect from invaders. Therefore, a good solution should take into consideration not only the security issue of reducing the possibility of being tampered with imperception but also the effectiveness of reducing image distortion resulting from authentication signal

embedding. In this study, we propose an authentication method for binary document images with a good balance between the mutually conflicting goals of distortion reduction and security enhancement.

So far, there are very few researches about binary image authentication, though some works on hiding data in binary images have been reported. Tseng, et al. [1] mapped block contents into the data to be hidden. Wu, et al. [2] embedded bits in image blocks by pattern matching. The method can be used for image authentication. In [3, 4], secret data were embedded into binary document images by manipulating dithering patterns. In [5, 6], spaces in textural document images were used for embedding copyright-protecting watermarks.

The method proposed in this paper for binary image authentication is based on the idea of embedding randomly-generated codes, called *authentication codes*, into the blocks of a given *cover image*, resulting in a *stego-image*. Authentication is achieved by verifying the codes in the blocks of a given stego-image. Tampering of a stego-image block will destroy the code in the block and cause an erroneous verification result. To reduce image distortion resulting from embedding codes in a cover image, a new technique of allowing more than one pixel group, called a *code holder*, for embedding a code is proposed, and the *optimal code holder* whose pixel values are minimally different from those of the authentication code is chosen to embed the code. Detailed analysis of the probability for a tampered image to be verified to be authentic is also conducted. The result may be utilized to decide strategies for choosing proper authentication code lengths as well as appropriate numbers of code holders.

In the remainder of this paper, we first describe the proposed authentication code embedding and verification processes in Section II, followed by the security analysis in Section III. Some experimental results are given in Section IV, followed by a conclusion in Section V.

2.2.3 Proposed Authentication Method

The input to the proposed authentication code embedding process includes a cover image I with L blocks, two keys K_1 and K_2, and two random number generators f_1 and f_2. The output is a stego-image I' in which authentication codes are embedded. The steps of the algorithm are as follows.

1. Use f_1 with K_1 as the seed to generate a sequence of L random numbers $c_1, c_2, \ldots, c_L$, each with m bits, as the authentication codes.
2. Embed each c_i into a corresponding block B_i in I in the following way to yield I':

   3.1 Use f_2 with K_2 as the seed to select randomly in B_i a certain number, say n, of code holders,

each including a certain number, say m, of *ordered* pixels.

3.2 Compare c_i with each code holder g_k by matching respectively the m corresponding bits in c_i and g_k, and find the *optimal* code holder g_{opt} with the minimum number of different bit values.

3.3 Replace, if necessary, the value of each bit in g_{opt} with the corresponding one in c_i to complete the code embedding work.

As an example, let the pixels in a given 3×3 image block B in a raster scanning order be denoted as P_1 through P_9 whose contents, when concatenated, are 011110010. A 2-bit authentication code c = 01 generated by f_1 is to be embedded in B. Assume that three code holders H_1, H_2, and H_3 are allowed, which are decided by f_2 to be the pixel pairs (P_1, P_5), (P_2, P_9), and (P_5, P_1), respectively. Note that the two pairs H_1 = (P_1, P_5) and H_3 = (P_5, P_1) are *distinct* as the pixels in each pair are regarded to be *ordered*. Accordingly, since the contents of the three code holders in sequence are (0, 1), (1, 0), and (1, 0), the optimal code holder is just H_1 = (P_1, P_5) because the bits of c match those in H_1 exactly. And so no bit replacement is necessary when c_i is embedded into H_1.

On the other hand, the proposed authentication code verification process is as follows. The input includes a stego-image I' with L blocks, as well as the two keys K_1 and K_2 and the two random number generators f_1 and f_2 used in the authentication code embedding process. The output is an authentication report for I'.

1. Re-generate the L m-bit authentication codes c_1, c_2, …, c_L using f_1 and K_1.
2. Verify each c_i in a corresponding block B_i in I'in the following way:
   2.1 Use f_2 and K_2 to re-select in B_i the n m-pixel *code holders*.
   2.2 Compare c_i with each code holder g_k by matching the m corresponding bits in c_i and g_k, and label B_i as *tampered* if there exists no code holder with content identical to c_i; or as *authentic*, otherwise.
3. If there exists any block being labeled tampered, report negative fidelity and output all tampered blocks; otherwise, regard the entire image of I' untampered.

2.2.4 Security Analysis

First, we want to analyze the probability that a tampered image block B is erroneously verified to be authentic. Recall that we embed in each block an authentication code c with m bits into one of n

allowed code holders. If *c* is found in any of the *n* holders, we decide the authentication to be successful. The probability for a tampered bit (0 or 1) in *B* found in a code holder *H* to be authentic, i.e., to be with a value identical to the corresponding bit in the stego-image, is obviously ½, and so the probability for all the *m* tampered bits in *H* to be authentic is $(½)^m$. This means that the reverse probability for a code holder to be *safe* is $1 - (½)^m = \frac{2^m - 1}{2^m}$. There are *n* code holders in *B*, therefore the probability for all the *n* code holders to be safe is $\left(\frac{2^m - 1}{2^m}\right)^n$. Inversely, the probability for the tampered block *B* to be verified to be authentic is accordingly $1 - \left(\frac{2^m - 1}{2^m}\right)^n$

Now, we can compute the probability for a tampered image with *L* blocks to be verified to be authentic, which is just $\left(1 - \left(\frac{2^m - 1}{2^m}\right)^n\right)^L$. This probability, called *erroneous authentication probability* and abbreviated as EAP subsequently, has nothing to do with the size of the image block.

For example, if we choose 6 bits as the authentication code length and allow 8 code holders for a tampered image with $L = 64$ blocks, then the EAP is $\left(1 - \left(\frac{2^6 - 1}{2^6}\right)^8\right)^{64} \approx (0.11837)^{64} \approx 4.79 \times 10^{-60}$ which is negligibly small. That is, the safety of the proposed scheme is no problem in this case.

The tradeoff between distortion reduction and security enhancement can be checked from the formula $\left(1 - \left(\frac{2^m - 1}{2^m}\right)^n\right)^L$ for computing the EAP. To reduce image distortion, we have to allow more code holders (i.e., to allow large *n* values), use less bits in the authentication code (i.e., to decrease the value of *m*), and partition the image into less blocks to restrain code embedding (i.e., to make the value of *L* smaller). These activities all result in an increase of the EAP as can be seen from the formula $\left(1 - \left(\frac{2^m - 1}{2^m}\right)^n\right)^L$. On the other extreme, we may take *m* to be as large as possible (so that $\frac{2^m - 1}{2^m}$ approaches 1) and allow just a single code holder ($n = 1$). Then the EAP will approaches 0, meaning that an invader has almost no chance to pass authentication with a tampered image. But the price we pay is that the content of each image block has almost been destructed. A compromise obviously must be considered, and the optimal choice is of course problem-dependent.

In case it is desired to protect the image to the block detail from *partial* image tampering, we have to consider the EAP value for a single block, which is $1-\left(\frac{2^m-1}{2^m}\right)^n$. Then, selections of n and m become critical. For $n = 8$, $m = 6$, and the EAP is $1 - [(2^6 - 1)/(2^6)]^8 \approx 0.11837$. If this is unsatisfactory to applications with higher security requirements, selections of a smaller n, say 6, and a larger m, say 10, may be considered, if the block size is larger enough to hold the code. This will result in an EAP value of $1 - [(2^{10} - 1)/(2^{10})]^6 \approx 0.00585$ which is reasonably small and safe, but at the sacrifice of increasing image distortion due to less code holder choices and more bit replacements.

2.2.5 Experimental Results

An example of experimental results of the proposed method is shown in Figure 2.2.1. Figure 2.2.1(a) shows a cover image of size 512×512. Figures 2.2.1(b) and 2.2.1(c) show two stego-images resulting from embedding authentication codes into Figure 2.2.1(a) with $m = 8$, $n = 2$ and with $m = 8$ and $n = 5$, respectively. The block size was selected to be 64×64. It can be seen that the image in Figure 2.2.1(c) includes less distortion than Figure 2.2.1(b) because of the use of more code holders. Figure 2.2.1(d) shows an image resulting from tampering Figure 2.2.1(b). And Figure 2.2.1(e) shows the result of authentication in which tampered blocks are marked in gray.

2.2.6 Conclusion

A new approach to binary image authentication for distortion reduction and security enhancement has been proposed. It may be employed to design application-dependent schemes for embedding, from the viewpoint of enhance the security of the image, proper amounts of authentication codes into image blocks and then verifying them for fidelity and integrity checks without introducing unacceptable distortion. The security analysis and the experimental results show the feasibility of the proposed approach. Extensions of the approach to other image types may be tried in the future.

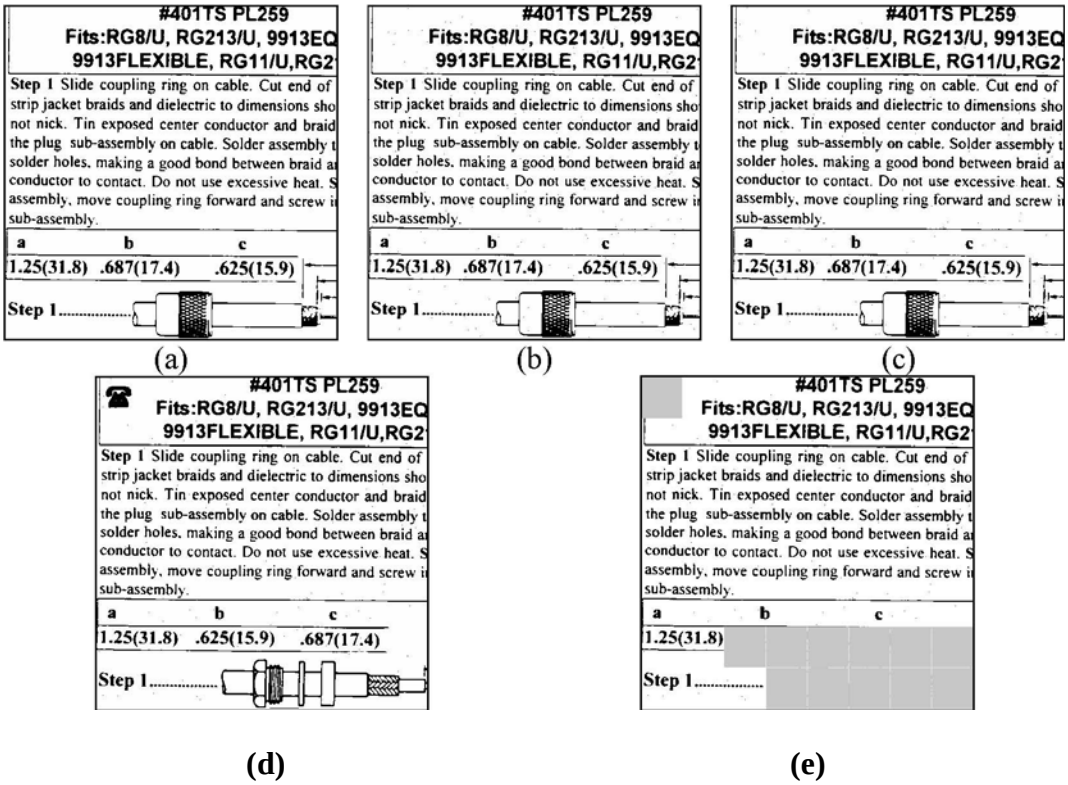


Figure 2.2.1. An experimental result: (a) a cover image; (b) a stego-image resulting from embedding authentication codes into (a) with $m = 8$ and $n = 2$; (c) another stego-image resulting from embedding authentication codes into (a) with $m = 8$ and $n = 5$; (d) a tampered version of (b); (e) authentication result of (d) with tampered blocks marked in gray.

References

[1] Y.-C. Tseng, Y.-Y. Chen, and H.-K. Pan, "A secure data hiding scheme for binary images," *IEEE Transactions on Communications*, vol. 50, no. 8, pp. 1227-1231, August 2002.

[2] M. Wu, E. Tang, and B. Liu, "Data hiding in digital binary images," presented at the *IEEE International Conference on Multimedia and Expositions*, New York, 2000.

[3] K. Matsui and K. Tanaka, "Video-steganography: how to secretly embed a signature in a picture," *Proceedings of IMA Intellectual Property Project*, vol. 1, no. 1, 1994.

[4] H. C. Wang, "Data hiding techniques for printed binary images," *Proceedings of the International Conference on Information Technology: Coding and Computing*, Las Vegas, NV, USA, April 2001, pp. 55-59.

[5] S. H. Low, N. F. Maxemchuk, and A. M. Lapone, "Document identification for copyright protection using centroid detection," *IEEE Transactions on Communications*, vol. 46, no. 3, pp. 372-383,

March 1998.

[6] D. Huang and H. Yan, “Interword distance changes represented by sine waves for watermarking text images,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1237-1245, December 2001.

Chapter 3
Image or Video Authentication for Copyright Claim

3.1 Introduction

Digital watermarking has been proposed for the purposes of copyright and integrity protection of digital media. When suspicious images or videos are discovered, with the help of the extracted watermark, the image or videos ownership can be protected. With the help of the extracted authentication signals, the image integrity and fidelity can be verified. Traditional authentication centers put their focuses on establishing a trusted environment of network transactions. However, in this chapter, some extension of services that the traditional authentication centers do not provide are described. We focus on the issue of copyright protection of images.

3.2 Proposed mechanism for Image Authentication Center

Multiple authorization mechanisms are added to the functions of image authentication centers (IACs). That is, the IAC will contain two other service functions, called authorization testimony and authorization tracking. They are described as follows:

1. Authorization testimony: When an owner distributes an authorized product to some authorized users, according to the aforementioned situation, the owner will embed an identifiable fingerprint each time in the product to create a stego-product. Then, the owner distributes each stego-product to each authorized user. The IAC can play a witness to prove the authorized procedure and ask the authorized user to sign a contract whose content is that he/she cannot release the authorized product. And the IAC can record the identifiable fingerprints and their

associated authorized users.

2. Authorization tracking: If a contract violation is detected, the IAC can trace back to the violating authorized user. It can extract the fingerprint from the detected product and capture the traitor according to the records in the IAC. Besides, for the authorization of a software package, if users attempt to replace the original copyright with their copyright in an image, the IAC also can trace back to the original owner and the violating authorized user. When a copyright is embedded into an image by a user, the embedding process will extract the original identifiable code form the image and simultaneously embed it together with the user's identifiable code into the image. That is, the owner's identifiable code and the user's one will be in the image. The IAC can extract the owner's identifiable code from the stego-images to judge who is the owner of the image and who is the cheater.

When suspicious images or videos are found, it is important to conduct image or video authentication. Image or video authentication can prove the ownership of images and verify the integrity and fidelity of images or videos. It is appropriate to set up an image or video authentication center (IAC) to play this role. An ICA with a two-level structure was proposed by this project in the last year, as shown in Figure 3.1. The top-level is a trusted third party like a court, called a central IAC and the lower-level is composed of many organizations such as museums, gallery, and so on, called local IACs. The central IAC delivers a software package for copyright and annotation protection with different identifiable codes with respect to different local IACs. A local IAC can embed its copyright or annotation data in its treasurable images by the software package. The local IAC can also register images or videos to the central IAC. When suspicious images or videos are found, the local IAC can authenticate these images. When a misappropriating user refutes the authentication result of a local IAC, the local IAC can send these images or videos to the central IAC. The central IAC can authenticate the images or videos by the registration information. According to the registration information, the central IAC can judge the image copyright. Because the central IAC is a trusted third party, it is convinced for misappropriating users.

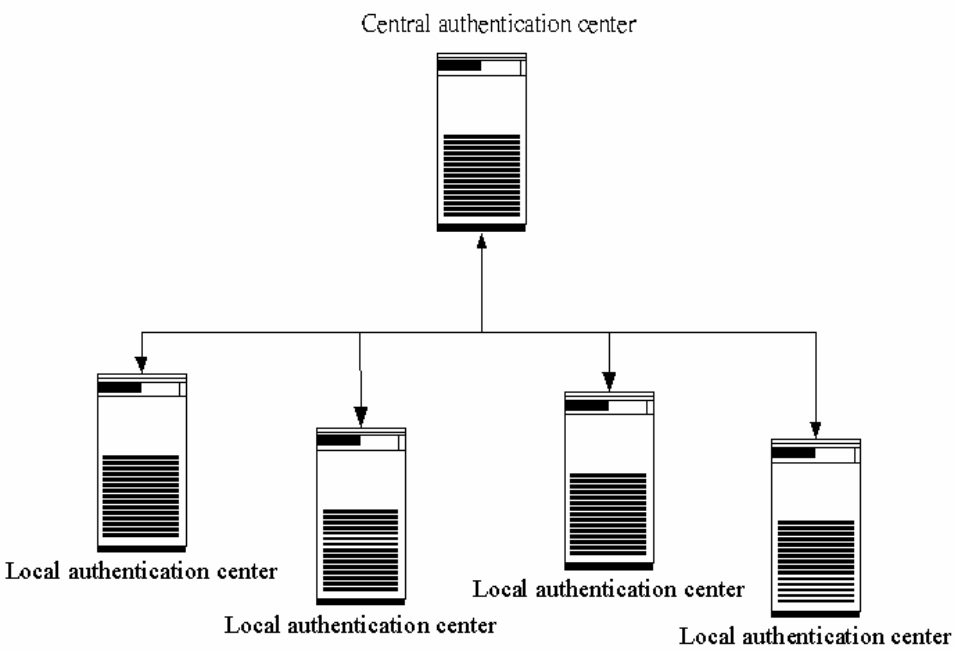


Figure 3.1 Two-level image authentication center

3.3 Discussions and Summary

Image or video authentication centers thus can support image or video authentication for copyright claim. The structure of an image or video authentication center may be divided into two parts from our viewpoint. One is the central image or video authentication center. The other is the local image or video authentication center. Besides, image or video authentication centers are suggested to fulfill several service functions, namely, image or video integrity authentication, image or video copyright verification, authorization testimony, and authorization tracking. With the capability of multiple authentications and the image or video authentication center, the ownership of the image or video will can be protected well.

附件一

參加「第三屆數位典藏研討會」發表之論文

Copyright Protection by Watermarking for Color Images against Rotation and Scaling Attacks Using Coding and Synchronization of Peak Locations in DFT Domain

Yen-Chung Chiu

Department of Computer and Information Science
National Chiao Tung University
03-5131545, Hsinchu, Taiwan 300

gis91528@cis.nctu.edu.tw

Wen-Hsiang Tsai
2nd author's affiliation
Department of Computer and Information Science
National Chiao Tung University
03-5728368, Hsinchu, Taiwan 300

whtsai@cis.nctu.edu.tw

ABSTRACT

The proposed method for copyright protection of color images against rotation and scaling attacks is described. The main idea is to embed a watermark as coefficient-value peaks circularly and symmetrically in the middle band of the DFT domain of an input image. Then, by detecting the peaks in the DFT domain, the embedded watermark can be extracted.

Keywords

watermarking, color image, rotation attack, scaling attack, DFT domain, peak location coding, synchronization peak, copyright protection.

1. INTRODUCTION

Digital watermarking is a technique for embedding a watermark into an image to protect the owner's copyright of the image. The embedded watermark must be robust. The stego-image may be rotated or scaled by illicit users. It is desirable that after applying these operations on the stego-image, the watermark is not fully destroyed and can be extracted to verify the copyright of the image.

Many different watermarking techniques for copyright protection have been proposed in recent years. Watermarking techniques that are robust to rotation and scaling are mostly performed in the frequency domain. O'Ruanaidh and Pun [1] proposed the use of Fourier-Mellin transform-based invariants for digital image watermarking. A public watermarking method based on the Fourier-Mellin transform and an extension of it based on the Radon transform was proposed by Wu, et al. [2]. In Lin, et al. [3] a watermark is embedded into a one-dimensional (1-D) signal obtained by taking the Fourier transform of the image, re-sampling the Fourier magnitudes into log-polar coordinates, and then summing a function of those magnitudes along the log-radius axis. Su and Kuo [4] proposed a spatial-frequency composite digital image watermarking scheme to make the embedded watermark survive rotation and scaling transformations. The frequency-domain watermark was embedded in the discrete Fourier transform coefficients. The spatial-domain watermarking is used to help recover the image to its original orientation and scale.

This paper is organized as follows. In Section 2, the idea of the proposed method will be described. By certain properties of the DFT coefficients, we can embed a watermark in the DFT domain with robustness against rotation and scaling attacks. In Section 3, the proposed watermark embedding process is presented. In Section 4, the proposed watermark extraction process is described. In Section 5, some experimental results are illustrated. Finally, in Section 6 some discussions and a summary are given.

2. Idea of Proposed Method

2.1 Properties of Coefficients in DFT Domain

After applying a discrete Fourier transformation (DFT) to an input image, the DFT coefficients in the frequency domain can be obtained. The DFT of an image $f(x,y)$ of size $M \times N$ can be described by the equation described below:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)} \quad (2.1)$$

The Fourier transform is a complex function of the real frequency variables. It has several properties, and some of them are described in the following.

A. Symmetry property

If a 2D signal is real, then the Fourier transform has a symmetry property, as shown by the following equation [5]:

$$F(u,v) = F^*(-u,-v) . \quad (2.2)$$

The symbol (*) indicates complex conjugation. Because the Fourier transform of an image can be complex, we can divide it into two functions. One is the *magnitude* function or spectrum $|F(u,v)| = [R^2(u,v) + I^2(u,v)]^{\frac{1}{2}}$, and the other the *phase* function $\phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right]$, where $R(u,v)$ and $I(u,v)$ are the real and imaginary parts of $F(u,v)$. And for real signals, Equation (2.2) leads to:

$$|F(u, v)| = |F(-u, -v)|. \quad (2.3)$$

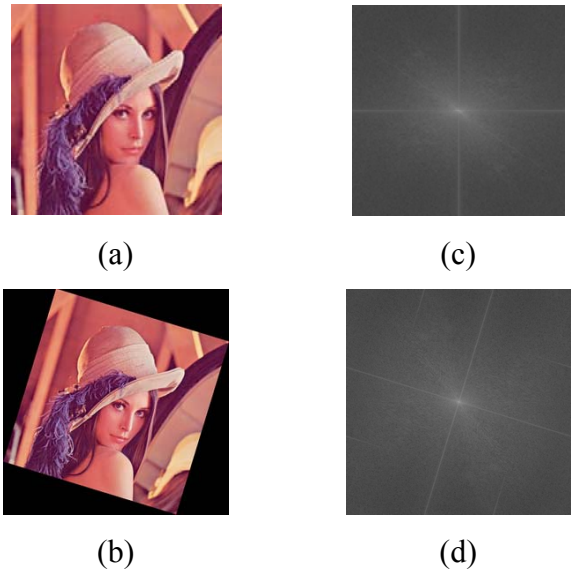
It means that the magnitude value of a coefficient (or simply a coefficient value) and its symmetric version are equal. In addition, both the magnitude and the phase functions are necessary for complete reconstruction of an image from its Fourier transform. But the magnitude part is less important than the phase part. The magnitude-only image is unrecognizable. On the contrary, the phase-only image is barely recognizable [6]. Therefore, we may calculate and adjust the magnitude values of the DFT coefficients to embed information without causing significant loss of image quality.

B. Invariant properties of rotation and scaling

After we apply some image processing operations like rotation and scaling to an image, the coordinates and magnitude values of the DFT coefficients of the image will be altered, too. Changes of the DFT coefficients after scaling and rotation operations in the discrete image domain are listed in Table 1 [7]. The scaling operation has almost no effect on the DFT coefficients. It means that when an image is scaled, each DFT coefficient is the same as the original one except only with some noise. On the other hand, after rotating an image in the spatial domain, the locations of the DFT coefficient values will have the same rotation in the DFT domain. Figures 1(a) and (b) show an original image and a rotated version of it. And the corresponding Fourier spectrum images, in which each pixel value is equal to the magnitude value of the DFT coefficient, are shown in Figures 1(c) and (d), respectively. Note that the Fourier spectrum image in Figure 1(d) has the same rotation like Figure 1(b).

Table 1. Changes of DFT coefficients after operations in discrete spatial domain.

Operations	Scaling	Rotation
Changes of DFT coefficients	Almost no effect	Rotation



(a) (c)

(b) (d)

Figure 1 Input images, and Fourier spectrum images of G channel. (a) Image "Lena". (b) Image "Lena" after rotation. (c) Fourier spectrum image of image "Lena" (d) Fourier spectrum image with the same rotation angle of (b).

2.2 Properties of Color Channels

A full-color image has three color channels, namely, red (R), green (G), and blue (B). Generally speaking, we can embed watermark information into all of these three channels. However, human eyes are less sensitive to the frequency of blue color. And its greatest sensitivity is distributed over the region of the yellow/green frequency [8]. In addition, according to experiments, a watermark can be embedded into both red and blue channels in the DFT domain without creating perceivable effects. On the contrary, hiding information in the green channel is too sensitive to human vision. If we embed the watermark in the DFT domain of the green channel, the stego-image will appear to include obvious reticular effects.

2.3 Proposed Technique for Coding Peak Locations for Watermarking

In the proposed method, after the zero frequency point $F(0,0)$ is shifted to the center of the transform domain, a watermark is embedded in a ring region which covers a middle band in the frequency domain between two circles with radiuses R1 and R2, with R1< R2, as shown in Figure 2. The middle band of the DFT domain is divided into n equally-spaced concentric circles, each with a radius $r \in R = \{r_1, r_2, \cdots, r_n\}$ and into m angle ranges with each range starting at a direction $\theta \in \Theta = \{\theta_1, \theta_2, \cdots, \theta_m\}$, as seen in Figure 3. Then, an embeddable position $p_k \in P = \{p_1, p_2, \cdots, p_l\}$, where the coefficient value is adjusted to be a peak, is selected to be located at (x_k, y_k) described by:

$$(x_k, y_k) = (r_i \cos\theta_j, r_i \sin\theta_j) \qquad (2.4)$$

where $0<i<n$, $0<j<m$ and $0<k<l = n \times m$.

In addition, we already know that the DFT of a real image has the complex conjugate property (2.2), and that the coefficient values have the symmetric property (2.3). Furthermore, when conducting a watermarking work by changing the coefficient values, we must preserve the positive symmetry [9] in the following way:

$$\begin{aligned} |F(u,v)| &\rightarrow |F(u,v)| + \delta \\ |F(-u,-v)| &\rightarrow |F(-u,-v)| + \delta \end{aligned} \qquad (2.5)$$

where δ is a pre-selected constant. This means that if the value of an embeddable position is changed, the coefficient value of the symmetric location must also be adjusted with the same amount in the mean time.

In the proposed method, let $M(u, v)$ be a coefficient value, which equals $|F(u, v)|$ and $M'(u, v)$ be the modified value. A watermark W is taken to be a serial number in this study and it is converted into a bit stream $W = w_1 w_2 \cdots w_K$, which we call a *watermark bit stream*, with bit length K. Then, when conducting the watermarking work, the value of $M'(u, v)$ is modified to be a peak by the following equation:

$$M'(u,v) = M(u,v) + c \times w_i \qquad (2.6)$$

where c is a pre-selected factor that determines the embedded watermark strength and w_i is the *i-th* bit value (1 or 0) of W, called a *watermark bit*.

During watermarking, because of the property of the DFT coefficients specified in (2.5), we must select a pair of coefficients at (x_k, y_k) and $(-x_k, -y_k)$

and adjust them to be peaks simultaneously. Otherwise, a peak will be counteracted by the symmetric coefficient value after applying the inverse DFT. In addition, if a watermark bit w_k equals "1," the coefficient values of the corresponding embeddable location (x_k, y_k) and its symmetric location $(-x_k, -y_k)$ are adjusted in this study to be peaks to embed the watermark bit by Eq. (2.6). On the contrary, if w_k equals "0," the values of the coefficient pair are not changed.

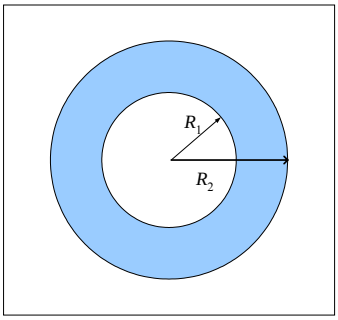


Figure 2 A ring region of middle frequency band.

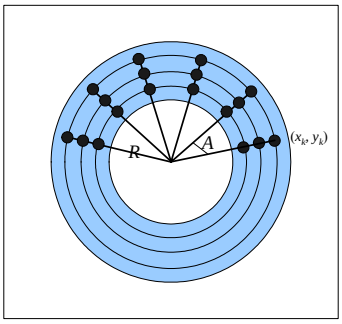


Figure 3 The ring region divided into concentric circles and into angular sectors.

2.4 Proposed Technique for Synchronizing Peak Locations for Protection against Rotation and Scaling Attacks

In order to deal with rotation and scaling attacks, an extra peak P_{syn} is used in this study to synchronize the peak locations and is embedded into the middle frequency band of the DFT domain at the location described as follows:

$$P_{syn}(x, y) = (r\cos\theta_{ori}, r\sin\theta_{ori}) \quad (2.7)$$

where r is selected to be larger than R_2 and θ_{ori} is a pre-selected constant. We adjust the magnitude M of P_{syn} to be $M' = M + c$, where c is a constant mentioned previously.

In the watermark extraction process, we use P_{syn} to calculate the rotated angle of a tampered image which suffered from a rotation attack. Because of the DFT property shown in Table 2.1, if a stego-image is rotated, the location of P_{syn} is changed with the same angle of rotation. We calculate the new angle θ'_{ori} of P_{syn}. The difference $\Delta\theta$ between θ'_{ori} and θ_{ori} can be use to decide whether the stego-image has suffered from a rotation attack. Then, The angles θ'_k of the remaining peaks are obtained. Finally, we use $\Delta\theta$ and θ'_k to reconstruct the original angles θ''_k of the remaining peaks by the following equations:

$$\theta''_k = \theta'_k - \Delta\theta. \quad (2.8)$$

In addition, according to the DFT property shown in Table 1, if an image is scaled, the coefficient values of the DFT domain are almost unaffected. It means that the radiuses of the peaks will not be changed.

3. Watermark Embedding Process

As mentioned previously, a watermark used for image ownership protection is assumed to be a serial number in this study, and the watermark is transformed into a watermark bit stream. In this section, the process of embedding a watermark bit stream in a color image will be described.

3.1 Embedding of Watermarks

In the proposed watermark embedding process, we use the two channels of red and blue to embed a watermark bit stream in the DFT domain according to the idea described in Section 2.2. And the middle band area of the Fourier spectrum is divided into several concentric circles. Then, the watermark bit stream is embedded in the region of the concentric circles.

Furthermore, the watermark bit stream is divided into two halves to be embedded in the red and blue color channels, respectively. For either channel, the spatial domain is transformed into the frequency domain by the DFT. In the middle

band of the DFT domain, locations that can be use to create peaks are decided according to the scheme described in Section 2.3. Then, we can get pairs of locations (x_k, y_k) and $(-x_k, -y_k)$. Using the watermark bit stream W, if a bit wk of W equals "1," coefficient values of the corresponding embeddable positions (x_k, y_k) and $(-x_k, -y_k)$ are adjusted to be peaks by Eq. (2.6) to embed a watermark bit. On the contrary, if wk equals "0", the corresponding coefficient values are not changed. In addition, a synchronization peak is also embedded into the middle frequencies according to the scheme described in Section 2.4.

3.2 Detailed Algorithm

The inputs to the proposed watermark embedding process are a color image C and a watermark W. The output is a stego-image S. The process can be briefly expressed as an algorithm as follows.

Algorithm 1: *Watermark embedding process*.
Input: A given color image *C* and a watermark *W*.
Output: A stego-image *S*.
Steps.

1. Transform the red and blue channels of *C* into the frequency domain by the DFT to get C'_{red} and C'_{blue}.
2. Transform *W* into binary form $W = w_1 w_2 \cdots w_{2l}$ with length $2l$, and divide *W* equally into two parts $W_{\text{red}} = w_1 w_2 \cdots w_l$ and $W_{\text{blue}} = w_{l+1} \cdots w_{2l}$.
3. Embed W_{red} and W_{blue} into C'_{red} and C'_{blue}, respectively, by performing the following operations.
   3.1 Decide *n* radiuses $R = \{r_1, r_2, \cdots, r_n\}$ of equally-spaced concentric circles in the middle band between two circles with radiuses R_1 and R_2, with $R_1 < R_2$.
   3.2 Decide *m* angles $\Theta = \{\theta_1, \theta_2, \cdots, \theta_m\}$ equally distributed in the range from 0° to 180°. Also, take *l* to be $m \times n$.
   3.3 Obtain *l* positions $P = \{p_1, p_2, \cdots, p_l\}$ with p_k (k = 1, 2, …, l) located at $(r_i \cos\theta_j, r_i \sin\theta_j)$ with $k = (i - 1) \times m + j$, and their *l* symmetric positions $Q = \{q_1, q_2, \cdots, q_l\}$ with q_k located at the symmetric location of p_k, where $1 \le i \le n$, and $1 \le j \le m$.
   3.4 If the value of the watermark bit w_k equals 1, then adjust the pair of the coefficient values located at p_k and q_k to be peaks by Eq. (2.6), where $1 \le k \le l$.
   3.5 Add a synchronization peak P_{syn} according to the scheme described in Section 2.4.
4. Transform the C'_{red} and C'_{blue} back into the spacial domain by the inverse DFT.
5. Take the final result as the desired stego-image *S*.

4. Watermark Embedding Process

In the proposed watermark extraction process, no other information but the stego-image is needed as the input. The watermark can be extracted to verify the copyright. The processes of applying this technique will be described in this section. And a detailed algorithm for the process will be given.

4.1 Extraction of Watermarks

In the proposed watermark extraction process, the red and blue channels of a stego-image are accessed. Each of these two channels is transformed into the DFT domain. Then, the peaks in the middle frequency band of the DFT domain are detected using a pre-selected threshold value *T*. If any DFT coefficient value *M* is larger than *T*, it is judged to be a peak. Because of the symmetric property of the DFT coefficient values specified in Eq. (2.3), we can only detect peaks within the range of the upper-half Fourier spectrum image. After collecting all the peaks, a detected peak with the longest radius is taken to be the synchronization peak, which is then used to synchronize the peak locations, and its angle θ'_{ori} is obtained. Then, we reconstruct the angles of the remaining *h* peaks in $P = \{p_1, p_2, \cdots, p_h\}$ by Eq. (2.8) to get their new locations $P' = \{p'_1,$

$p'_2, \cdots, p'_h\}$.

Also, we separate the ring area of the middle frequency band between two circles with radiuses R'_1 and R'_2, with $R'_1 < R'_2$, into n equally-spaced homocentric circles and into m sectors to make the middle frequency band become l areas D = $\{d_1, d_2, \cdots, d_l\}$, where $l = m \times n$, as seen in Figure 2.5. Then, the P' and D are compared to decide the bits of a watermark bit stream $W = w_1 w_2 \cdots w_l$ by the following way:

$$w_k = \begin{cases} 1 & \text{if certain } p_i' \in d_k, \\ 0 & \text{otherwise}, \end{cases} \quad (4.1)$$

where $0 < k \le l$ and $0 < i \le h$. This means that, if there is a peak within an area d_k, the bit w_k is set to be "1"; otherwise, "0". Finally, transform the bit stream into an integer number as the extracted watermark. This completes the extraction process of the watermark.

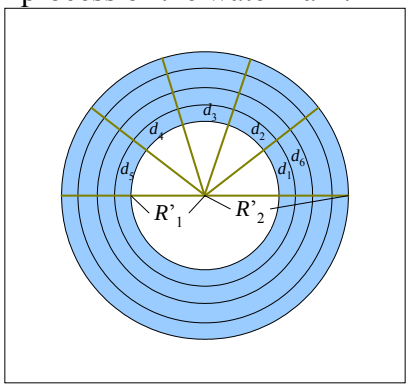


Figure 4 The middle frequency band is separated into concentric circles and into angular sectors.

4.2 Detailed Algorithm

The input to the proposed watermark extraction process includes just a stego-image S. The output is a watermark W that is a serial number embedded presumably in S. The extraction algorithm can be expressed as an algorithm as follows.

Algorithm 2: *Watermark extraction process*.
Input: A stego-image *S*.
Output: A watermark *W*.
Steps.

1. Transform the red and blue color channels of *S* into the DFT domain to get Fourier spectra S'_{red} and S'_{blue}.
2. Detect peaks within the upper-half areas of S'_{red} and S'_{blue}, respectively, by performing the following operations.
   2.1 Use a threshold value *T* to detect peaks in the middle-frequency band. If a coefficient value is larger than *T*, it is considered as a peak.
   2.2 Select a peak with the longest radius to be the synchronization peak, and calculate its changed angle $\Delta\theta$ with respective to the original angle of the synchronization peak.
   2.3 Reconstruct the angles of the remaining *h* peaks by Eq. (2.8) to get their new locations P′ = $\{p'_1, p'_2, \cdots, p'_h\}$.
   2.4 Divide the middle frequency band between R'_1 and R'_2 into *n* equally-spaced concentric circles and into *m* sectors to make the middle band become several areas D = $\{d_1, d_2, \cdots, d_l\}$, where $l = m \times n$.
   2.5 Compare P′ and D to decide the watermark bit stream according to the way specified by Eq. (4.1).
3. Link two watermark bit streams from S'_{red} and S'_{blue} sequentially.
4. Transform the linked watermark bit stream into a serial number.
5. Take the final result as the desired watermark *W*.

5. Experimental Results

Some experimental results of applying the proposed method are shown here. A serial number 877 is transformed into binary form to be a watermark bit stream. The factor c that determines the embedded watermark strength is assigned to be 1.5. Figure 5 shows an input image with size 512×512. And Figure 6(a) shows the stego-image of Figure 5 after embedding the watermark. In addition, Figures 6(b) and (c) show

the corresponding Fourier spectrum image and the detected locations of the peaks marked with red and green marks. The green mark is the synchronization peak. Figure 6(d) show a rotated image of Figure 6(a) and the corresponding Fourier spectrum image and the detected peak locations are shown in Figures 6(e) and (f), respectively. It shows that the Fourier spectrum image have the same angle of rotation with the tampered image. Figure 7(a) shows a scaled image of Figure 6(a) and the corresponding Fourier spectrum image with the detected peak locations are shown in Figure 7(b). The embedded peaks can be successfully detected in our experiments.

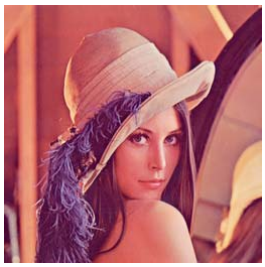
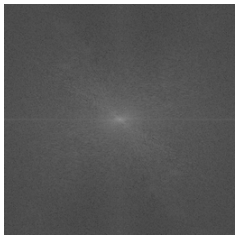


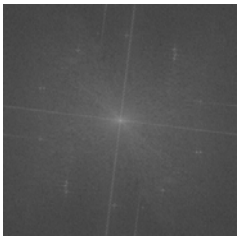
Figure 5 An input image "Lena".

(a)

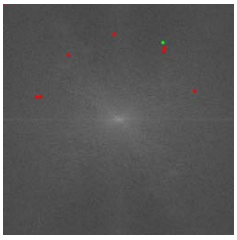
(d)



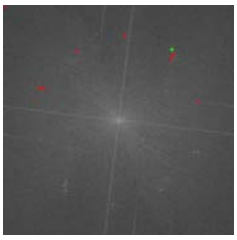
(b)



(e)



(c)

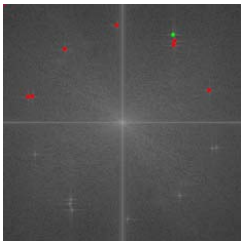


(f)

Figure 6 An output stego-images with the watermark, the tampered image and Fourier spectrum images. (a) Stego-Image "Lena". (b) Fourier spectrum image of (a). (c) Peak locations of (c). (d) Tampered image after rotating 13 degree clockwise. (e) Fourier spectrum image of (d). (f) Peak locations of (e).

Figures 8(a) and (b) show two other color images both with size 512×512. And the corresponding stego-images after embedding the watermark are shown in Figures 8(c) and (d), respectively. The corresponding PSNR values are shown in Table 2, which show that the quality of each of the stego-images is still good. And the embedded watermark is imperceptible by human vision.

(a)



(b)

Figure 7 The tampered image and the Fourier spectrum image. (a) Tampered image after scaling to 90%. (b) Fourier spectrum image of (a) with peak locations.

(a)

(c)


(b) (d)

Figure 8 Input images, and output stego-images with the watermark. (a) Image "Pepper". (b) Image "Jet". (c) and (d) Stego-images after embedding the watermark, respectively.

Table 2. The PSNR values of recovered images after embedding watermarks.

	Lena	Pepper	Jet
PSNR	33.0	33.0	33.0

6. Discussions and Summary

In this chapter, we have proposed a method for embedding a watermark into a color image by coding and synchronization of coefficient-value peak locations in the DFT domain. According to the properties of image coefficients in the DFT domain, we embed the watermark by creating the peaks circularly and symmetrically in the middle frequencies. On the other hand, an extra synchronization peak is added to synchronize the peak locations. The embedded watermark is shown to be robust and can survive the rotation and scaling attacks by the experimental results. The proposed method can achieve the goal to protect image copyright of the owner.

However, in the proposed watermark embedding method, the capacity of hiding data is not large. It is not enough to embed a logo image. In future works, it may be tried to solve this problem.

7. REFERENCES

[1] J. J. K. O'Ruanaidh and T. Pun, "Rotation, scale and translation invariant digital image watermarking," *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, CA USA, Vol. 1, pp. 536-539, Oct. 26-29, 1997.

[2] M. Wu and M. L. Miller, J. A. Bloom, and I. J. Cox, "A rotation, scale and translation resilient public watermark," *Proceedings of 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing*, Phoenix, AZ USA, Vol. 4, pp. 2065, March 15-19, 1999.

[3] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller, and Y.M. Lui, "Rotation, scale, and translation resilient watermarking for images," *IEEE Transactions on Image Processing*, Vol. 10, Issue 5, pp. 767-782, May 2001.

[4] P. C. Su and C. C. Kuo, "An Image Watermarking Scheme to Resist Generalized Geometrical Transformations," *Proceedings of SPIE Conference on Multimedia Systems and Applications III*, Boston, Massachusetts, Vol. 4209, pp. 354-365, November 5-8, 2000.

[5] R.C. Gonzalez and R. E. Woods, "Digital Image Processing," second edition, pp. 155, 2002.

[6] http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Properti-2.html, Properties of Fourier Transforms.

[7] C. Y. Lin and S. F. Chang, "Distortion Modeling and Invariant Extraction for Digital Image Print-and-Scan Process," *Proceeding of International Symposium on Multimedia Information Processing* (ISMIP), Taipei, Taiwan, Dec. 1999.

[8] A. Navarro and J. Tavares, "Joint Source-Channel PCM Image Coding for Binary Symmetric Channels", *Proceeding of International Conference on Signal Processing Applications and Technology*, Orlando-USA, 1999.

[9] J. O'Ruanaidh, W. J. Dowling, and F. M. Boland, "Phase watermarking of digital images," *Proceeding of ICIP'96*, Lausanne, Switzerland, vol. 3, pp. 239–242, Sept. 1996.

利用人類視覺模型和邊界線在大型影像中嵌入強韌性浮水印

Using a Human Visual Model and Boundary Lines for Embedding Robust Watermarks in Large Images

吳大鈞*(Da-Chun Wu)、馮志成(Chin-Chen Feng)
國立高雄第一科技大學 資訊管理研究所
National Kaohsiung First University of Science and Technology
Department of Information Management
dcwu@ccms.nkfust.edu.tw*, u9224804@ccms.nkfust.edu.tw

摘要

在科技快速發展的時代中，人類將資訊和知識數位化以保存智慧的結晶，而如何保護數位資料也成為重要的課題。浮水印為近年保護數位資料常使用的方法。頻率域的浮水印技術較強韌但是其運算時間十分冗長，並不適合大型影像。多數有價值的數位影像尺寸十分龐大。因此我們提出了一個簡單，有效率方法，利用人類視覺模型以及邊界線的定位機制，在空間域加入強韌的浮水印。在擷取浮水印的過程中利用了抽樣的概念，減低了大型影像可能的 冗長運算時間，提升擷取過程的效率。

關鍵字：浮水印、人類視覺模型、邊界線、空間域、大型影像、抽樣。

1. 前言

數位多媒體隨著資訊科技的進步，廣泛被應用在各式各樣的領域，例如資訊的表達或呈現。但由於數位化的結果，資訊交換及傳輸較為容易，因此也造成許多所有權或版權上的爭議。浮水印（watermark）[1]技術在 1993 年被提出，為用來解決這些問題的主要技術。在數位多媒體上加入人類無法感知的浮水印，藉以判斷數位多媒體的來源，或是幫助擁有者確認其真實性。防止惡意侵權行為或不當使用，是浮水印最主要的功用。

浮水印技術可分為空間域和頻率域。在空間域上嵌入浮水印的方法大多做用在 RGB 的色彩模式上，常見的是最低位元（Least Significant Bits: LSBs）方法 [2]，不過 LSB 的強韌性(robustness)較為不足，攻擊者只需透過簡單的數學運算就可以將LSB所加入的訊息破壞。不過由於其簡單，且隱藏資訊量較大，仍然有許多基於 LSB 變形的方法被提出[6][7]。而在頻率域嵌入浮水印的方法需先利用轉換公式如 DCT(離散餘弦轉換)[2]、DFT（離散傅立葉轉換）[2]、或 DWT（離散小波轉換）[2]等，轉換至不同的頻率域中嵌入合適的浮水印。其優點是相對空間域較可以抵抗攻擊。因此多數強韌性浮水印的方法在用頻率域進行，但其轉換演算時間十分冗長，並不適合在大型影像(大於 5000×5000 像素)。其次部份方法採用固定的區塊（如 DCT），當圖檔遭受放大縮小的攻擊時，偵測浮水印難度大為提升。

本文針對大型圖檔提出一個在空間域上嵌入強韌浮水印之新且有效率的方法，利用人類視覺模型（human visual model）[3][4]設計出二種樣版（block patterns）。一種用來嵌入浮水印訊號，而另一種用來加入分界線（boundary lines），藉以幫住定位，讓被嵌入的浮水印更加強韌。在擷取浮水印的過程中不需要參考原圖，並利用抽樣的原理以增加擷取的效率。

在樣版的設計上是不需要原圖就可以把浮水印擷取出來，使得本浮水印嵌入方法為 blind 的方法，在價值上更為提高。除此之外我們在擷取浮水印的過程中利用了抽樣的觀念，以及浮水印的嵌入過程是在空間域上做處理，因而大大的減低了在大型影像上冗長的運算時間，提升了整體嵌入擷取過程的效率。

在本篇文章第二節說明本方法所採用的色彩模型；第三節中介紹本方法使用的人類視覺模型；於第四節說明嵌入浮水印過程；第五節闡述擷取浮水印的過程；第六節說明實驗結果；最後一節為結論。

2. 色彩模型

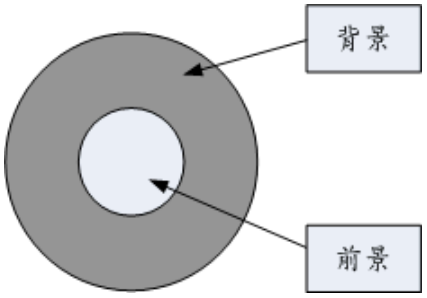
影像色彩模型中，最常使用的為 RGB 色彩模型。RGB 色彩模型之三個通道之像素值分別表現紅色、綠色、藍色色光的強度。透過三種色光的混合來表現出不同的顏色。而在這三種色光中，藍色色光的強度變化視覺較不易察覺，M. Kutter [8][9]提出在藍色通道上做微量的修改，以達到浮水印嵌入或資訊隱藏的目的。不過，一些失真性影像壓縮技術也針用此特性設計以達到高壓縮比（如 JPEG），所以利用前法所隱藏的訊號很容易會被破壞了。在本方法中，我們將 RGB 色彩模型轉換成 YCbCr 色彩模型，Y 代表明亮的程度（luminance），而 Cb 和 Cr 則代表色度 (chrominance)，透過下方的轉換公式可以 將 RGB 色彩模型轉為 YCbCr[2]，

$$\begin{aligned} Y &= 0.299\,R + 0.587\,G + 0.114\,B; \\ Cb &= 0.5 + (B - Y)/2; \\ Cr &= 0.5 + (R - Y)/1.6. \end{aligned} \tag{1}$$

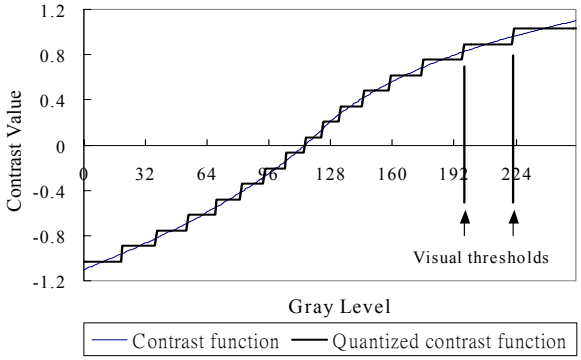
在人類的視覺效果中明亮度是最敏感的部份，也由於它的敏感，在大部份的彩色影像失真壓縮技術（如 JPEG），對其破壞的程度相對上比較小，因此我們選擇在 Y channel 上配合人類視覺模型使在人類的視覺效果不容易察覺的情況下嵌入浮水印。

3. 人類視覺模型

本方法所採用之視覺模型是以 Kuo 和 Chen[3] 所提出亮度對比函數為基礎，利用 Da-Chun Wu 和 Wen-Hsiang Tsai[5]提出量化對比函數訂定出影像之前景(foreground) 亮度改變而人眼無法感受其變化的亮度範圍。圖一為前景和背景的示意圖。圖二為在背景亮度為 60 之量化對比函數及其視覺門檻值（thresholds）的範圍示意圖。圖二中的每一個間距代表，在二個門檻值（如 192 和 224）之間的微量變化是人類的視覺所感覺不出來的。



圖一． 前景色和背景色的意識圖。

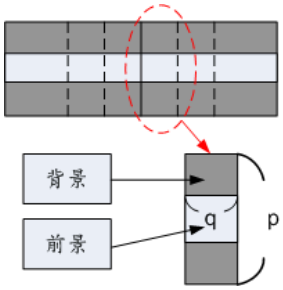


圖二． 背景亮度平均為 60，所呈現之亮度對比函數。

4. 嵌入程序

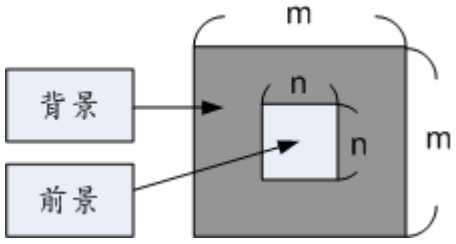
4.1 樣版設計

本方法以前節所提人類視覺模型為基礎，設計兩種樣版。第一種樣版為一個 $p\times1$ 大小的區塊，用來做為邊界線訊號嵌入之用。 $p\times1$ 區塊中包含了一個 $q\times1$ 子區塊，本方法將在 $p\times1$ 區塊中不包含 $q\times1$ 子區塊之 $p-q$ 個像素視為背景之像素； $q\times1$ 子區塊中之 q 個像素視為前景之像素。連續之多個 $p\times1$ 區塊可形成一條邊界線，圖三為邊界線之示意圖。



圖三． 邊界線的示意圖。

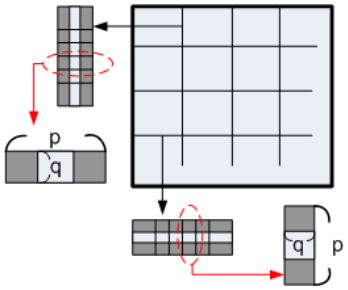
第二種樣版為 $m\times m$ 區塊，以嵌入浮水印資訊的本體。在 $m\times m$ 區塊中包含了一個 $n\times n$ 子區塊。本方法將在 $m\times m$ 區塊中不包含 $n\times n$ 子區塊之 m^2-n^2 個像素視為背景之像素； $n\times n$ 子區塊中之 n^2 個像素像素視為前景之像素。圖四為 $m\times m$ 區塊的示意圖。



圖四． $m\times m$ 區塊的示意圖。

4.2 邊界線嵌入

在嵌入浮水印資訊前首先需嵌入邊界線。在影像之水平及垂直方向分別嵌入多條邊界線，各邊界線以多個 $p\times1$ 區塊形成。邊界線間距為固定值 r，水平與垂直邊界線所形成 $r\times r$ 區塊，可供嵌入浮水印資訊使用。圖五說明不同方向嵌入的邊界線及其構成樣版的方向。



圖五． 垂直和水平的邊界線及樣版。

組成邊界之 $p\times1$ 區塊均各嵌入相同之邊界線訊號。邊界線訊號嵌入是在每個 $p\times1$ 區塊中隱藏一個位元的資訊。 $p\times1$ 區塊中央有一個 $q\times1$ 的區塊，令在 $p\times1$ 區塊中不包含 $q\times1$ 區塊之 $p-q$ 個背景像素的像素值分別為 $b_i, i=1..p-q$；$q\times1$ 區塊中之 q 個前景像素的像素值為 $f_i, i=1..q$。而背景之所有像素的亮度平均值為

$$\bar{y}_b=\frac{\sum_{i=1}^{p-q}b_i}{p-q}\ ; \tag{2}$$

前景之所有像素的亮度平均值為

$$\bar{y}_f=\frac{\sum_{i=1}^{q}f_i}{q}\ 。 \tag{3}$$

嵌入邊界線位元訊息的方式為調整前景像素之亮度值，方法為

$$f_i'=\min(f_i-\bar{y}_f+\bar{y}_b+T,255), \tag{4}$$

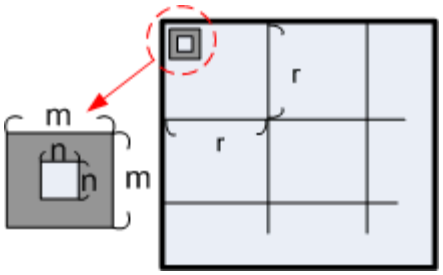
其中 f_i' 為 f_i 調整後之像素值，$i=1..q$，T 為可修改的亮度值。T 值的大小設定與量化對比函數之視覺門檻值及背景區域之像素值標準差 σ_b 有關。σ_b 可表示為

$$\sigma_b=\sqrt{\frac{\sum_{i=1}^{p-q}(b_i-\bar{y}_b)^2}{p-q}}, \tag{5}$$

當 σ_b 的值越大時可 T 值相對變大；當 σ_b 的值越小時 T 值相對變小。這樣可以在不影響視覺的前提下在影像中嵌入強韌邊界線訊息。

4.3 浮水印嵌入

完成邊界線嵌入後，接著在水平與垂直邊界線所分隔的出的 $r\times r$ 區塊中進行浮水印資訊嵌入。嵌入之方法是將 $r\times r$ 區塊分割成不重疊之 $m\times m$ 區塊，在每個 $m\times m$ 的區塊中欲隱藏一個位元的浮水印訊號。 $m\times m$ 區塊中央有一個 $n\times n$ 的子區塊，圖六為 $r\times r$ 區塊分割成不重疊之 $m\times m$ 區塊的意識圖

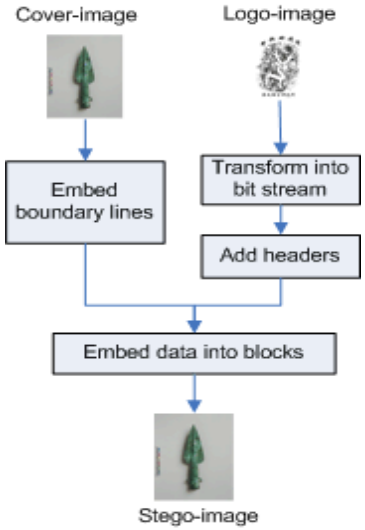


圖六． $r\times r$ 區塊分割成不重疊之 $m\times m$ 區塊的意識圖。

令在 $m\times m$ 區塊中不包含 $n\times n$ 子區塊之 m^2-n^2 個背景像素的像素值分別為 $b_j,\ j=1..m^2-n^2$ ； $n\times n$ 子區塊中之 n^2 個前景像素的像素值為 $f_j,\ j=1..n^2$ 。嵌入浮水印位元資訊 w 的方法為

$$f_i'=\begin{cases}\min(f_i-\bar{y}_f+\bar{y}_b+T,255) & if\ w=1;\\ \max(f_i-\bar{y}_f+\bar{y}_b-T,0) & if\ w=0.\end{cases} \quad (6)$$

將要嵌入的浮水印訊息重複嵌入至整個影像中。圖七為說明浮水印嵌入的流程圖。由於每條邊界線的相隔距離固定，所以每個 $r\times r$ 區塊中可嵌入訊息量相同。如果浮水印訊息過大，則需使用多個 $r\times r$ 區塊來嵌入一份浮水印。本方法在每個區塊中先嵌入額外的編碼訊息，以使在擷取過程中可以整合浮水印資訊



圖七． 浮水印嵌入流程圖。

5. 擷取程序

5.1 邊界線偵測

本方法首先需從影像中找出邊界線的正確位置才可以進而擷取所隱藏之浮水印訊息。邊界線訊號之擷取是利用 $p\times 1$ 區塊中內含 $q\times 1$ 子區塊之樣板與函數 $D(p,q)$ 擷取出邊界線訊號， $D(p,q)$ 可以表示為

$$D(p,q)=\begin{cases}yes, & \bar{y}_f\geq\bar{y}_b;\\ no, & \bar{y}_f<\bar{y}_b,\end{cases} \quad (7)$$

其中 $\bar{y}_f$ 為在 $p\times 1$ 區塊中不包含 $q\times 1$ 區塊之前景像素的像素值平均， $\bar{y}_b$ 為 $q\times 1$ 區塊所影像素的像素值平均。

本方法分別在水平及垂直方向循序擷取邊界線訊息以偵測邊界線。當水平或垂直方向某行、列發現邊界線訊號個數大於門欄 t，則將其視為邊界線。大型影像的像素數量十分龐大，因此本方法以亂數抽樣方式來加快偵測邊界線的速度。除此之外，偵測邊界線時由於影像可能已遭受攻擊，本方法利用所有相鄰邊界線之距離值投票（voting）來排除誤判之邊界線。

偵側出邊界線後，透過邊界線的資訊可以取得影隱藏浮水印之區塊位置以及區塊的大小，倘若區塊曾遭受尺寸放大縮小的攻擊，則先將區塊縮放回原先設定之固定尺寸大小，以利進行浮水印資訊的擷取。

5.2 浮水印擷取

取得邊界線所圍成 $r\times r$ 區塊的位置後，即可在每個區塊中擷取出所隱藏資訊。隱藏資訊之擷取是透過在 $m\times m$ 區塊中內含 $n\times n$ 子區塊的樣版，擷取出其中之位元訊號 w。令在 $m\times m$ 區塊中不包含 $n\times n$ 區塊之 m^2-n^2 個像素的像素值為 $b_j, j=1..m^2-n^2$； $n\times n$ 區塊中之 n^2 個像素像素的像素值為 $f_j, j=1..n^2$。w 之擷取的方法為

$$w=\begin{cases}1, & \dfrac{\sum_{j=1}^{n^2} f_j}{n^2}\geq\dfrac{\sum_{j=1}^{m^2-n^2} b_j}{m^2-n^2};\\ 0, & \dfrac{\sum_{j=0}^{n^2} f_j}{n^2}<\dfrac{\sum_{j=1}^{m^2-n^2} b_j}{m^2-n^2}.\end{cases} \tag{8}$$

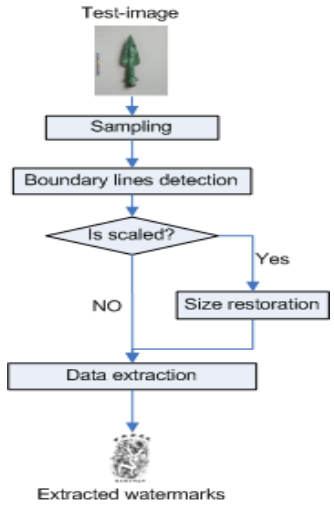
本方法在每個 $m\times m$ 區塊中取得的隱藏資訊，其中包含標頭資訊與浮水印訊號，本方法利用多數決的投票機制，來修正可能的錯誤，以完成擷取浮水印的程序。圖八為浮水印擷取流程圖。

6. 實驗結果

本論文以『中央研究院 2004 浮水印技術評比』所提供的兩 8000×10013 大型全彩影像檔‘銅矛’與‘酒器’(圖九)實驗。浮水印影像(圖十)為 161×229 之黑白影像，使用電腦為 Pentium 4-1.8 GHz。嵌入時間約需 4 分 35 秒，擷取時間約需 4 分 40 秒。利用 PSNR（Peak Signal to Noise Ratios）值來評估影像嵌入浮水印後之品質。擷取出之浮水印影像，以 BER(Bit Error Rate）來衡量技術之強韌性，BER 的定義為

$$\text{BER} = \text{Number of Wrong Extracted Bits} \,/\, \text{Number of Original Watermark Bits.} \tag{9}$$

圖十一為圖十嵌入浮水印後之結果影像，PSNR 值分別為 45.99dB 與 45.19dB 。圖十二為圖十一影像分別遭後 Jpeg 壓縮、鏡像、更模糊化、旋轉、剪裁連續攻擊後之結果影像與由此結果影像擷取出之浮水印影像及其 BER 值。圖十三為圖十一影像受到尺寸縮放的攻擊的結果影像與從其中所擷取出之浮水印影像及其 BER 值。



圖八． 浮水印擷取流程圖。

7. 結論

本文提出利用人類視覺模型配合邊界線的定位機制於大型影像中加入強韌浮水印的方法。因為本方法在空間域操作，擷取步驟又利用抽樣的概念，大大降低了在大型影像上嵌入、擷取浮水印時可能的冗長運算，使得執行很有效率。

8. 致謝

感謝國立高雄第一科技大學資料隱藏與數位版權管理實驗室的所有成員。使本方法於中央研究院所主辦之『2004 浮水印技術評比』，獲得大型影像組第一名。

參考文獻

1. A.Z. Tirkel, G.A. Rankin, R.M. van Schyndel, W.J. Ho, N.R.A. Mee, C.F. Osborne. “Electronic Water Mark.” DICTA-93. Macquarie University, Sydney, pp. 666-672 , Decmeber 1993.
2. Dugelay JL, Roche S, “A survey of current watermarking techniques. In:Katzenbeisser S, Petitcolas FA(eds) Information hiding techniques for steganography and digital watermarking.” Artech House, Norwood, MA, pp 121-148, 2000.
3. Kuo, C.H., Chen, C.H., “A prequantizer with the human visual effect for the DPCM.” Signal Processing: Image Communication 8, pp 433-442, 1996.
4. Kuo, C.H., Chen, C.H. “A vector quantizer scheme using prequantizers of human visual effect.” Signal Processing: Image Communication 12, pp 13-21, 1998.
5. Da-Chun Wu, Wen-Hsiang Tsai, “Embedding of any type of data in image based on human visual model and multiple-based number conversion.” Elsevier Science B.V., pp 1511-1517, 1999.
6. Y.K.Lee, L.H.Chen, “A high capacity image steganographic model.” IEE Proceedings Vision, Image and Signal Processing, pp 288- 294, 1999.
7. Y.K.Lee, L.H.Chen, “An adaptive image steganographic model based on minimun-error LSB replacement.” Proceedings of the Ninth National Conference on Information Security, pp 8-15, May.14-15, 1999.
8. M. Kutter, F. Jordan, and F. Bossen, “Digital watermarking of color images using amplitude modulation.” J. Electron. Imag., vol. 7, no. 2, pp 326-332, 1998.
9. Kutter, M., Winkler, S., “A vision-based masking model for spread-spectrum image.” Image Processing, IEEE Transactions on Jan. pp 16 - 25, 200

(a) (b)

圖九． 實驗所採用的大型影像資料。(a) 銅矛；(b) 酒器。

圖十． 黑白浮水印影像。

(a) (b)

圖十一． 圖九(a)、(b)分別嵌入浮水印後的結果影像。(a) 銅矛(PSNR=45.99dB)；(b) 酒器(PSNR=45.19dB)。

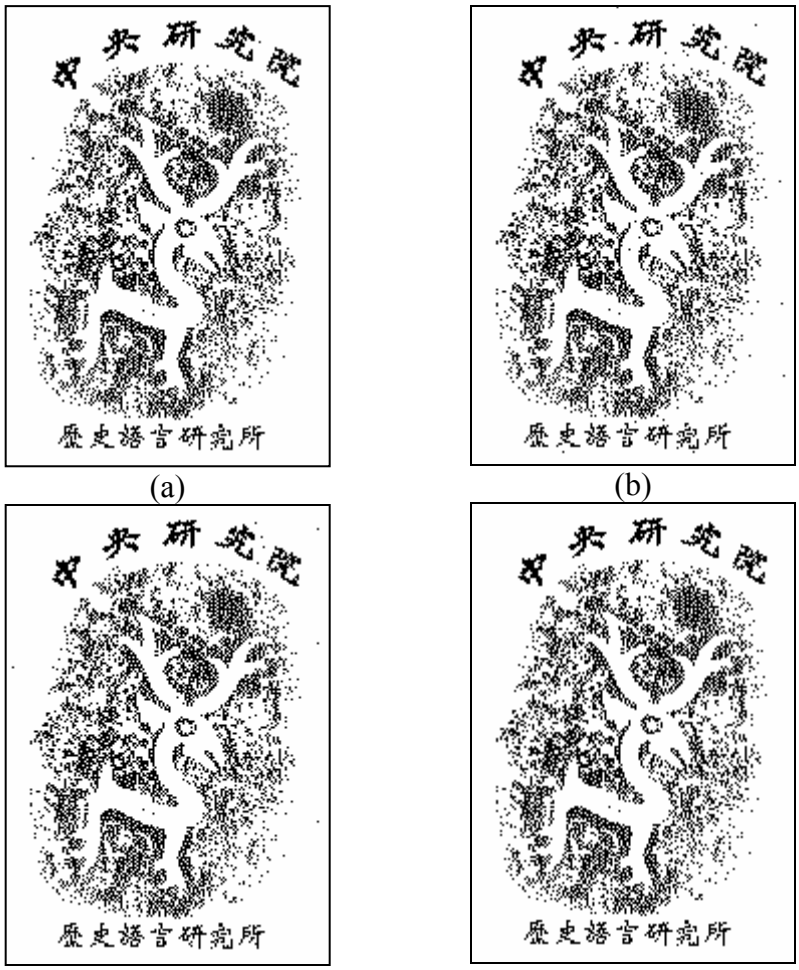
(a)

(b)

(c)

(d)

圖十二． (a) 圖十一 (a)受到多重連續攻擊（Jpeg、鏡像、更模糊化、旋轉、剪裁）後之結果影像；(b) 圖十一(b)受到多重連續攻擊（Jpeg、鏡像、更模糊化、旋轉、剪裁）後之結果影像；(c) 由圖十二(a)擷取出來的 浮水印影像(BER 為 0.001439)；(d) 由圖十二(b)擷取出來的浮水印影像(BER 為 0)。



圖十三．(a) 圖十一(a)受到尺寸放大 20%攻擊後所擷取出的浮水印影像(BER 為 0.000325)；(b) 圖十一(a)受到尺寸縮小 30%攻擊後所擷取出的浮水印影像(BER 為 0.002361)；(c) 圖十一(b)受到尺寸放大 20%攻擊後所擷取出的浮水印影像(BER 為 0.000922)；(d) 圖十一(b)受到尺寸縮小 50%攻擊後所擷取出的浮水印影像(BER 為 0)。

附件二

交通大學影像認證中心

註冊證明書

本影像認證中心於民國___年___月___日，茲收到_____單位註冊之圖檔共幾張，以及添加浮水印之軟體名稱_____。

經過註冊程序後，本中心於民國____年____月____日寄還貴單位圖檔，以及註冊後圖檔之編號，請貴單位務必留存圖檔編號，往後若遇到圖檔版權糾紛情事，貴單位將可攜帶此證明書與註冊圖檔編號來本中心予以認證，證明圖檔版權。

本聲明書一式兩份，由本影像認證中心及註冊單位各留存一份為憑。

單位/計畫名稱			
主持人姓名			
聯絡電話			
傳真			
地址			
電子郵件信箱			
添加浮水印之軟體名稱			
BMP 圖檔格式		張數	
TIFF 圖檔格式		張數	
JPEG 圖檔格式		張數	
GIF 圖檔格式		張數	